УДК 519.85

Т.Є. РОМАНОВА, П.І. СТЕЦЮК, А.М. ЧУГАЙ, С.Б. ШЕХОВЦОВ

ТЕХНОЛОГІЇ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ДЛЯ РОЗВ'ЯЗАННЯ ОПТИМІЗАЦІЙНИХ ЗАДАЧ ГЕОМЕТРИЧНОГО ПРОЄКТУВАННЯ

Анотація. Описано застосування технологій паралельних обчислень у системах зі спільною пам'яттю та розподіленою пам'яттю для розв'язання оптимізаційних задач геометричного проєктування. Перша технологія ґрунтується на властивостях максимінних *phi*-функцій для складених геометричних об'єктів, а в другій технології використано стратегію мультистарту та методи мінімізації негладких функцій. Це дало змогу в декілька разів зменшити витрати часу під час пошуку локально оптимальних розміщень 2D та 3D об'єктів та отримати кращі результати за значенням цільової функції.

Ключові слова: розміщення геометричних об'єктів, *phi*-функція, нелінійне програмування, паралельні обчислення.

ВСТУП

У багатьох пріоритетних галузях науки та техніки з-поміж задач, що інтенсивно досліджуються протягом останніх десятиліть, можна виділити задачі комп'ютерного моделювання оптимального розміщення об'єктів різної природи. Ці задачі стають дедалі більш затребуваними через те, що заміна натурних експериментів на комп'ютерне моделювання дає змогу суттєво заощаджувати матеріальні ресурси та час.

На сьогодні розроблено широкий спектр методів (здебільшого евристик та метаевристик) для розв'язання NP-складних оптимізаційних задач розміщення геометричних 2D та 3D об'єктів [1–7]. Метод *phi*-функцій Стояна [8–11] дає змогу будувати математичні моделі задач геометричного проєктування у вигляді задач нелінійного програмування [11–20].

Різноманітність цільових функцій, форм об'єктів і контейнерів, геометричних та механічних обмежень зумовлює створення множини оптимізаційних задач розміщення геометричних об'єктів, що охоплює широкий спектр актуальних наукових та прикладних задач, які виникають у машинобудуванні, хімічній промисловості, матеріалознавстві, сучасній біології, нанотехнологіях, робототехніці, системах розпізнавання образів, адитивних технологіях та багатьох інших галузях [21–24].

Слід зауважити, що під час розв'язання задач цього класу виконується оброблення великого обсягу складної геометричної, аналітичної та логічної інформації. У зв'язку з цим актуальною проблемою є скорочення часу обчислень з метою ефективного застосування сучасного програмного забезпечення для розв'язання задач нелінійного програмування.

У статті запропоновано застосування технологій паралельних обчислень у системах зі спільною пам'яттю та розподіленою пам'яттю для розв'язання оптимізаційних задач геометричного проєктування.

ПОСТАНОВКА ЗАДАЧІ ТА МАТЕМАТИЧНА МОДЕЛЬ

Розглянемо задачу розміщення геометричних об'єктів у такій постановці.

Розмістити множину геометричних об'єктів $T_i \subset R^d$, $d = 2, 3$, $i \in I_N = \{1, 2, \ldots, N\}$, у контейнер Ω таким чином, щоб виконувались обмеження на розміщення об'єктів і задана цільова функція досягала екстремального значення.

Допускаються неперервні переміщення (трансляції) та повороти об'єктів. Обмеження на розміщення об'єктів передбачають розміщення об'єктів у межах

контейнера, неперетин об'єктів або мінімальні допустимі відстані між об'єктами, зони заборони, обмеження на повороти об'єктів, а також інші технологічні обмеження (наприклад, допустимі розміри контейнеру, діапазони кутів повороту, умови збереження балансу системи).

Позначимо вектор усіх змінних $u = (p, u_1, u_2, \dots, u_N) \in R^\sigma$, де p — вектор змінних метричних характеристик контейнера Ω, $u_i = (\upsilon_i, \theta_i)$ — вектор змінних параметрів розміщення (вектор руху) об'єкта T_i, $i \in I_N$, R^σ — арифметичний евклідовий простір σ-розмірності ($\sigma = 3N + |\,p\,|$ для $d = 2$ та $\sigma = 6N + |\,p\,|$ для $d = 3$).

Математична модель оптимізаційної задачі розміщення має вигляд

$$\min \kappa(u),\ u \in W \subset R^\sigma, \tag{1}$$

$$W = \{u \in R^\sigma : \Phi'_\tau(u) \ge 0, \Phi''_i(u) \ge 0, \varphi_t(u) \ge 0, \tau = 1, \dots, \Xi, i = 1, \dots, N, t = 1, \dots, M\}, \tag{2}$$

де $\kappa(u)$ — цільова функція, $\Phi'_\tau(u)$ — *phi*-функція для опису умови неперетину об'єктів A і B ($\operatorname{int} A \bigcap \operatorname{int} B = \varnothing$), або псевдонормалізована *phi*-функція для опису умови знаходження об'єктів A і B на мінімально допустимій відстані $\rho > 0$ ($\operatorname{dist}(A, B) \ge \rho$), де $\operatorname{int}(\cdot)$ є внутрішність об'єкта $(\cdot)$, $\operatorname{dist}(A, B) = \min\limits_{a \in A,\, b \in B} d(a, b)$, $d(a, b)$ — евклідова відстань між точками a і b, $\Xi = N(N-1)/2$; функція Φ''_i — *phi*-функція для опису умови включення об'єкта A в контейнер Ω ($A \subset \Omega \Leftrightarrow \operatorname{int} A \bigcap \Omega^* = \varnothing$, $\Omega^* = R^d \setminus \operatorname{int} \Omega$); $\varphi_t \ge 0$, $t = 1, \dots, M$, являє собою систему додаткових обмежень на значення компонентів вектора u, якщо такі є, за умови, що кожна функція φ_t є гладкою. Зауважимо, що кожну *phi*-функцію в (2) визначено в термінах операцій максимуму та/або мінімуму гладких функцій. Функцію

$$\Upsilon(u) = \min \{\Phi'_\tau(u), \Phi''_i(u), \varphi_t(u), \tau = 1, \dots, \Xi, i = 1, \dots, N, t = 1, \dots, M\} \tag{3}$$

називають функцією розміщення. Функція (3) є неперервною, кусково-гладкою і залежить від усіх змінних $(p, u_1, u_2, \dots, u_N)$ задачі (1), (2). Зазначимо, що $\Upsilon(u) \ge 0$, якщо $\Phi'_\tau(u) \ge 0$ для всіх $\tau = 1, \dots, \Xi$ і $\Phi''_i(u) \ge 0$ для всіх $i = 1, \dots, N$ і $\varphi_t(u) \ge 0$ для всіх $t = 1, \dots, M$.

Задача оптимізації (1), (2) є NP-складною задачею нелінійного програмування. Множина допустимих розв'язків W має складну структуру: це, в загальному випадку, незв'язна множина, кожна компонента якої є багатозв'язною, а межа множини W складається з нелінійних поверхонь, що містять долини, яри тощо.

Нехай $\Re$ — множина базових об'єктів (двовимірних або тривимірних). Як відомо [10], *phi*-функція для об'єктів $A = \bigcup\limits_{i=1}^{n_A} A_i$ і $B = \bigcup\limits_{j=1}^{n_B} B_j$, $A_i, B_j \in \Re$, має вигляд

$$\Phi^{AB}(u) = \min \{\Phi_{ij}(u), i = 1, \dots, n_A, j = 1, \dots, n_B\}, \tag{4}$$

де $\Phi_{ij}(u)$ — *phi*-функція для пари базових об'єктів A_i і B_j. Представимо співвідношення (4) таким чином:

$$\Phi^{AB}(u) = \min \{\Phi_k(u),\ k = 1, \dots, n_A \cdot n_B\}, \tag{5}$$

де $\Phi_k(u)$ є однією з базових *phi*-функцій.

Позначимо всі базові *phi*-функції в (3) $\Phi_k(u)$, $k = 1, \dots, n$, $n = \sum\limits_{i=1}^{N-1} \sum\limits_{j=i+1}^{N} n_i \cdot n_j + \sum\limits_{i=1}^{N} n_i$, де $\sum\limits_{i=1}^{N-1} \sum\limits_{j=i+1}^{N} n_i \cdot n_j$ — кількість усіх базових *phi*-функцій $\Phi'_\tau(u)$,

$\tau = 1, 2, \ldots, \Xi$, $\Xi = \dfrac{N(N-1)}{2}$; $\sum\limits_{i=1}^{N} n_i$ — кількість базових *phi*-функцій $\Phi_i''(u)$, $i = 1, \ldots, N$, n_i — кількість базових об'єктів, які утворюють складений об'єкт T_i, n_j — кількість базових об'єктів, що утворюють складений об'єкт T_j.

Таким чином, функція розміщення (3) може бути визначена у вигляді

$$\Upsilon(u) = \min \{\Phi_k(u), k = 1, \ldots, n, \varphi_t, t = 1, \ldots, M\}. \tag{6}$$

Наведемо співвідношення (2) в еквівалентній формі

$$W = \{u \in R^{\sigma} : \Upsilon(u) \geq 0\}. \tag{7}$$

Зауважимо, що кожна базова *phi*-функція Φ_k в (6) може бути перетворена за допомогою формул алгебри логіки і матиме вигляд

$$\Phi_k(u) = \max_{i=1,\ldots,\eta_k} f_i^k(u) = \max_{i=1,\ldots,\eta_k} \min_{j=1,\ldots,J_i^k} f_{ij}^k(u), \tag{8}$$

де $f_{ij}^k \in \{f\}$. Надалі $\{f\}$ буде визначати сімейство гладких функцій.

Для виконання умови $\Upsilon(u) \geq 0$ необхідно, щоб $\Phi_k(u) \geq 0$ для всіх $k = 1, 2, \ldots, \eta$, де $\Phi_k(u)$ має вигляд (8).

Приклад 1. Нехай базову *phi*-функцію $\Phi_k(u)$ задано у вигляді

$$\varphi(u) = \min \{\max \{\varphi_1(u), \varphi_2(u), \varphi_3(u)\}, \max\{\varphi_4(u), \varphi_5(u)\}\},$$

де $\varphi_j \in \{f\}$, $j = 1, 2, 3, 4, 5$.

Функція може бути подана в еквівалентній формі

$$\varphi(u) = \max \{f_1(u), f_2(u), f_3(u), f_4(u), f_5(u), f_6(u)\},$$

де

$$f_1(u) = \min \{\varphi_1(u), \varphi_4(u)\},\ f_2(u) = \min \{\varphi_2(u), \varphi_4(u)\},$$

$$f_3(u) = \min \{\varphi_3(u), \varphi_4(u)\},\ f_4(u) = \min \{\varphi_1(u), \varphi_5(u)\},$$

$$f_5(u) = \min \{\varphi_2(u), \varphi_5(u)\},\ f_6(u) = \min \{\varphi_3(u), \varphi_5(u)\},$$

або відповідно до (8)

$$f_1(u) = \min \{f_{11}(u), f_{12}(u)\},\ f_2(u) = \min \{f_{21}(u), f_{22}(u)\},$$

$$f_3(u) = \min \{f_{31}(u), f_{32}(u)\},\ f_4(u) = \min \{f_{41}(u), f_{42}(u)\},$$

$$f_5(u) = \min \{f_{51}(u), f_{52}(u)\},\ f_6(u) = \min \{f_{61}(u), f_{62}(u)\},$$

де $f_{11}(u) = f_{41}(u) = \varphi_1(u)$, $f_{21}(u) = f_{51}(u) = \varphi_2(u)$, $f_{31}(u) = f_{61}(u) = \varphi_3(u)$, $f_{12}(u) = f_{22}(u) = f_{32}(u) = \varphi_4(u)$, $f_{42}(u) = f_{52}(u) = f_{62}(u) = \varphi_5(u)$), тобто

$$\varphi(u) = \max \{\min \{\varphi_1(u), \varphi_4(u)\}, \min \{\varphi_2(u), \varphi_4(u)\}, \min \{\varphi_3(u), \varphi_4(u)\},$$
$$\min \{\varphi_1(u), \varphi_5(u)\}, \min \{\varphi_2(u), \varphi_5(u)\}, \min \{\varphi_3(u), \varphi_5(u)\}\}.$$

На основі виразу (8) функція $\Upsilon(u)$, яка визначається формулою (3), може бути перетворена в еквівалентну форму

$$\Upsilon(u) = \max \{\Upsilon_s(u), s = 1, 2, \ldots, \eta\}, \tag{9}$$

$$\Upsilon_s(u) = \min \{f_s^i(u), i \in I_s\},\ f_s^i \in \{f\}. \tag{10}$$

Зазначимо, що $\min\limits_{j=1,\ldots,J_i^k} f_{ij}^k(u) \geq 0$ є еквівалентним $f_{ij}^k(u) \geq 0$ для всіх j, а $\max\limits_{i=1,\ldots,\eta_k} f_i^k(u) \geq 0$ означає, що принаймні одна з нерівностей, скажімо $f_{i^*}^k(u) \geq 0$, повинна бути виконана. Враховуючи вирази (9), (10), множину допус-

тимих розв'язків W (7), можна представити у вигляді

$$W = W_1 \cup \ldots \cup W_s \cup \ldots \cup W_\eta, \tag{11}$$

де $W_s = \{u \in R^\sigma : \Upsilon_s(u) \geq 0\}$, $\Upsilon_s(u)$ визначається за формулою (10), а нерівність $\Upsilon_s(u) \geq 0$ є еквівалентною системі нерівностей $f_s^i \geq 0$, $i \in I_s$.

Приклад 2. Візьмемо множину W^φ, що задається нерівністю $\varphi(u) \geq 0$, де $\varphi(u)$ має форму, визначену в прикладі 1. Область W^φ може бути представлена у вигляді об'єднання підмножин W_1, W_2, W_3, W_4, W_5, W_6, та буде описуватися відповідними нерівностями з гладкими функціями, тобто

$$\min\{\varphi_1(u), \varphi_4(u)\} \geq 0, \ \min\{\varphi_2(u), \varphi_4(u)\} \geq 0, \ \min\{\varphi_3(u), \varphi_4(u)\} \geq 0,$$
$$\min\{\varphi_1(u), \varphi_5(u)\} \geq 0, \ \min\{\varphi_2(u), \varphi_5(u)\} \geq 0, \ \min\{\varphi_3(u), \varphi_5(u)\} \geq 0.$$

Це означає, що можна реалізувати розпаралелювання для розв'язання задачі (1), (2).

Ґрунтуючись на виразі (11), задачу (1), (2) можна звести до такої задачі:

$$\kappa(u^*) = \min\{\kappa(u_1^*), \ldots, \kappa(u_s^*), \ldots, \kappa(u_\eta^*)\}, \tag{12}$$

де

$$\kappa(u_s^*) = \min_{u \in W_s \subset R^\sigma} \kappa(u), \ s = 1, 2, \ldots, \eta. \tag{13}$$

Очевидно, що глобальний мінімум задачі (12), (13) можна отримати і довести шляхом точного розв'язання всіх підзадач, визначених у (13). Кожна з підзадач у (13) в загальному випадку є задачею нелінійного програмування з гладкими функціями і може бути принаймні теоретично розв'язана за допомогою сучасного програмного забезпечення для розв'язання задач нелінійного програмування.

Для розпаралелювання процесу пошуку локальних екстремумів задачі (1), (2) створимо генератор множини допустимих розв'язків W (7), що дасть змогу автоматично генерувати непорожню підмножину W_s для підзадачі (13). Для цього скористаємося поняттям «дерево розв'язків».

ГЕНЕРАТОР МНОЖИНИ ДОПУСТИМИХ РОЗВ'ЯЗКІВ W

Покажемо, як можна отримати підмножини W_s, $s = 1, 2, \ldots, \eta$, з множини допустимих розв'язків W в (11) за допомогою так званого «дерева розв'язків» $\Im^\Upsilon$, використовуючи функцію (9). Кожна кінцева вершина v_s з дерева відповідає нерівності $\Upsilon_s(u) \geq 0$, $s = 1, 2, \ldots, \eta$, де функція $\Upsilon_s(u)$ визначається за формулою (10).

Дерево розв'язків $\Im^\Upsilon$ для множини допустимих розв'язків W (7) будують у такий спосіб. Корінь дерева відповідає нерівності $\varphi(u) = \min\{\varphi_t(u), t = 1, \ldots, M\} \geq 0$, де $\varphi_t(u) \in \{f\}$. Для кожної нерівності $\Phi_k(u) \geq 0$ будуємо дерево, яке назвемо базовим *phi*-деревом. Позначимо η_k кількість кінцевих вершин базового *phi*-дерева. Кожна кінцева вершина відповідає системі нерівностей $f_i^k(u) \geq 0$. На першому рівні $\Im^\Upsilon$ маємо $\tau_1 = \eta_1$ вершин, де η_1 — кількість кінцевих вершин *phi*-дерева, які відповідають $\Phi_1(u) \geq 0$, де $\Phi_1(u) = \max\limits_{i=1,\ldots,\eta_1} f_i^1(u)$, $f_i^1(u) = \min\limits_{j=1,\ldots,J_i^1} f_{ij}^1(u)$ відповідно до (8). Кожній вершині першого рівня відповідають нерівності системи $\varphi(u) \geq 0$, $f_{i_1}^1(u) \geq 0$. Для побудови другого рівня $\Im^\Upsilon$ необхідно додати η_2 вершин *phi*-дерева, які відповідають $\Phi_2(u) \geq 0$, до кожної вершини першого рівня, де $\Phi_2(u) = \max\limits_{i=1,\ldots,\eta_2} f_i^2(u)$, $f_i^2(u) = \min\limits_{j=1,\ldots,J_i^2} f_{ij}^2(u)$. Кількість вершин другого рівня $\Im^\Upsilon$ тепер дорівнює $\tau_2 = \eta_1 \cdot \eta_2$. Кожній вершині другого рівня відповіда-

ють нерівності системи $\varphi(u) \ge 0, f_{i_1}^1(u) \ge 0, f_{i_2}^2(u) \ge 0$. Для побудови k-рівня $\Im^{\Upsilon}$ необхідно додати η_k вершин *phi*-дерева, які відповідають $\Phi_k(u) \ge 0$, $\Phi_k(u) = \max\limits_{i=1,\dots,\eta_k} f_i^k(u)$, $f_i^k(u) = \min\limits_{j=1,\dots,J_i^k} f_{ij}^k(u)$ до кожної $(k-1)$-ї вершини $\Im^{\Upsilon}$. Кількість вершин k-рівня $\Im^{\Upsilon}$ дорівнює $\tau_k = \eta_1 \cdot \eta_2 \cdot \ldots \cdot \eta_k$. Кожній вершині $\Im_k^{\Upsilon}$ відповідають нерівності системи $\varphi(u) \ge 0, f_{i_1}^1(u) \ge 0, f_{i_2}^2(u) \ge 0, \dots, f_{i_k}^k(u) \ge 0$. Слід зауважити, що $\tau_n = \eta_1 \cdot \eta_2 \cdot \ldots \cdot \eta_{n-1} \cdot \eta_n = \eta$, де η — кількість вершин $\Im^{\Upsilon}$. Оскільки множина допустимих розв'язків W є об'єднанням підмножин W_s, $s = 1, 2, \dots, \eta$ (11), то кожній W_s відповідає s-та кінцева вершина $\Im^{\Upsilon}$. Отже, підмножини W_s визначаються за допомогою системи нерівностей виду $\varphi(u) \ge 0, f_{s_k}^k(u) \ge 0, \ k = 1, \dots, n$.

Розглянемо спосіб генерації непорожньої підмножини W_s для початкової точки $u^0 \in W$. Для генерації підмножини потрібно із системи $\Upsilon(u) \ge 0$ виділити підсистему $\Upsilon_s(u) \ge 0$, яка описує підмножину $W_s \subset W$ таку, що $u^0 \in W_s = \{u \in R^{\sigma} : \Upsilon_s(u) \ge 0\}$, де $\Upsilon_s(u)$ визначається за формулою (10).

Формування підмножини W_s здійснимо у такий спосіб. Спочатку виконуємо послідовне вичерпне перебирання вершин v_s^1, $s = 1, \dots, \eta_1$, першого рівня $\Im$ і пошук s_1 такого, що $f_{s_1}^1(u^0) = f^1(u^0) = \max\{f_1^1(u^0), f_2^1(u^0), \dots, f_{\eta_1}^1(u^0)\}$. Після цього здійснюємо вичерпний пошук нащадків v_s^2, $s = 1, \dots, \eta_2$, вершини $v_{s_1}^1$ і пошук s_2 такого, що $f_{s_2}^2(u^0) = f^2(u^0) = \max\{f_1^2(u^0), f_2^2(u^0), \dots, f_{\eta_1}^2(u^0)\}$, і так далі. На n-му рівні дерева розв'язків $\Im$ виконується вичерпне перебирання вершин v_s^n, $s = 1, \dots, \eta_n$, які є нащадками вершин $v_{s_{n-1}}^{n-1}$ і пошук s_n такого, що $f_{s_n}^n(u^0) = f^n(u^0) = \max\{f_1^n(u^0), f_2^n(u^0), \dots, f_{\eta_n}^n(u^0)\}$. Після цього формується система нерівностей, яка відповідає кінцевій вершині дерева розв'язків $\Im$, що описує множину $W_s = \{u \in R^{\sigma} : \varphi(u) \ge 0, f_{s_1}^1(u) \ge 0, f_{s_2}^2(u) \ge 0, \dots, f_{s_n}^n(u) \ge 0\}$, де кожній послідовності чисел $s_1, s_2, \dots, s_k, \dots, s_n$ відповідає число s.

ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ У СИСТЕМАХ ЗІ СПІЛЬНОЮ ПАМ'ЯТТЮ

Запропоновані в роботах [10–17] методи розв'язання задачі (1), (2) ґрунтуються на стратегії декомпозиції основної задачі до послідовності підзадач меншої вимірності зі значно меншою кількістю нерівностей. Для формування кожної з підзадач потрібно виконати генерацію множини допустимих розв'язків підзадач (13). Слід зауважити, що залежно від обраної величини параметра декомпозиції така послідовність підзадач (13) може бути досить великою, а витрати машинного часу на генерацію та перехід між підзадачами є також суттєвими. Це зумовлено тим, що для формування кожної підзадачі потрібно виконати підстановку отриманої початкової точки в систему нерівностей, яка описує основну задачу, та виділити з неї нову підсистему нерівностей за допомогою дерева розв'язків.

Оскільки система нерівностей (2), що описує множину допустимих розв'язків основної задачі, задається великою кількістю нелінійних нерівностей, то на формування з них нової підсистеми, яка буде задавати множину допустимих розв'язків задачі (1), (2), витрачається час $O(N^2)$. З метою мінімізації часових витрат для алгоритмів розв'язання задачі (1), (2) можна використовувати техно-

логії паралельних обчислень у системах зі спільною пам'яттю [25] та в системах з розподіленою пам'яттю (кластерні системи) [26]. Слід зазначити, що обсяг зусиль, витрачених на перероблення послідовного алгоритму у паралельний, значною мірою залежить від типу задачі.

Побудова паралельного алгоритму передбачає такі етапи: декомпозиція задачі (1), (2) на підзадачі, які можна буде виконувати паралельно; виявлення інформаційних залежностей між виділеними підзадачами; масштабування підзадач; балансування навантаження для кожного процесора.

Алгоритм для паралельних обчислень зводиться до розбиття масиву вихідних даних на фрагменти, оброблення яких здійснюється незалежно на різних процесорах. Здатність алгоритму до розпаралелювання потенційно пов'язана з однією з двох (або одночасно з обома) внутрішніх властивостей, які характеризують як паралелізм задач, так і паралелізм даних. Якщо алгоритм ґрунтується на паралелізмі задач, то задача розбивається на низку відносно самостійних підзадач, кожна з яких завантажується у «свій» процесор. Кожна підзадача реалізується незалежно, але використовує спільні дані або обмінюється результатами своєї роботи з іншими підзадачами. Для реалізації такого алгоритму у багатопроцесорній системі необхідно виявляти незалежні підзадачі, які можуть виконуватися паралельно.

Основний принцип розпаралелювання алгоритму пошуку локального екстремуму задачі виду (1), (2) ґрунтується на тому, що *phi*-функції складених об'єктів є максимінними функціями, що дає змогу представити множину допустимих розв'язків задачі (1), (2) у вигляді об'єднання підмножин, кожна з яких описується системою диференційовних функцій. Алгоритм формування підсистеми, який задає множину допустимих розв'язків підзадачі локальної оптимізації, можна представити у вигляді графа у ярусно-паралельній формі. Для ярусно-паралельної форми алгоритму важливим є те, що операції, яким відповідають вершини одного ярусу, не залежать одна від одної, і тому можливою є реалізація алгоритму, в якій вони можуть бути виконані паралельно.

На рис. 1. наведено ярусно-паралельну форму алгоритму формування підзадачі локальної оптимізації задачі оптимальної упаковки складених об'єктів, на кожному з чотирьох ярусів якого зазначені процедури можуть бути виконані паралельно.

На першому ярусі знаходяться процедури обчислення *phi*-функцій (4) для кожної пари складених об'єктів T_i та T_j, $i, j \in I_N$. Для виконання кожної процедури першого ярусу на другому ярусі виконується процедура обчислення *phi*-функцій (8) для кожної пари базових об'єктів. На третьому ярусі для кожної

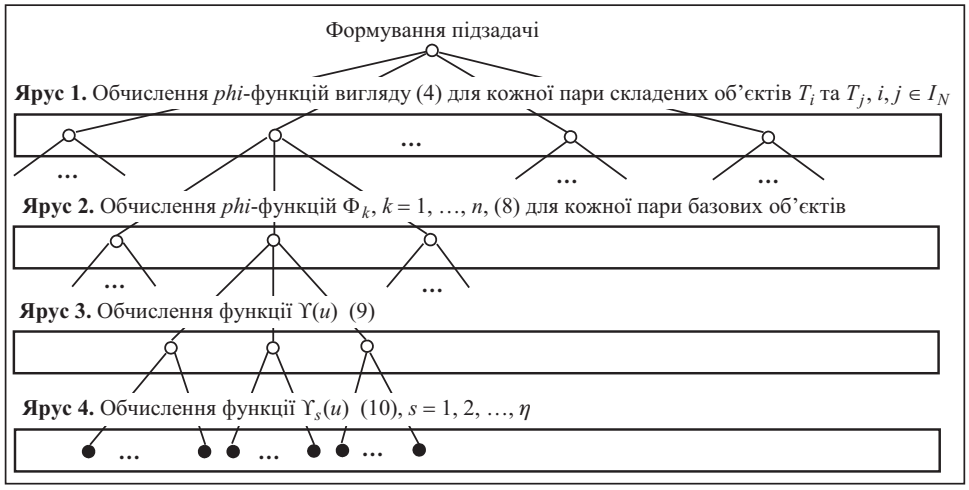

Рис. 1. Ярусно-паралельна форма алгоритму локальної оптимізації

процедури другого ярусу виконуються процедури обчислення функції розміщення (9). На четвертому ярусі виконуються процедури обчислення функцій (10), що визначають значення функції розміщення третього ярусу.

Задачі перших трьох ярусів підтримують паралелізм даних, оскільки для їхнього виконання використовуються однакові обчислювальні процедури, але над різними масивами даних. Задачі четвертого ярусу підтримують паралелізм задач, оскільки задачі цього ярусу реалізуються окремими процедурами над однаковими масивами даних.

Для реалізації паралельних обчислень алгоритму ярусно-паралельної форми необхідно забезпечити синхронізацію виконання паралельних задач другого та третього ярусів, оскільки результати процедур цих ярусів залежать від результатів, отриманих на третьому та четвертому ярусах відповідно.

Отже, для реалізації представленої схеми паралельних обчислень потрібно застосувати сучасні засоби, які забезпечать програму механізмами автоматичної генерації паралельних обчислювальних потоків, масштабування задач, синхронізації потоків.

Як засіб паралельних обчислень обрано бібліотеку розпаралелювання задач TPL (Task Parallel Library) для платформи NET. За рахунок цього можна значно вдосконалити багатопотокове програмування у два базових способи. По-перше, вона спрощує створення і застосування багатьох потоків, по-друге, дає змогу автоматично використовувати кілька процесорів. З огляду на ці дві особливості бібліотека TPL рекомендується до застосування для організації багатопотокового оброблення, адже завдяки їм полегшується використання системних ресурсів. Застосовуючи TPL, паралелізм у програму можна ввести двома основними способами: за допомогою паралелізму даних та паралелізму задач. Бібліотека TPL надає можливість автоматично масштабувати виконання коду на кілька процесорів та автоматично розподіляти навантаження прикладних програм (applications) між доступними процесорами в динамічному режимі, використовуючи пул потоків CLR, а також займається розподілом роботи, плануванням потоків, управлінням станом та іншими низькорівневими деталями.

Результати обчислювальних експериментів показали, що технології, які використовують паралельні обчислення на системах зі спільною пам'яттю, забезпечили скорочення витрат часу на розв'язання задач у середньому до 50 %. На рис. 2 наведено діаграму, яка відображає витрати часу на пошук розв'язку задачі пакування неопуклих неорієнтованих багатогранників у кубоїд мінімального об'єму із застосуванням та без застосування технологій паралельних обчислень [27].

Як видно з діаграми, ефективність застосування технологій паралельних обчислень стає більшою зі зростанням кількості об'єктів. Це пов'язано з тим, що для малої кількості об'єктів наявність витрат на обслуговування паралельних обчислень зумовлює зниження ефективності їхнього застосування.

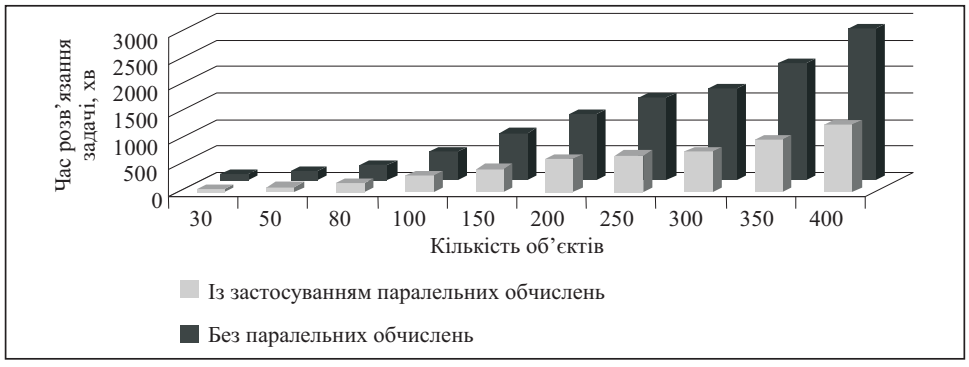

Рис. 2. Витрати часу у системах зі спільною пам'яттю

Для розв'язання задачі (1), (2) зазвичай використовують стратегію мультистарту (multistart) [28–30], тому для отримання кращих наближень до глобальних розв'язків можуть бути застосовані технології паралельних обчислень у системах з розподіленою пам'яттю (кластерні системи). Зокрема, паралельний алгоритм на основі мультистарту розроблено для задачі рівноважного пакування m кругів із заданими радіусами r_i та вагами w_i, $i=1,\dots,m$, у круг S мінімального радіуса [30]. Задачі рівноважного пакування сімейства кругів S_i, $i=1,\dots,m$, відповідає багатоекстремальна задача нелінійного програмування

$$r^* = \min_{R,x,y} r \tag{14}$$

для обмежень

$$x_i^2 + y_i^2 \le (r - r_i)^2,\ i=1,\dots,m, \tag{15}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 \ge (r_i + r_j)^2,\ 1 \le i < j \le m, \tag{16}$$

$$\sum_{i=1}^{m} \lambda_i x_i = 0,\ \sum_{i=1}^{m} \lambda_i y_i = 0, \tag{17}$$

$$r \ge r_{\text{low}}, \tag{18}$$

де $x=(x_1,\dots,x_m)$, $y=(y_1,\dots,y_m)$, (x_i, y_i) — невідомий центр круга S_i, r — невідомий радіус круга S, $\lambda_i = w_i / \sum_{j=1}^{m} w_j$, $i=1,\dots,m$, $r_{\text{low}} = \max_{i=1,\dots,m} r_i$ — нижня межа шуканого радіуса. Лінійні обмеження (17) означають, що центр тяжіння сімейства кругів S_i, $i=1,\dots,m$, локалізовано у початку координат.

Паралельний алгоритм запускає множинний пошук локальних екстремумів задачі (14)–(18) за допомогою r-алгоритму Шора [31–33]. Реалізація паралельного алгоритму використовує процедуру «Master-Slave» на $(k+1)$ процесорах. Один з них обирається «провідним» (Master) процесором, а решта k — «веденими» (Slave).

На початку обчислень у Master-процесорі випадковим чином генеруються k стартових точок у крузі радіуса $R = \sum_{i=1}^{m} r_i$, які пересилаються у Slave-процесори. Slave-процесор займається пошуком локального мінімуму задачі (14)–(18) для своєї стартової точки. Як тільки r-алгоритм Шора закінчує роботу на якомусь зі Slave-процесорів, результат пошуку передається в Master-процесор. Якщо при цьому знайдено локальний мінімум задачі (14)–(18), то значення радіуса зовнішнього круга порівнюється з найкращим із знайдених до цього моменту значенням r. Якщо радіус менше r, то він стає новим значенням і запам'ятовуються відповідні йому координати центрів кругів (x, y). Потім Master-процесор генерує нову стартову точку, яка передається для чергового пошуку локального мінімуму в той Slave-процесор, для якого r-алгоритм закінчив роботу. Процес завершується, якщо перевищено задану кількість стартових точок або замовлений час.

Програмну реалізацію паралельного алгоритму [34, 35] виконано мовою програмування C++ у середовищі паралельного програмування MPI (Message Passing Interface). Як генератор псевдовипадкових чисел програма використовує стандартну функцію rand із бібліотеки C++. Пошук локальних мінімумів здійснюється програмним модулем ralgb5 [33, с. 384, 385]. Паралельна програма призначена для роботи на кластері із середовищем MPI під керуванням операційної системи Linux. Вона може працювати як на одному процесорі, так і на багатьох паралельних процесорах.

На кластері СКІТ-3 Інституту кібернетики імені В.М. Глушкова проведено дослідження прискорення та ефективності паралельного алгоритму для за-

Рис. 3. Прискорення паралельного алгоритму P_k для розв'язання задачі на СКІТ-3

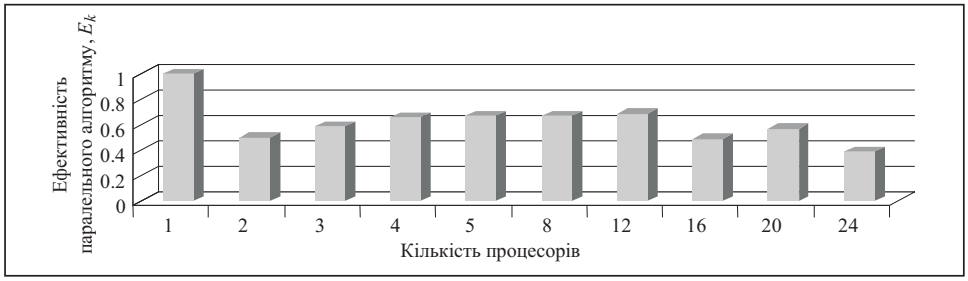

Рис. 4. Ефективність паралельного алгоритму E_k для розв'язання задачі на СКІТ-3

дачі (14)–(18) для $m = 40$. Кількість точок, що генеруються для методу мультистарту, дорівнювала 50, а кількість процесорів кластера варіювалася від 1 до 24. На рис. 3 та 4 представлено діаграми прискорення паралельного алгоритму P_k (рис. 3) та ефективності паралельного алгоритму E_k (рис. 4).

Паралельне прискорення обчислюється за формулою $P_k = t_0 / t_k$, де t_0 — час роботи послідовного алгоритму, а t_k — час розв'язання задачі паралельним алгоритмом з використанням k процесорів, $k = 1, \ldots, n$. Ефективність паралельного алгоритму обчислюється за формулою $E_k = S_k / k$. З діаграми випливає, що для чотирьох процесорів час розв'язання задачі зменшується в 2.65 рази, для восьми — у 5.38 рази, для шістнадцяти — у 7.87 рази, а для двадцяти чотирьох — у 9.37 рази.

На рис. 5 наведено діаграму, яка відображає залежність часу розв'язання задачі від кількості використовуваних процесорів. З неї легко побачити, що, починаючи з 12 процесорів, подальше додавання процесорів не чинить істотного впливу на час розв'язання задачі. Це пов'язано зі зростанням втрат на обміни інформацією при збільшенні ступеня паралелізму. Загалом прискорення та ефективність паралельного алгоритму відповідають стандартним характеристикам для процедури «Master-Slave», коли час розв'язання кожної окремої підзадачі на Slave-процесорі є порівняно невеликим.

Паралельна програма дала змогу поліпшити знайдене у роботі [30] значення радіуса зовнішнього кола $r = 714.563$ для рівноважної упаковки 40 кругів. Два близьких значення $r^* = 711.9252$ та $r^{**} = 711.9522$ (зменшені більше ніж на дві

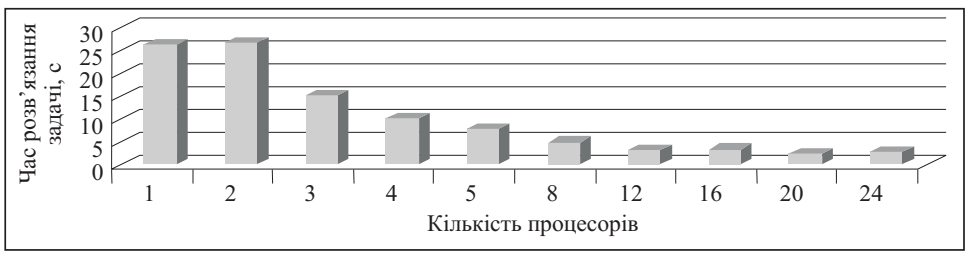

Рис. 5. Витрати часу на розв'язання задачі на СКІТ-3

Таблиця 1. Координати центрів розташовуваних кругів для їхнього рівноважного пакування: $m = 40$, $r^* = 711.9252$, $r^{**} = 711.9522$

i	r_i	w_i	x_i^*	y_i^*	x_i^{**}	y_i^{**}
1	106	11	– 598.7338	93.0767	50.7008	– 211.0646
2	112	12	– 8.6709	– 223.7210	417.2515	– 103.7310
3	98	9	– 47.9483	– 430.0152	– 605.3276	102.5470
4	105	11	– 424.4947	– 428.1792	130.1298	176.7494
5	93	8	– 139.2399	187.0966	– 394.1751	477.2084
6	103	10	339.3850	– 495.2939	– 58.3472	606.1505
7	82	6	– 412.7108	146.7259	205.7851	– 104.7966
8	93	8	136.4842	– 380.3600	283.7769	51.8631
9	117	13	592.8649	49.4694	– 354.9349	– 164.0976
10	81	6	– 186.7412	– 149.2900	461.7161	84.0772
11	89	7	– 423.4067	456.9055	592.3418	192.8748
12	92	8	– 216.7105	– 342.7265	– 146.5073	– 193.3742
13	109	11	– 367.0216	– 209.2815	359.4074	484.1257
14	104	10	– 302.8163	306.2155	355.3129	235.4158
15	115	13	563.4406	– 197.1153	– 586.6447	– 110.4535
16	110	12	390.4387	– 53.2565	– 84.1099	194.8138
17	114	12	243.8727	326.4675	156.6935	577.0563
18	89	7	– 445.4699	– 27.4853	– 434.8079	25.7874
19	82	6	466.6679	423.1156	– 396.0844	– 465.1835
20	120	14	320.5037	– 273.0947	– 510.8747	299.0226
21	108	11	146.7110	– 581.3039	– 315.0662	182.2192
22	86	7	493.1765	– 385.4340	– 244.9335	576.0415
23	93	8	– 609.8479	– 105.6127	– 11.9597	– 600.0933
24	100	10	552.6614	262.7126	–251.3826	– 354.7961
25	102	10	379.6380	158.4682	284.1286	– 271.2848
26	106	11	– 530.0235	293.6332	– 514.8889	– 319.4801
27	111	12	– 102.4866	387.7584	– 212.4027	– 562.1643
28	107	11	– 256.6874	25.2136	14.7713	– 1.1169
29	109	11	– 38.9945	601.6629	9.5224	405.3080
30	91	8	– 48.6834	– 619.0138	510.6637	353.2765
31	111	12	173.4128	– 94.9789	508.6244	– 320.0698
32	91	8	– 553.7740	– 280.8603	203.9259	358.3262
33	101	10	48.0307	237.7494	– 58.6326	– 411.7913
34	91	8	192.4937	111.2803	172.0081	– 596.6531
35	108	11	– 249.7244	549.8758	– 200.2283	– 0.7241
36	114	12	– 240.7383	– 547.3204	– 211.8952	378.7892
37	118	13	– 32.5944	5.0314	346.8533	– 482.1535
38	85	7	154.9182	607.4830	608.4048	– 151.3693
39	87	7	85.4279	450.1455	624.6365	19.8631
40	98	9	318.2680	524.9854	140.3620	– 410.3213

одиниці) отримано у випадку використання 24 процесорів СКІТ-3 і 5000000 стартових точок [34]. Координати центрів розташовуваних кругів наведено в табл. 1 (стовпці 4 і 5 відповідають $r^* = 711.9252$, а стовпці 6 і 7 — $r^{**} = 711.9522$). Час виконання програми в першому випадку становив 104762 с (приблизно 29 год 10 хв), а в другому — 120910 с (приблизно 33 год 30 хв).

ВИСНОВКИ

У роботі застосовано технології паралельних обчислень для розв'язання задач оптимізаційного геометричного проєктування, що дало змогу в декілька разів зменшити витрати часу під час пошуку локально оптимальних розміщень 2D та 3D об'єктів із застосуванням сучасного програмного забезпечення для розв'язання задач нелінійного програмування та отримати кращі результати за значенням цільової функції.

Основні принципи розпаралелювання алгоритму пошуку локального екстремуму задачі виду (1), (2) ґрунтуються на використанні технології паралельних обчислень у системах зі спільною пам'яттю та в системах з розподіленою пам'яттю (кластерні системи). Перша технологія ґрунтується на тому, що *phi*-функції складених об'єктів є максимінними функціями. Це дає змогу представити множину допустимих розв'язків оптимізаційної задачі геометричного проєктування у вигляді об'єднання підмножин, кожна з яких описується системою нерівностей з гладкими функціями, та повною мірою використати всі можливості бібліотеки розпаралелювання задач TPL для платформи .NET під час реалізації розробленого алгоритму. Друга технологія використовує стратегію мультистарту та методи мінімізації негладких функцій і дає змогу застосувати новітнє програмне забезпечення.

СПИСОК ЛІТЕРАТУРИ

1. Liu X., Liu J., Cao A. HAPE3D — a new constructive algorithm for the 3D irregular packing problem. *Frontiers Inf. Technol. Electronic Eng*. 2015. Vol. 16, N 5. P. 380–390.
2. Guangqiang L., Fengqiang Z., Rubo Z. Du Jialu Du., Chen G., Yiran Z. A parallel particle bee colony algorithm approach to layout optimization. *Journal of Computational and Theoretical Nanoscience*. 2016. Vol. 13, N 7. P. 4151–4157.
3. Karabulut K., Ínceoğlu M.A. Hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method. In: *Advances in Information Systems*. *ADVIS 2004. Lecture Notes in Computer Science.* Yakhno T. (Ed.). 2004. Vol. 3261. P. 441–450.
4. Litvinchev I., Infante L., Ozuna L. Approximate packing: integer programming models, valid inequalities and nesting. In: *Optimized Packings and Their Applications (Ser. Springer Optimization and Its Applications)*. Fasano G., Pinter J.D. (Eds.). 2015. Vol. 105. P. 187–205.
5. Litvinchev I., Infante L., Ozuna L. Packing circular-like objects in a rectangular container. *Journal of Computer and Systems Sciences International.* 2015. Vol. 54, N 2. P. 259–267.
6. Burke E.K., Hellier R.S. R., Kendall G., Whitwell G. Irregular packing using the line and arc no-fit polygon. *Operations Research.* 2010. Vol. 58, N 4. P. 948–970.
7. Cherri L.H., Mundim L.R., Andretta M., Toledo F.M., Oliveira J. F., Carravilla M.A. Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*. 2016. Vol. 253. P. 570–583.
8. Stoyan Yu., Romanova T. Mathematical models of placement optimization: two- and three-dimensional problems and applications. In: *Modeling and Optimization in Space Engineering. (Ser. Springer Optimization and Its Applications)*. Fasano G., Pinter J.D. (Eds.). New York: Springer, 2012. Vol. 73. 404 p.
9. Stoyan Yu., Yakovlev S. Configuration space of geometric objects. *Cybernetics and Systems Analysis.* 2018. Vol. 54, N 5. P. 716–726.
10. Stoyan Yu., Pankratov A., Romanova T. Placement problems for irregular objects: mathematical modeling, optimization and applications. In: *Optimization Methods and Applications. Modeling and Optimization in Space Engineering (Ser. Springer Optimization and Its Applications)*. Butenko S., Pardalos P., Shylo V. (Eds.). New York: Springer, 2017. Vol. 130. P. 521–559.
11. Stoyan Yu., Chugay A. Mathematical modeling of the interaction of non-oriented convex polytopes. *Cybernetics and Systems Analysis*. 2012. Vol. 48, N 6. P. 837–845.

12. Stoyan Yu., Pankratov A., Romanova T. Cutting and packing problems for irregular objects with continuous rotations: mathematical modeling and nonlinear optimization. *Journal of the Operational Research Society.* 2016. Vol. 67, Iss. 5. P. 786–800.

13. Stoyan Yu., Pankratov A., Romanova T., Chugay A. Optimized object packings using quasi-*phi*-functions. In: *Optimized Packings and Their Applications (Ser. Springer Optimization and Its Applications)*. Fasano G., Pinter J.D. (Eds.). New York: Springer, 2015. Vol. 105. P. 265–291.

14. Stoyan Y.G., Chugay A.M. Packing different cuboids with rotations and spheres into a cuboid. *Advances in Decision Sciences*. 2014. URL:: https://www.hindawi.com/journals/ads/2014/571743.

15. Stoyan Y.G., Semkin V.V., Chugay A.M. Modeling close packing of 3D objects. *Cybernetics and Systems Analysis.* 2016. Vol. 52, N 2. P. 296–304.

16. Stoian Y.E., Chugay A.M., Pankratov A.V., Romanova T.E. Two approaches to modeling and solving the packing problem for convex polytopes. *Cybernetics and Systems Analysis*. 2018. Vol. 54, N 4. P. 585–593.

17. Romanova T., Bennell J., Stoyan Yu., Pankratov A. Packing of concave polyhedra with continuous rotations using nonlinear optimization. *European Journal of Operational Research*. 2018. Vol. 268, Iss. 1. P 37–53.

18. Pankratov A., Romanova T., Litvinchev I. Packing ellipses in an optimized convex polygon. *Journal of Global Optimization*. 2019. https://doi.org/10.1007/s10898-019-00777-y.

19. Pankratov A., Romanova T., Litvinchev I. Packing ellipses in an optimized rectangular container. *Wireless Networks.* 2018. https://doi.org/10.1007/s11276-018-1890-1.

20. Romanova T., Pankratov A., Litvinchev I., Pankratova Yu., Urniaieva I. Optimized packing clusters of objects in a rectangular container. *Mathematical Problems in Engineering*. Vol. 2019. Article ID 4136430. 12 p. https://doi.org/10.1155/2019/4136430.

21. Wang Y., Lin C.L., Miller J.D. 3D image segmentation for analysis of multisize particles in a packed particle bed. *Powder Technology.* 2016. Vol. 301. P. 160–168.

22. Li S.X., Zhao J., Lu P., Xie Y. Maximum packing densities of basic 3D objects. *Chinese Science Bulletin*. 2010. Vol. 55, Iss. 2. P.114–119.

23. Ramya A., Vanapalli S. 3D printing technologies in various applications. *International Journal of Mechanical Engineering and Technology*. 2016. Vol. 7, N 3. P. 396–409.

24. Baumers M., Dickens P., Tuck C., Hague R. The cost of additive manufacturing: machine productivity, economies of scale and technology-push. *Technological Forecasting & Social Change*. 2016. Vol. 102, Iss. C. P. 193–201.

25. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Санкт-Петербург: БХВ-Петербург, 2002. 608 с.

26. Кластерный комплекс Института кибернетики. Кластерный комплекс СКИТ. URL: https://icybcluster.org.ua/.

27. Chugay A., Stoian Ye. Cluster packing of concave non-oriented polyhedra in a cuboid. *Сучасні інформаційні системи.* 2018. Т. 2, № 1. С. 16–21.

28. Коваленко А.А., Романова Т.Е., Стецюк П.И. Задача балансной компоновки 3D-объектов: математическая модель и методы решения. *Кибернетика и системный анализ.* 2015. Т. 51, № 4. С. 71–81.

29. Stoyan Yu., Romanova T., Pankratov A., Kovalenko A., Stetsyuk P. Balance layout problems: Mathematical modeling and nonlinear optimization. In: *Space Engineering. Modeling and Optimization with Case Studies (Ser. Springer Optimization and its Applications)*. Fasano G., Pintér J. (Eds). New York: Springer, 2016. Vol. 114. P. 369–400.

30. Stetsyuk P., Romanova T., Scheithauer G. On the global minimum in a balanced circular packing problem. *Optimization Letters*. 2016. Vol. 10, Iss. 6. P. 1347–1360.

31. Stetsyuk P.I. Shor's r-algorithms: theory and practice. In: *Optimization Methods and Applications: In Honor of the 80th Birthday of Ivan V. Sergienko.* Butenko S., Pardalos P.M, Shylo V (Eds.). New York: Springer, 2017. P. 495–520.

32. Стецюк П.И. Теория и программные реализации *r*-алгоритмов Шора. *Кибернетика и системный анализ*. 2017. Т. 53, № 5. С. 43–57.

33. Стецюк П.И. Методы эллипсоидов и *r*-алгоритмы. Кишинэу: Эврика, 2014. 488 с.

34. Стецюк П.І., Лиховид О.П. Комп'ютерна програма «A parallel algorithm for a balanced circular packing problem». Свідоцтво про реєстрацію авторського права на твір № 62184. Україна. Міністерство освіти і науки. Державний департамент інтелектуальної власності. Дата реєстрації 20.10.2015.

35. Лиховид А.П. О реализации параллельного алгоритма для решения задач равновесной упаковки. *Теорія оптимальних рішень.* Київ: Ін-т кібернетики ім. В.М. Глушкова НАН України, 2015. С. 154–159.

Надійшла до редакції 25.03.2019

Т.Е. Романова, П.И. Стецюк, А.М. Чугай, С.Б. Шеховцов
ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ РЕШЕНИЯ ОПТИМИЗАЦИОННЫХ ЗАДАЧ ГЕОМЕТРИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Аннотация. В работе описано применение технологий параллельных вычислений на системах с общей памятью и распределенной памятью для решения оптимизационных задач геометрического проектирования. Первая технология базируется на максиминных свойствах *phi*-функций для составных объектов, а вторая технология использует стратегию мультистарта и методы минимизации негладких функций. Это позволило в несколько раз уменьшить затраты времени при поиске локально оптимальных размещений 2D и 3D объектов и получить лучшие результаты по значению целевой функции.

Ключевые слова: размещение геометрических объектов, *phi*-функция, нелинейное программирование, параллельные вычисления.

T.E. Romanova, P.I. Stetsyuk, A.M. Chugay, S.B. Shekhovtsov
PARALLEL COMPUTING TECHNOLOGIES FOR SOLVING OPTIMIZATION PROBLEMS OF GEOMETRIC DESIGN

Abstract. The paper describes the use of parallel computing technologies on systems with shared memory and distributed memory for solving optimization geometric design problems. The first technology is based on the maxmin properties of *phi*-functions for composed objects, and the second technology uses the multi-start strategy and methods for minimizing non-smooth functions. This made it possible to reduce several times the computational time spent in searching for locally optimal placements of 2D and 3D objects and to get the best results in terms of the objective function value.

Keywords: placement of geometric objects, *phi*-function, nonlinear programming, parallel computing.

Романова Тетяна Євгеніївна,
доктор техн. наук, професор, провідний науковий співробітник Інституту проблем машинобудування ім. А.М. Підгорного НАН України, Харків, e-mail: tarom27@yahoo.com.

Стецюк Петро Іванович,
доктор фіз.-мат. наук, завідувач відділу Інституту кібернетики ім. В.М. Глушкова НАН України, Київ, e-mail: stetsyukp@gmail.com.

Чугай Андрій Михайлович,
доктор техн. наук, старший науковий співробітник, старший науковий співробітник Інституту проблем машинобудування ім. А.М. Підгорного НАН України, Харків, e-mail: chugay@ipmach.kharkov.ua.

Шеховцов Сергій Борисович,
кандидат техн. наук, доцент, доцент кафедри Харківського національного університету внутрішніх справ, e-mail: ep109@ukr.net.