Кібернетика та комп'ютерні технології

УДК 519.85 DOI:10.34229/2707-451X.24.4.1

Б.О. ЗАДОРОЖНИЙ, Т.Є. РОМАНОВА, П.І. СТЕЦЮК,
С.Р. ТИВОДАР, С.Б. ШЕХОВЦОВ

ВИКОРИСТАННЯ R-АЛГОРИТМУ ДЛЯ ПАКУВАННЯ НЕРІВНИХ КРУГІВ У КРУГ МІНІМАЛЬНОГО РАДІУСУ

Досліджено два способи використання r-алгоритму Шора для задачі пакування нерівних кругів у зовнішній круг мінімального радіусу. Перший спосіб реалізує багаторазовий запуск r-алгоритму з дихотомією кроку із допустимої стартової точки, в якій досягається знайдений евристичним алгоритмом найменший радіус зовнішнього круга. Розглядаються дві версії алгоритму. Для першої версії величина кроку зменшується вдвічі по ходу ітераційного процесу, а для другої – додається опція, яка дозволяє відновлювати максимальне значення величини кроку r-алгоритму. Алгоритм реалізовано мовою програмування Rust з використанням бібліотеки nalgebra. Другий спосіб має на меті дослідження щодо підвищення точності за значенням радіуса зовнішнього круга локальних мінімумів багатоекстремальної негладкої функції та зменшення числа ітерацій r-алгоритму за рахунок підвищення розрядності чисел (128 та 256 біт). Відповідний алгоритм реалізований мовою програмування Julia.

Ключові слова: *пакування кругів, r-алгоритм, евристичний алгоритм, Rust, Julia.*

© Б.О. Задорожний, Т.Є. Романова, П.І. Стецюк, С.Р. Тиводар, С.Б. Шеховцов, 2024

Вступ. Статті [1, 2] присвячені дослідженню одного з евристичних алгоритмів для пакування нерівних кругів у круг (у подальшому, зовнішній круг) мінімального радіусу та розробці його модифікацій за допомогою *r*-алгоритму Шора [3, 4] з регулюванням величини кроку методом дихотомії. У роботі [5] показано, що розв'язання квадратичної багатоекстремальної задачі пакування кругів солвером BARON [6, 7] вимагає великих затрат у часі навіть для малої кількості кругів – від п'яти до десяти.

У цій статті опишемо досвід використання *r*-алгоритму для знаходження та покращення локальних мінімумів у задачах пакування десятків нерівних кругів у зовнішній круг мінімального радіусу. Матеріал статті викладено в такому порядку.

У розділі 1 наведено постановку задачі та описано евристичний алгоритм, результат роботи якого будемо покращувати. У розділі 2 описано *r*-алгоритм Шора з адаптивним регулюванням кроку. Розділ 3 присвячено опису двох стратегій покращення евристичного розв'язку за допомогою *r*-алгоритму. У розділі 4 наведено результати застосування обох стратегій для 50 конкурсних задач, проведено їх аналіз за кількістю отриманих балів та середньою кількістю ітерацій *r*-алгоритму. У розділі 5 наведено опис результатів використання *r*-алгоритму для чисел великої розрядності (128 та 256 біт), що дозволяє знаходити мінімальний радіус зовнішнього круга з точністю до 10–20 цифр після коми. Показано, що з підвищенням розрядності чисел кількість ітерацій *r*-алгоритму зменшується, а точність знаходження радіусу зростає.

1. Постановка задачі та евристичний алгоритм

Задано набір m кругів з радіусами r_i, $i=1,...,m$. Потрібно знайти центри (x_i, y_i), $i=1,...,m$ цих кругів і радіус R зовнішнього круга з центром $(0,0)$ такі, що:

1) кожен круг $i=1,...,m$ має повністю знаходитись всередині зовнішнього круга (круги можуть торкатися межі зовнішнього круга);

2) кожен круг $i = 1,...,m$ має повністю знаходитись всередині зовнішнього круга (круги можуть торкатися межі зовнішнього круга);

3) для будь-якої пари (i, j), де $i, j \in \{1,...,m\}$ та $i < j$, круги i, j не перетинаються (їм дозволено торкатися один одного);

4) радіус R зовнішнього круга має бути якомога меншим.

Це формулювання було використано для міжнародного конкурсу «Щільна упаковка кругів в круг мінімального радіусу» [8].

У конкурсі змагалися близько 190 учасників з 10 країн світу з метою знайти найкращі за значенням радіусу зовнішнього круга розташування кругів для 50 конкурсних задач (по 10 тестових задач для m =10, 20, 30, 40, 50) в умовах обмеженого часу – відводилося до 2-х секунд на кожну задачу. Друге місце в змаганні зайняв перший автор цієї статті, чий евристичний алгоритм отримав 4904 балів з 5000 можливих.

Наведемо опис евристичного алгоритму для розв'язання задачі пакування кругів у круг мінімального радіусу згідно статті [2].

Етап 1. Встановлюємо допустимі межі радіусу зовнішнього круга: ліва межа дорівнює $R_{low} = \max\limits_{i=1,..,m} r_i$, а права – $R_{up} = \sum_{i=1}^{m} r_i$.

Етап 2. Визначаємо стартове значення радіуса зовнішнього круга як $R^0 = 0.5(R_{low} + R_{up})$.

Етап 3. Пакуємо круги за такими правилами:

1. Встановлюємо центр першого круга з набору $i = 1,2,\ldots,m$ в точку $(0, R^0 - r_1)$.

2. Генеруємо шар кругів. За годинниковою стрілкою розміщуємо кожний наступний круг так, щоб він торкався попереднього круга та межі зовнішнього круга. Використовуючи координати центра попереднього круга, обчислюємо координати центра наступного круга: встановлюємо центр наступного i-го круга в точці $(0, R^0 - r_i)$; розраховуємо кут, на який радіус-вектор потрібно повернути за годинниковою стрілкою. Отриманий кут уточнюємо за допомогою бінарного пошуку (рис. 1,*а*).

3. Розміщуємо наступні круги між межею зовнішнього круга та першим шаром кругів (рис. 1,*б*).

4. Розміщуємо круги між кругами попереднього шару або між кругами через один (рис. 1,*в*).

Етап 4. Якщо розміщення кругів та радіус зовнішнього круга задовольняють вимогам 1) та 2), то встановлюємо $R_{up} = R^0$ та запам'ятовуємо центри розміщених кругів. Інакше, встановлюємо $R_{low} = R^0$.

Етап 5. Якщо $R_{up} - R_{low} > \Delta$, то переходимо до етапу 2, де Δ – точність знаходження розв'язку.

Етап 6. Порівнюємо розв'язок отриманий на етапі 4 з найкращим знайденим. Якщо результат розміщення кругів задовольняє вимогам 1) та 2) та має менший радіус зовнішнього круга за найкращий розв'язок, то останній встановлюємо рівним знайденому. Якщо досягнена максимальна кількість ітерацій, то алгоритм закінчує роботу. Інакше застосуємо "swap"-стратегію для розміщення пари випадкових розміщених кругів та переходимо до етапу 1.

Алгоритм не завжди може гарантувати розміщення кругів, задовольнивши обмеження 1) і 2), тому потрібно обрати кількість ітерацій при якій вдасться досягнути хоча б один допустимий розв'язок. Варто зауважити, що оскільки алгоритм базується на певному наборі правил розміщення кругів, то при малій кількості ітерацій залишається "вільний простір", який можна зменшити, якщо пакувати круги щільніше.

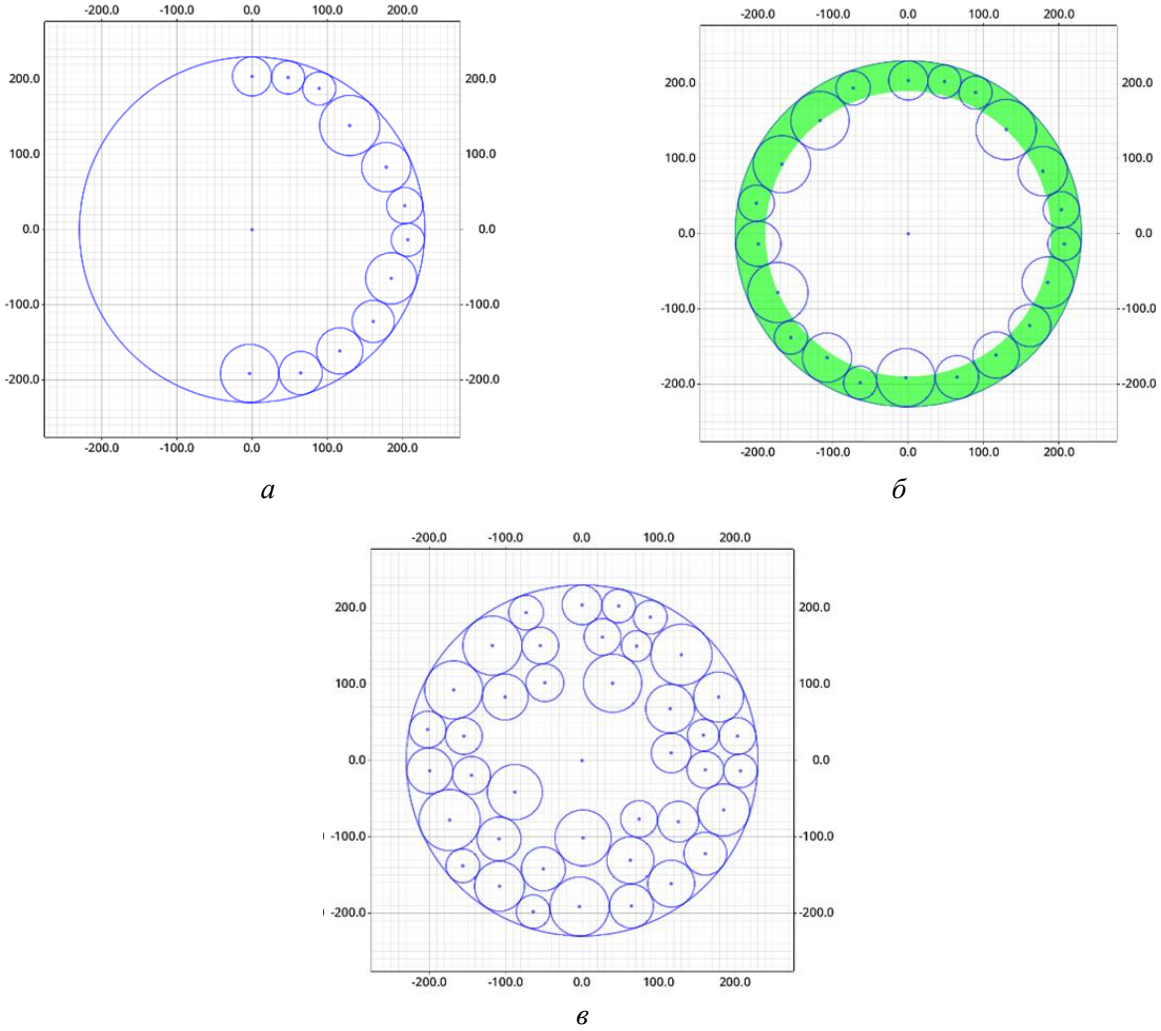

РИС. 1. Етап 3 евристичного алгоритму

У більшості конкурсних задач евристичний алгоритм знаходив такі розміщення кругів, які містили «вільний простір». Тому, отримані евристичним алгоритмом значення радіусу зовнішнього круга можна покращити, якщо застосувати інший алгоритм, який стартує зі знайденого евристичним алгоритмом розміщення кругів та зменшує, наскільки це можливо, радіус зовнішнього круга. Далі розглянемо використання *r*-алгоритму для покрашення знайденого евристичним алгоритмом розміщення кругів.

2. *r*-Алгоритм Шора з адаптивним регулюванням кроку

Нехай $f(z)$ – опукла функція, $z \in \mathbb{R}^n$. Мінімальне значення функції позначимо $f^* = f(z^*)$, де $z^* \in \mathbb{R}^n$. Нехай $\alpha > 1$ – коефіцієнт розтягу простору.

$r(\alpha)$-алгоритм призначений для мінімізації функції $f(z)$ є ітеративною процедурою знаходження послідовності n-вимірних векторів $\{z_k\}_{k=0}^{\infty}$ і послідовності $n \times n$-матриць $\{B_k\}_{k=0}^{\infty}$ за таким правилом:

$$z_{k+1} = z_k - h_k B_k \xi_k, \quad B_{k+1} = B_k R_{\beta_k}(\eta_k), \quad k = 0,1,2,..., \tag{1}$$

$$\xi_k = \frac{B_k^T g_f(z_k)}{\left\| B_k^T g_f(z_k) \right\|}, \quad h_k \geq h_k^* = \arg\min_{h \geq 0} f(z_k - h B_k \xi_k), \tag{2}$$

$$\beta_k = \frac{1}{\alpha_k} < 1, \quad \eta_k = \frac{B_k^T r_k}{\left\| B_k^T r_k \right\|}, \text{ де } r_k = g_f(z_{k+1}) - g_f(z_k), \tag{3}$$

де z_0 – початкова точка; $B_0 = I_n$ – одинична $n \times n$-матриця; h_k^* – величина кроку за умови мінімуму функції $f(z)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних; $R_\beta(\eta) = I_n + (\beta - 1)\eta\eta^T$ – оператор стиснення простору субградієнтів у нормованому напрямку η з коефіцієнтом $\beta = \frac{1}{\alpha} < 1$; $g_f(z_k)$ і $g_f(z_{k+1})$ – субградієнти функції $f(z)$ у точках z_k і z_{k+1}.

Якщо на ітерації k для цього процесу (1) – (3) виконуються певні критерії (умови) зупинки, то $k^* = k$, $z_k^* = z_k$ і робота алгоритму припиняється.

Сімейство $r(\alpha)$-алгоритмів визначається коефіцієнтом розтягу простору $\alpha > 1$ і послідовністю величин кроків $\{h_k\}_{k=0}^{\infty}$, які визначають такі два послідовні субградієнти $g_f(z_k)$ і $g_f(z_{k+1})$, розтягування за різницею яких зменшує ступінь витягнутості функції у перетвореному просторі змінних. Вибір коефіцієнта $\alpha > 1$ та адаптація величин $\{h_k\}_{k=0}^{\infty}$ до субградієнтного спуску та критеріїв зупинки визначають ті чи інші варіанти r-алгоритмів. Якщо $h_k = h_k^*$, то це буде відповідати точному пошуку мінімуму функції $\varphi(y) = f(B_k y) = f(z)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних $y = B_k^{-1} z$; якщо $h_k \approx h_k^*$, то – наближеному пошуку мінімуму функції $\varphi(y) = f(z)$.

Адаптивний спосіб регулювання кроку в напрямі антисубградієнта в перетвореному просторі змінних полягає у тому, що величина h_k налаштовується (адаптується) у процесі виконання одновимірного спуску, який завершується, як тільки знайдено субградієнт, що утворює негострий кут з субградієнтом, що визначає напрямок одновимірного спуску (умова завершення спуску за напрямом). Налаштування величини h_k здійснюється за допомогою чотирьох параметрів: $h_0 > 0$ – величина початкового кроку (використовується на першій ітерації, а на кожній наступній – уточнюється), q_1 ($q_1 \leq 1$) – коефіцієнт зменшення кроку (якщо умова завершення спуску за напрямком виконується за перший крок), q_2 ($q_2 \geq 1$) – коефіцієнт збільшення кроку. Через кожні n_h кроків одновимірного спуску ($n_h > 1$) величина кроку збільшується в q_2 раз.

Для зупинки ітераційного процесу використовуються параметри ε_x і ε_g – алгоритм закінчує роботу в точці $z_{k^*} \in [z_k, z_{k+1}]$, якщо $\| z_{k+1} - z_k \| \leq \varepsilon_z$ (зупинка за аргументом), або $\| g_f(z_{k^*}) \| \leq \varepsilon_g$ (зупинка за нормою градієнта, яка використовується для гладких функцій). Використовуються також стандартна зупинка, якщо перевищено задану максимальну кількість ітерацій **maxitn**, та аварійна зупинка, яка сигналізує про те, що або функція $f(z)$ не є обмеженою знизу, або початковий крок h_0 занадто малий, і його потрібно збільшити.

Програмна реалізація $r(\alpha)$ -алгоритму з адаптивним регулюванням кроку за формулами (1) – (3) виконана octave-функцією **ralgb5** [9, 10]. Тут абревіатура "**b5**" означає, що коректується $n \times n$ -матриця B, а кожна ітерація вимагає $5n^2$ арифметичних операцій множення, які визначають обчислювальну складність однієї ітерації. Спрощена (для зручності використання) версія **ralgb5**, для якої зафіксовані два найбільш часто вживані параметри $q_2 = 1.1$ і $n_h = 3$, реалізована octave-функцією **ralgb5a** [11].

Якщо ітераційний процес запускається зі стартової точки z_0, то параметри $r(\alpha)$ -алгоритму рекомендується вибирати такими: $\alpha \in [2,\ 4]$, $q_1 = 1.0$ (для негладких функцій), $q_1 = 0.8 \div 0.95$ (для гладких функцій), $h_0 \approx \| z_0 - z^* \|$ – оцінка відстані від стартової точки z_0 до точки мінімуму z^*. Як правило, використовуються такі параметри зупинки: $\varepsilon_z \approx 10^{-6}$, $\left\| x_r - x^* \right\|$, **maxitn** $\approx 20n$. Тут параметр ε_g використовується для гладких функцій, а параметр ε_z – для негладких функцій.

При правильному виборі параметрів r -алгоритму кількість ітерацій, необхідна для знаходження точки z_{k^*}, для якої $f(z_{k^*}) - f^* \le \varepsilon$, емпірично оцінюється як $k^* = O(n \log \frac{1}{\varepsilon})$.

3. Алгоритм покращення евристичного розв'язку

Використовуючи точку, знайдену евристичним алгоритмом як стартову, та розв'язуючи для задачі пакування кругів відповідну задачу оптимізації за допомогою *r*-алгоритму з дихотомією кроку [1, 2], можна отримати кращі розв'язки за значенням радіусу зовнішнього круга.

Для цього задачу пакування кругів сформулюємо як таку модель нелінійного програмування:

$$R^* = \min_{R,x,y} R \tag{4}$$

за обмежень

$$x_i^2 + y_i^2 \le \left(R - r_i\right)^2, \quad i = 1,...,m, \tag{5}$$

$$\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 \ge \left(r_i + r_j\right)^2, \ 1 \le i < j \le m, \tag{6}$$

$$R \ge R_{low}, \tag{7}$$

де $R_{low} = \max\limits_{i=1,..,m} r_i$, $x = (x_1,\ldots,x_m)$, $y = (y_1,\ldots,y_m)$.

Модель (4)–(7) – багатоекстремальна квадратична задача, яка відповідає оптимальній упаковці нерівних кругів. Тут лінійна цільова функція (4) відповідає радіусу зовнішнього круга. Квадратичні нерівності (5) означають, що всі круги лежать всередині зовнішнього круга. Квадратичні нерівності (6) означають, що будь-які два круги не перетинаються між собою. Лінійна нерівність (7) дозволяє уникнути від'ємних значень для $R - r_i$, які підносяться до квадрату у правій частині обмежень (5).

Для пошуку локального мінімуму задачі (4)–(7) використаємо метод негладких штрафних функцій, який зводить задачу нелінійного програмування (4)–(7) до задачі безумовної мінімізації негладкої функції. Безумовна негладка задача оптимізації буде мати такий вигляд:

$$\min_{R,x,y} \left\{ f\left(R,x,y\right) = R + \Phi_p\left(R,x,y\right) \right\}, \tag{8}$$

де негладка штрафна функція $\Phi_p\left(R,x,y\right)$ має такий вигляд:

$$\Phi_P\left(R,x,y\right) = P_1 F_1\left(R,x,y\right) + P_2 F_2\left(x,y\right) + P_3 \max\left\{0, -R + R_{low}\right\}, \tag{9}$$

де P_1, P_2 та P_3 – додатні штрафні коефіцієнти, а функції $F_1\left(R,x,y\right)$ та $F_2\left(x,y\right)$ мають вигляд

$$F_1(R,x,y)=\sum_{i=1}^{m}\max\left\{0,x_i^2+y_i^2-(R-r_i)^2\right\}, \tag{10}$$

$$F_2(x,y)=\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\max\left\{0,-(x_i-x_j)^2-(y_i-y_j)^2+(r_i+r_j)^2\right\}. \tag{11}$$

Локальний мінімум у задачі (4)–(7) можна знайти, аналізуючи локальний мінімум безумовної негладкої задачі (8)–(11) для деяких додатних значень коефіцієнтів P_1, P_2 та P_3. Якщо в знайденій точці локального мінімуму функції $f(R,x,y)$ штрафна функція $\Phi_P(R,x,y)=0$, то знайдена точка є локальним мінімумом для задачі (4)–(7). Вибір штрафних коефіцієнтів P_1, P_2 та P_3 дозволяє коригувати порушення обмежень (5)–(7), P_1 застосовується до обмежень (5), P_2 до обмежень (6) та P_3 до обмеження (7). Зазначимо, що у розрахунках, результати яких наведені далі, використовувалися такі значення штрафних параметрів: $P_1=2000$, $P_2=2000$ та $P_3=1000$.

Для покращення результатів, отриманих за допомогою евристичного алгоритму, будемо використовувати *r*-алгоритм з дихотомією кроку, що дозволяє знаходити різні локальні мінімуми в задачі (4)–(7). Тут дихотомія кроку пояснюється тим, що оскільки в *r*-алгоритмі з адаптивним регулюванням кроку можна пропустити локальні екстремуми по напрямку руху, то не завжди великі значення кроку h будуть ефективними. Тому доцільно буде зменшувати величину кроку для того, щоб можна було врахувати локальні екстремуми, які могли бути пропущені.

Розглянемо алгоритм покращення евристичного розв'язку [2], де для знаходження локальних екстремумів негладкої штрафної функції (8) застосовується *r*-алгоритм з знайденої евристичним алгоритмом стартової точки $z_0=z^*=(R^*,x^*,y^*)\in\mathbb{R}^{2m+1}$. При описі алгоритму використаємо такі позначення: $h_{\max}$ – максимальне значення величини кроку *r*-алгоритму, $h_{\min}$ – мінімальне значення величини кроку *r*-алгоритму та Δ_R – значення відносного покращення величини радіуса зовнішнього круга.

Алгоритм має такий вигляд.

Крок 1. Задаємо $h_{\max}$, $h_{\min}$ та Δ_R. Покладемо $z:=z^*$ та $h:=h_{\max}$.

Крок 2. Запускаємо *r*-алгоритм зі стартової точки $z_0:=z$ з величиною початкового кроку $h_0:=h$ та знаходимо $z^{**}=(R^{**},x^{**},y^{**})\in\mathbb{R}^{2m+1}$ – локальний мінімум негладкої штрафної функції (8).

Крок 3. Перевіряємо: чи задовольняє знайдена точка z^{**} обмеженням (5), (6), та (7). Якщо не задовольняє, то переходимо до кроку 5.

Крок 4. Якщо $\dfrac{R^*-R^{**}}{R^*}\ge\Delta_R$ (відносне покращення радіусу зовнішнього круга не менше за значення Δ_R), то $z:=z^{**}$ (точка локального мінімуму стає стартовою точкою). Додаткова опція: встановлюємо $h:=h_{\max}$ та переходимо до кроку 2.

Крок 5. Покладемо $h:=0.5h$ (величину кроку зменшуємо вдвічі).

Крок 6. Якщо $h>h_{\min}$ (величина кроку більша ніж мінімальне значення кроку), то переходимо до кроку 2.

Крок 7. Закінчуємо роботу алгоритму з $z^{***}=(R^{***},x^{***},y^{***})=z\in\mathbb{R}^{2m+1}$ (покращення евристичного розв'язку обов'язково буде мати місце, якщо алгоритм хоча би один раз виконав крок 4).

Формулювання кроку 4 має додаткову опцію (версія 2 алгоритму): якщо відносне покращення

радіусу зовнішнього круга буде більшим ніж значення Δ_R , тобто $\frac{R^* - R^{**}}{R^*} \geq \Delta_R$, то для чергового запуску *r*-алгоритму стартовою точкою буде знайдена точка $z = z^{**}$ локального мінімуму задачі (4) – (7), а величина початкового кроку буде встановлено $h_0 = h_{\max}$ та переходимо до кроку 2. В протилежному випадку, величину кроку h зменшуємо вдвічі на кроці 5.

На рис. 2 показано блок-схему алгоритму покращення евристичного розв'язку. Версія 2 алгоритму передбачає додаткову опцію $(h = h_{\max})^*$, яка дозволяє відновлювати максимальне значення величини кроку *r*-алгоритму, якщо вдається досягнути заданої величини Δ_R для відносного покращення радіусу зовнішнього круга на кроці 4.

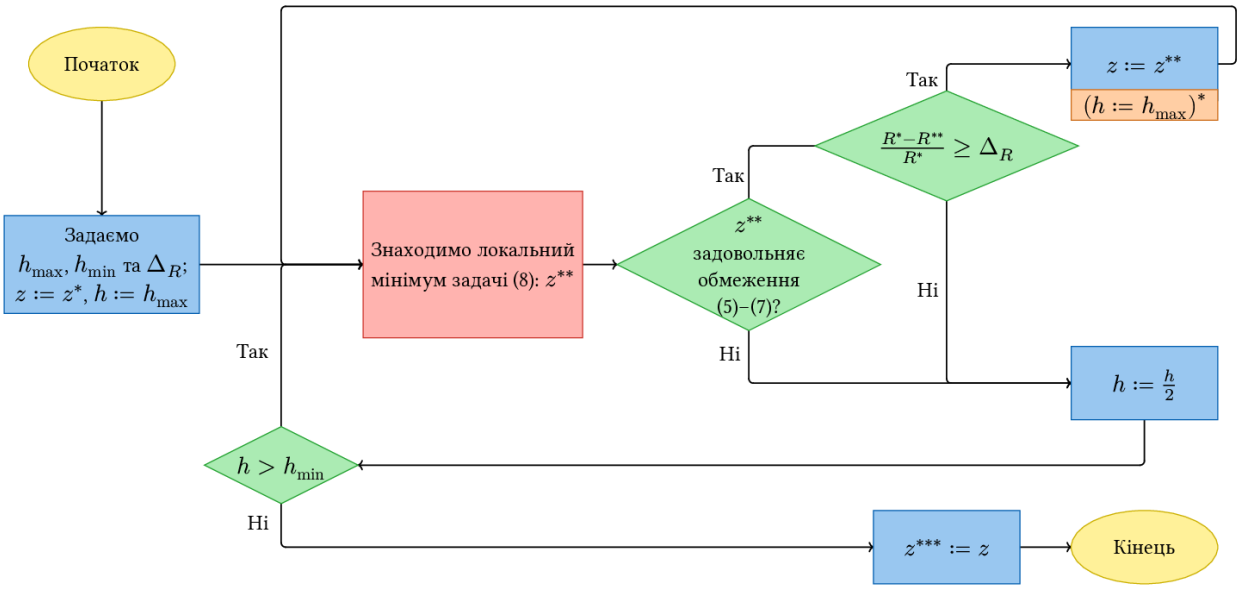

РИС. 2. Блок-схема алгоритму покращення евристичного розв'язку

Алгоритм реалізовано мовою програмування Rust версії 1.70.0 з використанням бібліотеки nalgebra 0.32.3. Для знаходження локальних мінімумів негладкої штрафної функції $f(z) = f(R, x, y)$ використано програмну реалізацію *r*-алгоритму **ralgb5a** [11]. Обчислення значення функції $f(\bar{z})$ та її узагальненого градієнта $g = g_f(\bar{z})$ в точці $\bar{z}$ реалізовано за допомогою функції **calcfg**.

4. Обчислювальні експерименти для покращення евристичного розв'язку конкурсних задач

Далі опишемо обчислювальні експерименти щодо покращення результатів, отриманих за допомогою евристичного алгоритму для тестових задач міжнародного конкурсу «Щільна упаковка кругів у круг мінімального радіусу» [8]. Для оцінювання програмних реалізацій алгоритму використано по десять тестів для $m = 10, 20, 30, 40, 50$. За кожен з цих 50 тестів нараховуються бали за наступною формулою:

$$\text{Кількість балів} = 100 \times \max\left\{0,\ 2 - \frac{R}{R^*}\right\}, \tag{12}$$

де R – знайдений радіус зовнішнього круга, R^* – найкращий (найменший) радіус зовнішнього круга, який було занесено у систему конкурсу.

У табл. 1 наведено результати роботи алгоритму при $\Delta_R = \{0, 10^{-5}, 10^{-4}, 10^{-3}\}$, $h_{\max} = 40.96$, $h_{\min} = 0.01$ щодо покращення евристичних розв'язків за кількістю отриманих балів для 50 конкурсних задач (по 10 тестових задач для $m = 10, 20, 30, 40, 50$). При цьому сумарна кількість балів для усіх 50 евристичних розв'язків була рівною 4832.04 балів. Параметри $r(\alpha)$-алгоритму вибиралися такими: $\alpha = 1.5$, $q_1 = 0.95$; параметри зупинки: $\varepsilon_z = 10^{-6}$, $\varepsilon_g = 10^{-7}$ та $maxitn = 15000$. Штрафні коефіцієнти були наступними: $P_1 = 2000$, $P_2 = 2000$ та $P_3 = 1000$. Тут *N_iter* – середня кількість ітерацій *r*-алгоритму для всіх 50 конкурсних тестових задач, яка розраховувалася за такою формулою:

$$N\_iter = \frac{\sum_{i=1}^{50} \frac{\text{Кількість ітерацій } r\text{-алгоритму для тесту } i}{\text{Кількість викликів } r\text{-алгоритму для тесту } i}}{50}. \qquad (13)$$

ТАБЛИЦЯ 1. Покращення евристичних розв'язків конкурсних задач

Δ_R	Алгоритм	Кількість балів	*N_iter*	Покращення в балах
10^{-3}	версія1	4 901.46	12 041.70	69.43
	версія 2	4 901.29	12 129.99	69.26
10^{-4}	версія1	4 905.47	11 543.00	73.43
	версія 2	4 908.10	12 006.31	76.07
10^{-5}	версія1	4 906.02	11 759.53	73.98
	версія 2	4 907.86	12 199.83	75.83
0.0	версія1	4 906.75	11 225.86	74.71
	версія 2	4 912.64	11 923.91	80.6
4832.04 – сумарна кількість балів для всіх 50 евристичних розв'язків				

Наведені в табл. 1 результати свідчать про приблизно однакову ефективність роботи алгоритму покращення евристичного розв'язку, причому версія 2 алгоритму показує більші покращення в балах для $\Delta_R = \{0, 10^{-5}, 10^{-4}\}$. Так при $\Delta_R = 10^{-4}$ покращення складає 76.07–73.43 = 2.64 балів, при $\Delta_R = 10^{-5}$ – 75.83–73.98 = 1.85 балів, а при $\Delta_R = 0$ покращення складає 80.06–74.71 = 5.35 балів. Версія 2 алгоритму програє версії 1 тільки при $\Delta_R = 10^{-3}$, де цей програш є невеликим та складає 69.43–69.26 = 0.17 балів.

Зауважимо, що зменшення значень відносного покращення величини радіуса зовнішнього круга від $\Delta_R = 10^{-3}$ до $\Delta_R = 0$ призводить до суттєвих покращень роботи алгоритму. Так, для алгоритму таке покращення складає 74.71–69.43 = **5.28** балів, а для версії 2 алгоритму покращення є більшим та складає 80.6–69.26 = **11.34** балів. При цьому середня кількість ітерацій є приблизно однаковою для двох версій та складає від 11225 до 12543 ітерацій.

Ефективність алгоритму залежить від вибору параметрів *r*-алгоритму: α – коефіцієнту розтягу простору та q_1 – коефіцієнту зменшення крокового множника. У табл. 2 наведено розраховані за формулами (12) та (13) кількості отриманих балів та затрачені середні кількості ітерацій алгоритмами 1 та 2 при використанні $\Delta_R = 0$ та $\Delta_R = 10^{-5}$ та всіх можливих комбінацій коефіцієнтів розтягу

простору $\alpha \in \{1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$ та коефіцієнтів зменшення крокового множника $q_1 \in \{0.80, 0.85, 0.90, 0.95, 1.0\}$. При розрахунках використовувалися $h_{\max} = 40.96$, $h_{\min} = 0.01$, $\varepsilon_z = 10^{-6}$, $\varepsilon_g = 10^{-7}$, $maxitn = 15000$, $P_1 = 2000$, $P_2 = 2000$ та $P_3 = 1000$.

ТАБЛИЦЯ 2. Порівняння роботи алгоритму для версії 1 та 2 при різних значеннях α та q_1

α	q_1	$\Delta_R = 0$				$\Delta_R = 10^{-5}$			
		Версія 1		Версія 2		Версія 1		Версія 2	
		Бали	*N*_iter	Бали	*N*_iter	Бали	*N*_iter	Бали	*N*_iter
1.5	0.80	4 900.19	7574.92	4 903.88	9152.12	4 900.17	8371.66	4 901.39	9643.32
1.5	0.85	4 898.71	7842.48	4 909.80	9359.58	4 899.88	8469.46	**4 908.10**	**9834.5**
1.5	0.90	4 899.51	9154.44	4 908.64	10756.64	4 899.94	9638.58	4 903.21	11018.98
1.5	0.95	**4 906.75**	**11225.84**	**4 912.64**	**11923.9**	**4 906.02**	**11759.52**	**4 907.86**	**12199.82**
1.5	1.00	4 902.13	14827.82	**4 912.79**	**14861.16**	4 901.71	14872.3	**4 909.98**	**14874.8**
2	0.80	4 903.41	11598.38	4 910.39	12686.6	4 901.13	11948.64	4 907.13	12913.7
2	0.85	4 904.06	12394.3	4 910.09	13250.92	4 903.47	12788.36	**4 909.53**	**13309.86**
2	0.90	4 902.86	13212.14	4 910.02	13483.58	4 902.84	13420.9	**4 911.29**	**13577.78**
2	0.95	4 902.17	14065.32	4 909.01	14135.34	4 902.07	14181.54	4 907.44	14196.9
2	1.00	4 901.40	14260.48	4 911.52	14260.76	4 901.35	14260.42	4 906.71	14260.58
2.5	0.80	4 894.98	13128.18	4 899.21	13389	4 894.96	13232.3	4 898.97	13401.1
2.5	0.85	4 900.23	13271.04	4 906.60	13498.72	4 900.45	13278.92	4 903.34	13496.7
2.5	0.90	4 895.57	13549.34	4 904.74	13625.04	4 896.51	13590.56	4 906.86	13647.66
2.5	0.95	4 897.95	13682.84	4 905.88	13701.96	4 897.51	13682.06	4 904.34	13700.76
2.5	1.00	4 898.58	13711.64	4 909.24	13711.94	4 899.97	13711.7	**4 908.35**	**13711.94**
3	0.80	4 892.18	13053.44	4 900.26	13146.6	4 898.05	13131.52	4 900.14	13179.92
3	0.85	4 891.61	13134.5	4 899.13	13159.4	4 894.54	13128.66	4 897.37	13171.38
3	0.90	4 893.30	13165.44	4 898.49	13202.72	4 891.44	13157.6	4 899.84	13197.96
3	0.95	4 900.60	13229.44	4 904.63	13219.56	4 900.58	13229.4	4 904.93	13219.62
3	1.00	4 893.61	13215.54	4 901.35	13215.26	4 893.39	13215.6	4 899.50	13215.42
3.5	0.80	4 889.40	12691.58	4 895.80	12694.02	4 889.38	12681.6	4 895.78	12690.32
3.5	0.85	4 891.34	12675.32	4 894.15	12683.72	4 890.55	12687.14	4 893.36	12693.5
3.5	0.90	4 893.77	12695.44	4 895.99	12689.58	4 893.74	12695.34	4 895.96	12689.5
3.5	0.95	4 891.28	12696.96	4 895.09	12696.62	4 892.88	12697.12	4 895.10	12696.48
3.5	1.00	4 888.57	12702.46	4 898.80	12699.4	4 888.46	12696.98	4 897.07	12697.2
4	0.80	4 887.77	12351.76	4 892.07	12355.5	4 888.55	12352.04	4 892.85	12356.58
4	0.85	4 890.59	12366.74	4 891.99	12354.38	4 890.57	12368.84	4 891.97	12354.82
4	0.90	4 894.18	12351.98	4 899.12	12350.14	4 892.70	12351.96	4 898.77	12351.32
4	0.95	4 889.30	12348.54	4 894.94	12340.6	4 888.79	12348.58	4 898.87	12342.64
4	1.00	4 887.93	12341	4 890.23	12342.78	4 887.85	12340.98	4 890.48	12342.76

Наведені в табл. 2 результати свідчать про те, що неможливо виділити певну пару параметрів α та q_1, використання яких призводить до суттєвого покращення евристичних розв'язків для всіх 50 конкурсних задач. Це випливає із порівняння отриманих балів для рядка з параметрами $\alpha=1.5$, $q_1=0.95$, які відображені в табл. 1, з виділеними жирним шрифтом отриманими балами, що переважають результати з табл. 1. Так якщо версія 1 не покращує кількість балів як при значенні $\Delta_R=0$, так і при значенні $\Delta_R=10^{-5}$, то версія 2 при значенні $\Delta_R=0$ забезпечує кращу кількість балів для одніє пари параметрів $(\alpha,q_1)=(1.5,1.0)$, а при значенні $\Delta_R=10^{-5}$ – для п'яти пар параметрів $(\alpha,q_1)\in\{(1.5,0.85),\ (1.5,1.0),\ (2.0,0.85),\ (2.0,0.9),\ (2.5,1.0)\}$. При цьому для параметрів $(\alpha,q_1)=(2.0,0.9)$ покращення в кількості балів є значним та складає $4911.29-4907.86=3.43$ балів, а середня кількість ітерацій збільшується на $13577.78-12199.82=1377.96$ ітерацій. Це сигналізує про те, що для кожної окремої задачі можна виділити таку пару параметрів α та q_1, використання яких дозволить суттєво покращити евристичний розв'язок за допомогою алгоритму 1 або алгоритму 2.

На рис. 3 показано залежності кількостей балів, отриманих версією 2 алгоритму при $\Delta_R=0$, від параметра максимальної кількості ітерацій *r*-алгоритму, де $maxitn\in\{15000,30000,50000,100000\}$. При розрахунках використовувались такі параметри зупинки: $\varepsilon_z=10^{-6}$, $\varepsilon_g=10^{-7}$.

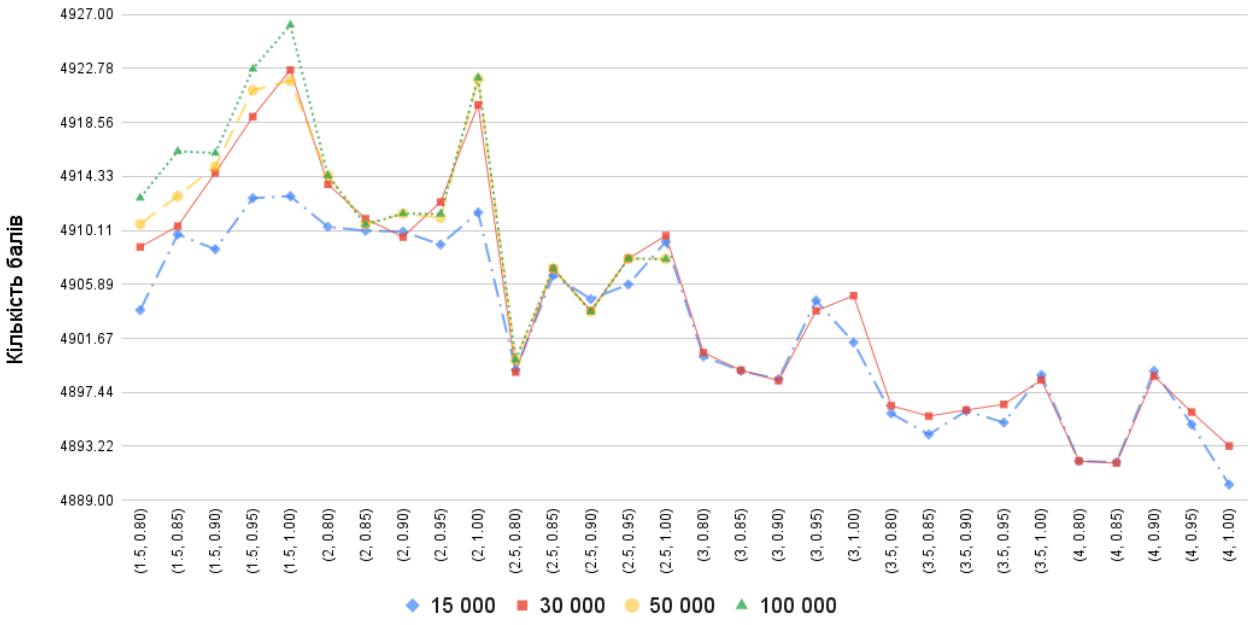

РИС. 3. Залежності кількостей балів алгоритму 2 ($\Delta_R=0$) від параметрів (α,q_1) та *maxitn*

Із рис. 3 видно, що при збільшенні коефіцієнта α зменшується кількість балів, отриманих алгоритмом 2. При цьому найкращі результати досягаються при $\alpha=1.5$, $q_1\in\{0.95,1.0\}$.

На рис. 4 показано порівняння роботи версій 1 та 2 алгоритму за кількістю балів у залежності від великої кількості ітерацій евристичного алгоритму (від 100.000 до 400.000) для значень відносного покращення величини радіуса зовнішнього круга $\Delta_R=10^{-5}$ та $\Delta_R=10^{-3}$. Параметри $r(\alpha)$-алгоритму вибиралися такими: $\alpha=1.5$, $q_1=0.95$; параметри зупинки: $\varepsilon_z=10^{-6}$, $\varepsilon_g=10^{-7}$.

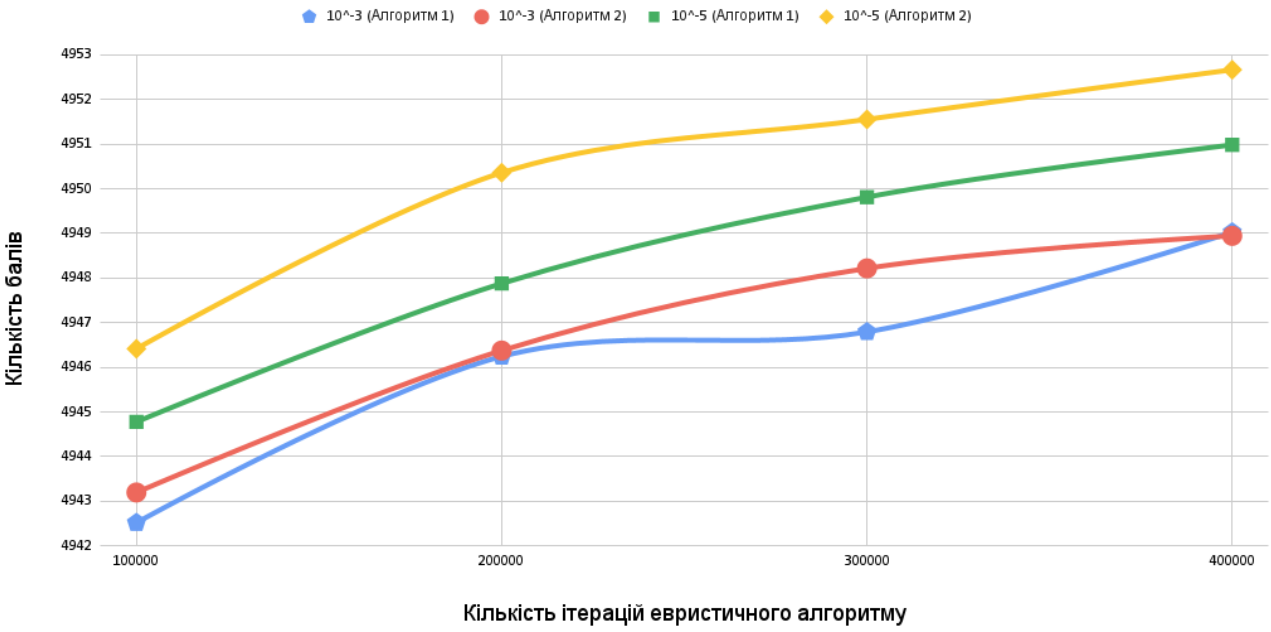

РИС. 4. Порівняння версій 1 та 2 для великої кількості ітерацій евристичного алгоритму

Показані на рис. 4 результати свідчать про ефективність роботи обох версій алгоритму, причому версія 2 демонструє більші покращення у балах незалежно від значення Δ_R. Зменшення величини Δ_R від $\Delta_R = 10^{-3}$ до $\Delta_R = 10^{-5}$ призводить до суттєвих покращень роботи обох версій. Для $\alpha = 1.5$ та $q_1 = 1.0$ спостерігаються схожі результати.

5. Використання *r*-алгоритму для чисел великої розрядності

Далі наведено опис результатів використання *r*-алгоритму для чисел великої розрядності (128 та 256 біт), що дозволяє знаходити мінімальний радіус зовнішнього круга з точністю до 10–20 цифр після коми. Обчислювальні експерименти проводилися за допомогою мови програмування Julia для тестової задачі з $m = 5$ кругами, радіуси яких дорівнюють 1, 2, 3, 4 та 5 відповідно. Результати розрахунку порівнювалися з розв'язком з веб-сайта http://www.packomania.com/ [12], що наведено у табл. 3.

ТАБЛИЦЯ 3. Розміщення кругів для тестового прикладу з веб-сайту www.packomania.com

i	r_i	$\frac{x_i}{R^*}$	$\frac{y_i}{R^*}$
1	1.0	0.71389257123241763122170169551 1	-0.34415584827387341158568858391 8
2	2.0	-0.68186597809504681260463738413 3	0.16444003262303818864666206527 8
3	3.0	0.64630533082747746103821104177 7	0.16371585215145096980242664189 1
4	4.0	-0.02572491186987950830179202104 8	0.55502872984784579736592320538 0
5	5.0	-0.00170561944114529463593830611 0	-0.44452743951028385188514036438 7
R^*		9.00139774605021931867244423 69	

В табл. 4 наведено результати впливу розрядності (32, 64, 128 та 256 біт) на виконання обмежень (5) та (6) для $m = 5$ з табл. 3. Якщо якесь із обмежень є порушеним, то відповідна йому нев'язка виділена жирним шрифтом.

ТАБЛИЦЯ 4. Перевірка виконання обмежень (14) та (15) – **Julia** (32, 64, 128 та 256 біт**)**

Номери кругів		32 біти	64 біти	128 біт	256 біт
круг i		Виконання обмежень (5)			
1		–1.313161e+01	–1.313161e+01	–1.313161e+01	–1.313161e+01
2		–9.156666e+00	–9.156670e+00	–9.156671e+00	–9.156671e+00
3		**7.629395e–06**	–7.629395e–06	–4.952827e–15	–4.952827e–15
4		–5.722046e–06	**3.814697e–06**	–2.152343e–15	–2.152343e–15
5		–3.814697e–06	0.000000e+00	–2.571545e–15	–2.571545e–15
круг i	круг j	Виконання обмежень (6)			
1	2	1.698073e+02	1.698073e+02	1.698073e+02	1.698073e+02
1	3	5.269249e+00	5.269243e+00	5.269243e+00	5.269243e+00
1	4	8.483502e+01	8.483503e+01	8.483503e+01	8.483503e+01
1	5	6.307716e+00	6.307712e+00	6.307712e+00	6.307712e+00
2	3	1.179316e+02	1.179316e+02	1.179316e+02	1.179316e+02
2	4	1.124421e+01	1.124420e+01	1.124420e+01	1.124420e+01
2	5	1.853119e+01	1.853119e+01	1.853119e+01	1.853119e+01
3	4	7.629395e–06	1.421085e–14	1.252090e–15	1.252090e–15
3	5	1.525879e–05	**0.000000e+00**	**–2.012473e–16**	**–2.012473e–16**
4	5	**–7.629395e–06**	1.421085e–14	3.880818e–15	3.880818e–15

Із табл. 4 видно, що обмеження (5) для $m = 5$, які означають що кожний круг повинен бути всередині зовнішнього круга, не виконуються з точністю 10^{-6} тільки для 32-бітних обчислень (третій круг, точність 7.629395e–06) та 64-бітних обчислень (4 круг, точність 3.814697e–06). Обмеження (6), які відповідають неперетину пар кругів, не виконуються для 32-бітних обчислень (четвертий та п'ятий круг, точність –7.629395e–06), для 128-бітних та 256-бітних обчислень (третій та п'ятий круг, точність –2.012473e–16).

В табл. 5 наведено результати використання *r*-алгоритму для знаходження найкращого радіусу зовнішнього круга для тестової задачі з $m = 5$ та порівняння його з R^* із табл. 3. Для цього була створена програма на мові Julia. При розрахунках *r*-алгоритм запускався $ntest \in \{50,100\}$ разів, де для стартової точки радіус зовнішнього круга вибирався рівним $R = \frac{3}{\sqrt{7}}\sqrt{1+2^2+3^2+4^2+5^2}$, а координати розміщення кругів генерувалися випадковим чином з рівномірним розподілом із діапазону $[-0.5R, 0.5R]$. Параметри r-алгоритму вибиралися такими: $h_0 = 1.0$, $\alpha = 2.0$, $q_1 = 1.0$ та $maxitn = 5000$. Штрафні коефіцієнти були наступними: $P_1 = P_2 = P_3 = 100$.

З табл. 5 видно що з ростом бітності для обчислень середня кількість ітерацій зменшується, а точність знаходження найкращого радіусу зовнішнього круга зростає. В колонці «найкращий знайдений радіус» жирним шрифтом виділені цифри, які співпадають з цифрами для R^* з табл. 3. У колонці «максимальне порушення обмежень» наведено нев'язку до максимального із порушених обмежень (14) та (15). Так, наприклад, при 128 та 256 бітних обчисленнях та точності $\varepsilon_z = 10^{-30}$ усі наведені цифри для найкращого знайденого радіуса співпадають з цифрами для R^* в табл. 3. При цьому обмеження (5) виконуються з такою точністю, що ні один із внутрішніх кругів не торкається межі зовнішнього круга, а обмеження (6) виконуються з точністю, що довільні два внутрішніх круги не перетинаються між собою. Це демонструє, що використання *r*-алгоритму для чисел великої розрядності (128 та 256 біт) дозволяє знаходити мінімальний радіус зовнішнього круга з точністю до 10–20 цифр після коми. З підвищенням розрядності чисел кількість ітерацій *r*-алгоритму зменшується, а точність знаходження радіусу зовнішнього круга зростає. При правильно підібраних параметрах *r*-алгоритм дозволяє знайти для радіуса зовнішнього круга всі 28 цифр після коми, які для задачі з 5-ма кругами представлені на веб-сайті http://www.packomania.com/.

ТАБЛИЦЯ 5. Знаходження найкращих радіусів $r(\alpha)$-алгоритмом: **Julia** (32, 64, 128 та 256 біт**)**

ε_x	ntest	Бітність	Середня кількість ітерацій	Найкращий знайдений радіус	Максимальне порушення обмежень
1.0e–12	50	32	4802.64	**9.00139**8086547851562500000000	–7.62939e–06
1.0e–12	100	32	4850.05	**9.00139**8086547851562500000000	–7.62939e–06
1.0e–30	50	32	4914.46	**9.00139**8086547851562500000000	–7.62939e–06
1.0e–30	100	32	4957.23	**9.00139**8086547851562500000000	–7.62939e–06
1.0e–12	50	64	4581.06	**9.0013977460502**214711368651479	0.00000e+00
1.0e–12	100	64	4581.01	**9.0013977460502196**947800257476	0.00000e+00
1.0e–30	50	64	4913.24	**9.0013977460502**214711368651479	0.00000e+00
1.0e–30	100	64	4924.8	**9.0013977460502196**947800257476	0.00000e+00
1.0e–12	50	128	1746.5	**9.00139774605021931868**29737093	0.00000e+00
1.0e–12	100	128	1832.94	**9.00139774605021931868**29737093	0.00000e+00
1.0e–30	50	128	5000.0	**9.0013977460502193186724442369**	0.00000e+00
1.0e–30	100	128	4970.58	**9.0013977460502193186724442369**	0.00000e+00
1.0e–12	50	256	1220.66	**9.00139774605021931868**15560850	0.00000e+00
1.0e–12	100	256	1191.82	**9.00139774605021931867**25902899	0.00000e+00
1.0e–30	50	256	4601.5	**9.0013977460502193186724442369**	0.00000e+00
1.0e–30	100	256	4615.06	**9.0013977460502193186724442369**	0.00000e+00

На рис. 5 показано розв'язок задачі з сайту http://www.packomania.com/ (табл. 3) та два найкращі розв'язки, які знайдені r-алгоритмом для обчислень великої розрядності (128 та 256 біт).

Із цього рисунку видно, що розв'язки тестової задачі з 5 кругами відрізняються за розміщенням центрів кругів, що пов'язано з тим, що задача (4)–(7) має багато розв'язків.

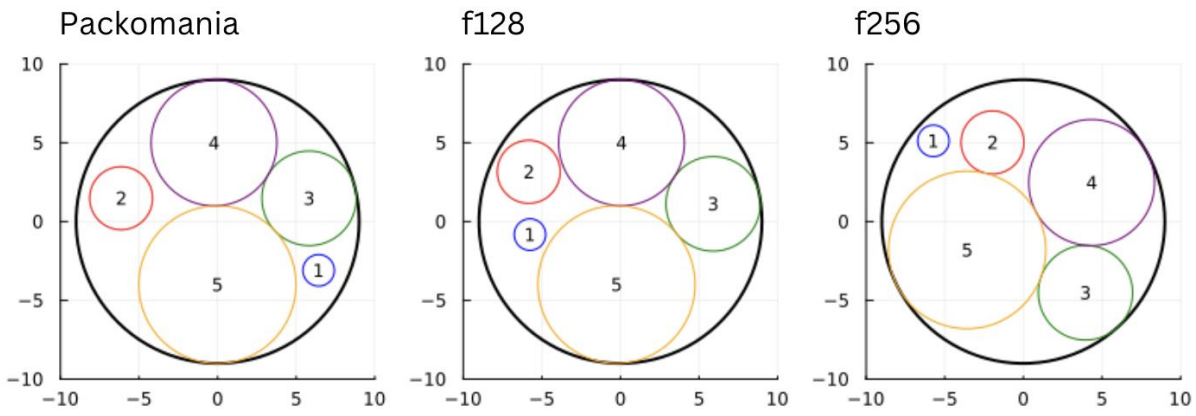

РИС. 5. Три розв’язки для тестової задачі з 5 кругами, радіуси яких дорівнюють 1, 2, 3, 4 та 5

Висновки. У статті описано досвід використання *r*-алгоритму для знаходження та покращення локальних мінімумів в оптимізаційних задачах упаковки декількох десятків нерівних кругів у круг мінімального радіусу. Досліджено два способи використання *r*-алгоритму. Перший спосіб пов'язаний з багаторазовим запуском *r*-алгоритму з дихотомією кроку зі стартової точки, в якій досягається знайдений евристичним алгоритмом найменший радіус зовнішнього круга. Другий спосіб пов'язаний з використання *r*-алгоритму задля підвищення точності за значенням радіуса зовнішнього круга локальних мінімумів багатоекстремальної негладкої функції та зменшення кількості ітерацій *r*-алгоритму за рахунок підвищення розрядності чисел.

Для реалізації першого способу розроблено алгоритм який застосовує дві версії управління величиною початкового кроку для *r*-алгоритму. Для першої версії величина кроку зменшується вдвічі по ходу ітераційного процесу, а для другої версії величина кроку вибирається рівною максимальному значенню величини початкового кроку, якщо вдається досягнути заданої величини для відносного покращення радіусу зовнішнього круга. Обидві версії алгоритму реалізовано мовою програмування Rust 1.70.0 з використанням бібліотеки nalgebra 0.32.3. Наведено результати застосування обох версій для 50 конкурсних задач (m = 10,20,30,40,50), проведено їх аналіз за кількістю отриманих балів та середньою кількістю ітерацій *r*-алгоритму.

Другий спосіб застосовує *r*-алгоритм з адаптивним способом регулювання кроку для знаходження найкращого локального мінімуму багатоекстремальної негладкої функції з заданої кількості випадкових стартових точок [13, 14]. Він реалізований мовою Julia та використовує числа великої розрядності (128 та 256 біт). Обчислювальні експерименти проводилися для тестової задачі з m = 5 кругами, радіуси яких дорівнюють 1, 2, 3, 4 та 5 відповідно. Результати порівнювалися з розв’язками, наведеними на веб-сайті http://www.packomania.com/. Показано, що з підвищенням розрядності чисел кількість ітерацій *r*-алгоритму зменшується, а точність знаходження радіусу зовнішнього круга зростає.

Фінансування. Робота підтримана грантом Volkswagen Foundation (грант № 97775).

Авторські внески. Стецюк П.І., Романова Т.Є., Шеховцов С.Б. – концептуалізація, методологія, керівництво; Задорожний Б.О., Тиводар С.Р. – дослідження, узагальнення, формальний аналіз, алгоритмічне та програмне забезпечення, чисельні експерименти, візуалізація; Стецюк П.І., Романова Т.Є., Задорожний Б.О. Тиводар С.Р., Шеховцов С.Б. – написання – оригінальна чернетка, редагування.

Список літератури

1. Задорожний Б.О., Міца О.В., Стецюк П.І. Про покращення евристичного алгоритму упаковки кругів в круг мінімального радіусу. *Cybernetics and Computer Technologies*. 2023. 2. С. 32–45. https://doi.org/10.34229/2707-451X.23.2.4
2. Задорожний Б.О., Романова Т.Є., Стецюк П.І. Покращення евристичного алгоритму пакування нерівних кругів із застосуванням r-алгоритму Шора. Матеріали XXVI Міжнародного науково-практичного семінару «*Комбінаторні конфігурації та їхні застосування*» (Кропивницький–Запоріжжя–Київ, 13-15 червня 2024 року). Кропивницький: ПП «ЕксклюзивСистем», 2024. С. 59–65.
3. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наукова думка, 1979. 200 с.
4. Shor N.Z. Non-Differentiable Optimization and Polynomial Problems. Kluwer Academic Publishers, Boston, Dordrecht, London. 1998. 412 p.
5. Стецюк П.І., Романова Т.Є., Тиводар С.Р. Використання солвера BARON для розв'язання квадратичної задачі оптимальної упаковки нерівних кругів. Матеріали XXVI Міжнародного науково-практичного семінару «*Комбінаторні конфігурації та їхні застосування*» (Кропивницький–Запоріжжя–Київ, 13-15 червня 2024 року). Кропивницький: ПП «ЕксклюзивСистем», 2024. С. 179–188.
6. Tawarmalani M., Sahinidis N.V. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*. 2005. 103 (2). P. 225–249.
7. Sahinidis N.V., BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs, User's manual, 2021.
8. Щільна упаковка кругів в круг мінімального радіусу. Eolymp. https://packing-circles.eolymp.io (звернення: 23.05.2024).
9. Стецюк П.И. Теория и программные реализации *r*-алгоритмов Шора. *Кибернетика и системный анализ*. 2017. № 5. С. 43–57.
10. Стецюк П.И. Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций. *Вычислительные технологии*. 2017. Т. 22, № 2. С. 127–149.
11. Стецюк П.І. Комп'ютерна програма "Octave-програма ralgb5a: r(α)-алгоритм з адаптивним кроком". Свідоцтво про реєстрацію авторського права на твір № 85010. Україна. Міністерство економічного розвитку і торгівлі. Державний департамент інтелектуальної власності. Дата реєстрації 29.01.2019.
12. Packomania. http://www.packomania.com/(звернення: 09.08.2023)
13. Romanova T., Pankratov A., Litvinchev I., Stetsyuk P., Lykhovyd O., Marmolejo-Saucedo J.A., Vasant P. Balanced Circular Packing Problems with Distance Constraints. *Computation*. 2022. Vol. 10, Iss. 7. P. 113. https://doi.org/10.3390/computation10070113
14. Романова Т.Є., Стецюк П.І., Чугай А.М., Шеховцов С.Б. Технології паралельних обчислень для розв'язання оптимізаційних задач геометричного проектування. *Кибернетика и системный анализ*. 2019. № 6. С. 17–29.

Одержано 05.09.2024

Задорожний Богдан Олександрович,
студент Ужгородського національного університету,
bohzador@gmail.com

Романова Тетяна Євгеніївна,
доктор технічних наук, професор,
запрошений професор Університету Лідса, Велика Британія,
https://orcid.org/0000-0002-8618-4917
t.romanova@leeds.ac.uk

Стецюк Петро Іванович,
доктор фізико-математичних наук, завідувач відділу методів негладкої оптимізації
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
stetsyukp@gmail.com
https://orcid.org/0000-0003-4036-2543

Тиводар Станіслав Романович,
аспірант Ужгородського національного університету,
stanislav.tyvodar@uzhnu.edu.ua

Шеховцов Сергій Борисович,
кандидат технічних наук, доцент,
доцент Харківського національного університету радіоелектроніки.
https://orcid.org/0000-0003-2381-7999
serhii.shekhovtsov@nure.ua

УДК 519.85

Б.О. Задорожний [1 *], Т.Є. Романова [2], П.І. Стецюк [1, 3], С.Р. Тиводар [1], С.Б. Шеховцов [4]

Використання *r*-алгоритму для пакування нерівних кругів у круг мінімального радіусу

[1] Ужгородський національний університет
[2] Університет Лідса, Велика Британія
[3] Інститут кібернетики імені В.М. Глушкова НАН України, Київ
[4] Харківський національний університет радіоелектроніки
[*] Листування: bohzador@gmail.com

Досліджено два способи використання *r*-алгоритму Шора для задачі пакування нерівних кругів в зовнішній круг мінімального радіусу. Перший спосіб реалізує багаторазовий запуск *r*-алгоритму з дихотомією кроку із допустимої стартової точки, в якій досягається знайдений евристичним алгоритмом найменший радіус зовнішнього круга. Розглядаються дві версії алгоритму. Для першої версії величина кроку зменшується вдвічі по ходу ітераційного процесу, а для другої – додається опція, яка дозволяє відновлювати максимальне значення величини кроку *r*-алгоритму. Алгоритм реалізовано мовою програмування Rust з використанням бібліотеки nalgebra.

Другий спосіб має на меті дослідження щодо підвищення точності за значенням радіуса зовнішнього круга локальних мінімумів багатоекстремальної негладкої функції та зменшення числа ітерацій *r*-алгоритму за рахунок підвищення розрядності чисел (128 та 256 біт). Відповідний алгоритм реалізований мовою програмування Julia. Обчислювальні експерименти проводилися для тестової задачі з п'ятю кругами та порівнювалися з розв'язками, наведеними на веб-сайті http://www.packomania.com/. Показано, що з підвищенням розрядності чисел кількість ітерацій *r*-алгоритму зменшується, а точність знаходження радіусу зовнішнього круга зростає. При правильно підібраних параметрах *r*-алгоритм знаходить всі 28 цифр після коми, які представлені на веб-сайті http://www.packomania.com/.

Ключові слова: пакування кругів, *r*-алгоритм, евристичний алгоритм, Rust, Julia.

UDC 519.85

Bohdan Zadorozhnyi [1 *], Tetyana Romanova [2], Petro Stetsyuk [1, 3], Stanislav Tyvodar [1], Sergiy Shekhovtsov [4]

Packing Unequal Circles into a Minimum-Radius Circle Using *r*-Algorithm

[1] Uzhhorod National University,
[2] University of Leeds, UK
[3] V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
[4] Kharkiv National University of Radioelectronics
[*] Correspondence: bohzador@gmail.com

Two approaches to employ the Shor's *r*-algorithm for solving a problem of packing unequal circles into a minimum-radius circle are studied.

The first approach uses multistart of the r-algorithm with a step dichotomy from a set of feasible starting points. Each feasible point is taken as the best solution found by a heuristic algorithm. Two versions of the algorithm are considered. For the first version, the step value is halved during the iteration process. The second version provides an option that allows to restore the maximum value of the *r*-algorithm step value. The algorithm

is implemented using Rust 1.70.0 programming language and nalgebra 0.32.3 library. Both versions of the algorithm are tested for 50 test problems of the international competition “Dense packing of circles in a circle of minimum radius” to improve the results found by the heuristic. In most cases, the second version showed better solutions.

The second approach employs the *r*-algorithm with an adaptive step to find the best local minimum of a multiextremal nonsmooth function using multistart strategy from a set of randomly chosen starting points. It is implemented using Julia programming language and uses large numbers (128 and 256 bits). Computational experiments are tested for a benchmark problem with five circles. These results are compared to the problem solutions provided on the website http://www.packomania.com/. It is shown that increasing the bit depth leads to decreasing the number of the *r*-algorithm iterations while increasing the accuracy of the objective function value. With correctly chosen parameters, the *r*-algorithm finds all 28 digits after the decimal point, which are presented on the website http://www.packomania.com/.

Keywords: circle packing, *r*-algorithm, heuristic algorithm, Rust, Julia.