КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.85 DOI:10.34229/2707-451X.23.2.4

Б.О. ЗАДОРОЖНИЙ, О.В. МІЦА, П.І. СТЕЦЮК

ПРО ПОКРАЩЕННЯ ЕВРИСТИЧНОГО АЛГОРИТМУ УПАКОВКИ КРУГІВ В КРУГ МІНІМАЛЬНОГО РАДІУСУ

Робота присвячена дослідженню евристичного алгоритму для розв'язання конкурсної задачі «Щільна упаковка кругів в круг мінімального радіусу» та розробці його покращеного варіанту за допомогою r-алгоритму Шора з дихотомією кроку. Для програмної реалізації алгоритмів використовувались мова програмування Python 3.9.5 та бібліотека NumPy 1.24.2. Проведено аналіз отриманих результатів за такими параметрами: кількість отриманих балів, час виконання.

Ключові слова: *пакування кругів, евристика, r-алгоритм, Python, NumPy.*

Вступ. Для більшого зацікавлення молоді наукою з 1 листопада до 21 грудня 2022 року проводився конкурс-змагання з розв'язання задачі «Щільної упаковки кругів в круг мінімального радіусу». Ця задача є досить складною проблемою, яка потребує чималих обчислень та яка має багато різних підходів до її розв'язання. Як наслідок цього, подальші дослідження в галузі цієї проблематики можуть активізувати розвиток нових математичних методів, алгоритмів та їх програмних реалізацій. Вони у свою чергу можуть використовуватися при розв'язанні інших складних задач у різноманітних галузях та застосунках.

Організатори конкурсу – Інститут кібернетики ім. В.М. Глушкова НАНУ (Олександр Хіміч, Петро Стецюк, Олена Ніколаєвська), Ужгородський національний університет (Олександр Міца), Інститут проблем машинобудування ім. А.М. Підгорного НАНУ (Тетяна Романова, Георгій Яськов) та Дрезденський технічний університет (Андреас Фішер). У конкурсі зареєструвалися 190 учасників із 80 навчальних закладів 10 країн світу – України, Німеччини, США, Грузії, Польщі, Франції, Румунії, Італії, Непалу й Мальти.

Друге місце у змаганні посів студент третього курсу Ужгородського національного університету Богдан Задорожний, програма якого використовувала розроблений ним евристичний алгоритм. Програма показала результат в 4904 бали, для чого їй знадобилося 84 запуски зі 100. Час роботи програми для кожного тесту був менше 2 секунд.

Отже залишок часу можна використати на покращення розв'язку, знайденого евристичним алгоритмом. Для цього проведено дослідження, як за допомогою *r*-алгоритму Шора можна покращити ефективність роботи розробленого Б. Задорожним евристичного алгоритму. У даній роботі описано: евристичний алгоритм (розділ 1), його покращення за допомогою *r*-алгоритму з дихотомією кроку (розділ 2), порівняльний аналіз програм за кількістю отриманих балів та за часом виконання для 50 задач конкурсу (розділ 3).

1. Задача та евристичний алгоритм. Задача з конкурсу «Щільна упаковка кругів в круг мінімального радіусу» була сформульована наступним чином.

Задано набір N кругів з радіусами r_i, $i=1,2,\ldots,N$. Потрібно знайти центри $(x_i;y_i)$, $i=1,2,\ldots,N$ цих кругів і радіус R зовнішнього круга з центром $(0;0)$ такі, що:

- кожний з кругів $i=1,2,\ldots,N$ має повністю знаходитись всередині зовнішнього круга (круги можуть торкатися межі зовнішнього круга);
- для будь-якої пари $(i;j)$, де $i,j\in\{1,\ldots,N\}$ та $i<j$, круги i,j не перетинаються (їм дозволено торкатися один одного);
- радіус R зовнішнього круга має бути якомога меншим.

Для оцінювання програми використовувалися по десять тестів для десяти кругів $N=10$, для двадцяти кругів $N=20$, для тридцяти кругів $N=30$, для сорока кругів $N=40$ та для п'ятдесяти кругів $N=50$. За кожен з цих 50 тестів нараховувалися бали за наступною формулою:

$$\text{кількість балів}=\left[\max\left(0,\left(2-\frac{R}{R^*}\right)\right)\right], \tag{1}$$

де $[x]$ – округлення x до цілого, R – знайдений радіус зовнішнього круга, R^* – найкращий (найменший) радіус зовнішнього круга, який було занесено у систему конкурсу. Для всіх 50 тестів найкращі радіуси зовнішнього круга та розміщення в них заданих кругів були знайдені за допомогою методів та програм, розроблених у відділі Математичного Моделювання й Оптимального Проектування Інституту проблем машинобудування ім. А.М. Підгірного НАН України під керівництвом члена-кореспондента НАН України Юрія Стояна.

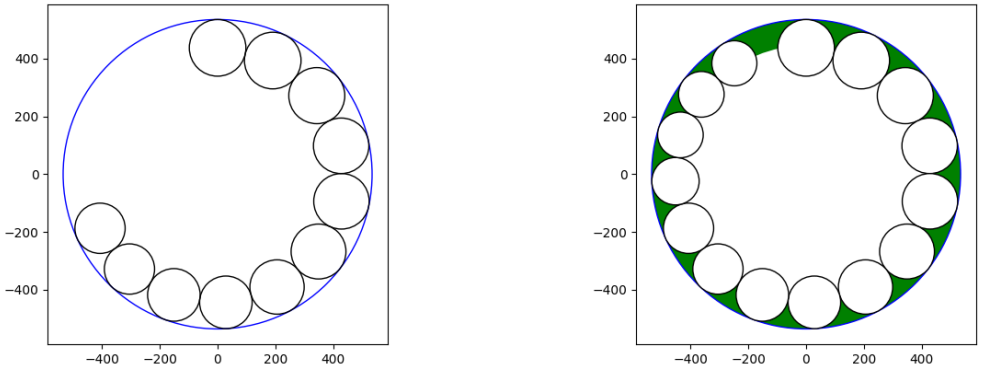
Евристичний алгоритм щільної упаковки кругів, розроблений Б. Задорожним для участі в конкурсі, полягає у наступному. Нехай обрано такий радіус зовнішнього круга, в який можна вмістити всі внутрішні круги. Нехай також є певний набір правил, за яким здійснюється розміщення внутрішніх кругів у зовнішньому крузі, що не залежить від того який радіус зовнішнього круга ми вибираємо. При поступовому зменшенні радіуса, у певний момент буде неможливо задовольнити умови задачі. Відповідно, найменший радіус, що задовольняє умовам, і буде найкращим серед тих, що можна знайти за допомогою цього алгоритму. Отримати певний радіус за мінімальну кількість ітерацій можна за допомогою бінарного пошуку за відповіддю на питання «радіус зовнішнього круга задовольняє умовам розміщення кругів чи не задовольняє?».

Нехай є межа мінімального значення радіуса зовнішнього круга (ліва межа), який є максимальним серед набору радіусів внутрішніх кругів, та межа максимального значення (права межа). Праву межу можна встановити як певне велике значення (наприклад, як суму всіх радіусів внутрішніх кругів). Відповідно, якщо при радіусі зовнішнього круга, що є середнім арифметичним лівої та правої меж, не вдасться розмістити внутрішні круги, то він є меншим за оптимальний. Внаслідок цього ліву межу потрібно збільшити. Аналогічно: якщо вдається розмістити круги, то праву межу потрібно зменшити. Зауважимо, що правила розміщення кругів не можуть змінюватися під час ітерацій, тому що це може впливати на вибір межі (ліва або права), що може призвести до некоректної роботи алгоритму.

Будемо вважати, що нам заданий певний порядок кругів, які потрібно розмістити, і впродовж всього процесу їх порядок змінюватися не буде. Відповідно, якщо використовувати описану процедуру при однакових правилах на наборах з різним порядком елементів, то на деяких з них вдасться розмістити круги, а на інших – не вдасться.

Наступним кроком буде розмістити певний набір внутрішніх кругів впритул до внутрішньої

межі зовнішнього круга: обрати перший круг і розмістити його у координатах $(0; R-r_1)$, де R – радіус зовнішнього круга, r_1 – радіус першого круга з набору. Далі за годинниковою стрілкою розміщувати наступний круг з набору так, щоб він торкався до зовнішнього та попереднього (див. рис. 1, *а*). Якщо ж його не вдається розмістити, то потрібно його пропустити, щоб використати на наступних етапах, і обрати наступний круг. Продовжувати процес, поки не закінчаться внутрішні круги, що не були використані. Варто зазначити, що внутрішній круг неможливо розмістити, якщо він не задовольняє умові неперетину з одним з уже розміщених кругів. При цьому потрібно здійснювати перевірку за всіма уже розміщеними кругами, адже неможливо гарантувати, що вони будуть перетинатися лише з попереднім розміщеним кругом.



а *б*

РИС. 1. *а* – розміщення кругів за годинниковою стрілкою впритул до межі зовнішнього круга; *б* – область, де можна спробувати розмістити внутрішні круги (позначена зеленим)

Після реалізації першого правила розміщення між зовнішньою межею та внутрішніми кругами є область, яку можна використати з метою розміщення в ній малих кругів (див. рис. 1, *б*). Це розміщення можна знайти як розв'язок наступної системи нерівностей:

$$\begin{cases} (x_R - x_m)^2 + (y_R - y_m)^2 \le (R - r_m)^2, \\ (x_p - x_m)^2 + (y_p - y_m)^2 \le (r_p - r_m)^2, \end{cases} \tag{2}$$

де R – радіус зовнішнього круга, $(x_R; y_R)$ – центр зовнішнього круга, r_p – радіус попереднього круга, $(x_p; y_p)$ – центр попереднього круга, r_m – радіус малого круга, $(x_m; y_m)$ – центр малого круга. Якщо вдається розмістити круг, так, щоб задовольнити систему (2) та забезпечити вимоги до розміщення кругів в задачі, тоді ефективно буде використовуватися область між внутрішніми кругами та зовнішньою межею.

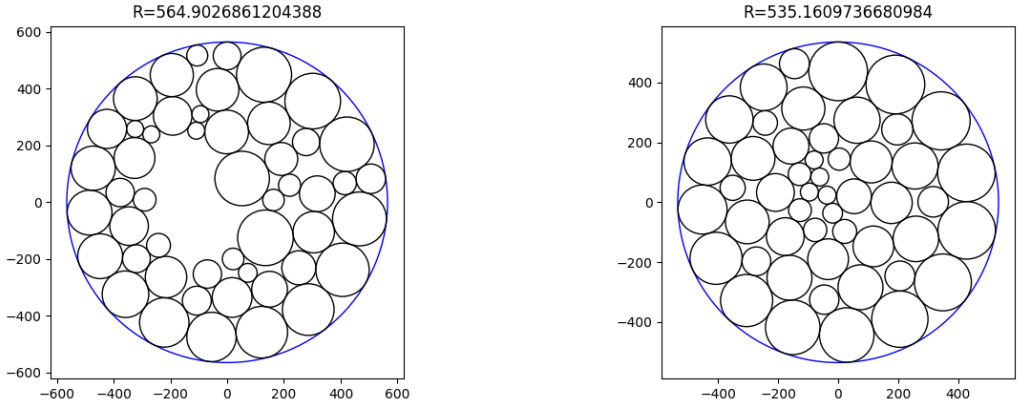
Внутрішні круги можна розміщувати у заглибленнях між кругами з першого етапу (див. рис. 2). Для того, щоб знайти центр дотичного до двох інших внутрішніх кругів, скористаємося системою нерівностей:

$$\begin{cases} (x_1 - x_d)^2 + (y_1 - y_d)^2 \ge (r_1 + r_d)^2, \\ (x_2 - x_d)^2 + (y_2 - y_d)^2 \ge (r_2 + r_d)^2, \end{cases} \tag{3}$$

де r_1 і r_2 – радіуси внутрішніх кругів, що вже розміщені, $(x_1; y_1)$ і $(x_2; y_2)$ – їх відповідні центри, r_d – радіус внутрішнього круга, що дотичний до першого і другого, $(x_d; y_d)$ – центр цього круга. Зауважимо, що нам потрібно у системі (3) мінімізувати відстань від третього круга до двох перших і серед знайдених координат вибрати ті, що найближчі до точки $(0;0)$.

Нерівності (3) можна замінити на рівності і розв'язати систему двох рівнянь, проте точність, з якою будуть знайдені координати кругів, не буде задовольняти умовам розміщення кругів. Це пов'язано з округленням деяких величин у більшу сторону, інших у меншу, в результаті чого координати, що будуть отримані, можуть не пройти перевірку на неперетин кругів. Більш ефективним щодо аналітичного розв'язку показує себе метод Ньютона – Рафсона, який хоч і знаходить координати, при яких третій круг не дотикається до двох попередніх, проте сума відстаней до них дещо більша (рис. 2, *а*).

Більш ефективний спосіб розміщення кругів: йти за годинниковою стрілкою, обирати круг, що не був ще використаний, намагатися розмістити його між двома суміжними кругами. Якщо це не вдається, то намагатися зробити це між кругами, що розміщені через один. Інакше перейти до наступного круга (рис. 2, *б*). Практика показала, що якщо розміщувати круги через 2 і більше кругів, то це не призводить до покращення результатів, тому *r*-розміщення між суміжними кругами та через 1 круг є найбільш ефективним.



а *б*

РИС. 2: *а* – розміщення кругів у заглибленнях між двома суміжними кругами з попереднього ряду;
б – розміщення кругів у заглибленнях суміжних кругів або через один круг

Якщо ж виконувати описаний алгоритм деяку кількість ітерацій, після кожної з яких певну кількість разів змінювати положення двох випадкових кругів (або обирати такі пари, які задовольняють деяким вимогам, наприклад, різниця радіусів не перевищує задане значення) і обираючи найкращий результат з поміж отриманих, то ефективність алгоритму зростає. Проте алгоритм не буде ефективним, якщо різниця між радіусами кругів доволі мала або ж дорівнює нулю, оскільки зміна положення внутрішнього круга у наборі не впливатиме на новий результат, тому що правила, за якими відбувається упаковка, не змінюються.

2. Покращення евристичного алгоритму. З метою покращення результатів, отриманих за допомогою евристичного алгоритму, використовувався *r*-алгоритм Шора [1–3] для знаходження ряду локальних мінімумів у задачі нелінійного програмування, за допомогою якої можна описати математичну модель задачі «Щільна упаковка кругів в круг мінімального радіусу». Для цього ви-

користовувалося формулювання задачі нелінійного програмування зі статті [4], яке має такий вигляд:

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

за обмежень

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

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

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

де R_{low} – максимальний із радіусів для набору кругів.

Для пошуку локального мінімуму в задачі (4)–(7) за допомогою *r*-алгоритму був використаний метод негладких штрафних функцій, для того щоб задачу (4)–(7) можна було замінити на задачу безумовної мінімізації негладкої функції:

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

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

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

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

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

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

Знайти локальний мінімум в задачі (4)–(7) можна за допомогою локального мінімуму безумовної негладкої задачі (8)–(11) для деяких значень коефіцієнтів P_1, P_2 та P_3. Якщо в знайденій точці локального мінімуму функції $f\left(R,x,y\right)$ штрафна функція $\Phi_p\left(R,x,y\right) = 0$, то знайдена точка є локальним мінімумом для задачі (4)–(6). 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, використовувався *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 раціонально буде зменшувати величину кроку для того, щоб можна було врахувати локальні екстремуми, які могли бути пропущені.

Алгоритм з покращенням евристичного розв'язку реалізовано за наступною схемою. Стартуємо з найкращого розміщення кругів, знайдених евристичним алгоритмом. Далі виконуємо наступні три етапи алгоритму з покращення розв'язку.

Етап 1. Встановимо доволі велике значення початкового кроку (визначаємо його в залежності від вхідних даних або ж просто обираємо як великий крок).

Етап 2. Запускаємо *r*-алгоритм зі стартової точки і знаходимо локальний мінімум задачі (4)–(7).

Етап 3. Якщо знайдений локальний мінімум:

- гарантує менший радіус зовнішнього круга, то точка локального мінімуму стає стартовою точкою;
- не гарантує меншого радіуса зовнішнього круга, то величину кроку *h* зменшуємо вдвічі.

Етап 4. Якщо величина кроку більша за задане значення, то переходимо до етапу 2. Інакше закінчуємо роботу алгоритму з покращення розв'язку.

Для реалізації алгоритму з покращенням була написана програма на мові програмування Python, яка реалізувала варіант $r(\alpha)$-алгоритму з адаптивним регулюванням кроку. У її основу була покладена octave-функція **ralgb5a** [5]. Тут абревіатура «**b5**» означає, що коригується $n \times n$-матриця B, а кожна ітерація вимагає $5n^2$ арифметичних операцій множення, які визначають обчислювальну складність однієї ітерації. Програма **ralgb5a** використовує octave-функцію **function [f, g] = calcfg(x)**, яка обчислює значення функції $f = f(x)$ та її узагальненого градієнта $g = g_f(x)$ в точці *x*. Ця функція готується користувачем та може мати довільне ім'я, яке підтримує синтаксис Octave. Параметри $r(\alpha)$-алгоритму вибиралися наступними: $\alpha \in [2,4]$, $q_1 = 1.0$. Використовувалися такі параметри зупинки: $\varepsilon_x \approx 10^{-6}$, $\text{maxitn} = 3000$.

3. Аналіз отриманих результатів. Евристичний алгоритм та його покращений варіант реалізовані мовою програмування Python 3.9.5 [6] з використанням бібліотеки NumPy 1.24.2 [7]. За допомогою розроблених програмних реалізацій проведено обчислювальні експерименти для усіх 50 тестових задач конкурсу «Щільна упаковка кругів в круг мінімального радіусу», по десять тестів для $N \in \{10, 20, 30, 40, 50\}$.

Для 50 конкурсних тестів та різної кількості ітерацій евристичного алгоритму: 10, 20, 30, 40 та 50 ітерацій, у табл. 1 наведено оцінки у балах, які розраховувалися за формулою (1). Якщо алгоритм з уточненням покращує результат евристичного алгоритму так, що збільшується кількість балів, то у відповідній клітинці через «/» записані кількості балів, отримані евристичним алгоритмом та алгоритмом з уточненням. Жирним шрифтом позначені покращення на 3 і більше балів.

Із табл. 1 видно, що для всіх конкурсних тестів кількість балів монотонно збільшується у залежності від кількості ітерацій евристичного алгоритму. Для більшої частини конкурсних тестів алгоритм з уточненням дозволяє отримати кращі результати в балах – у більшості випадків покращення становлять до 2 балів, за вийнятком виділених жирним шрифтом покращень від 3 до 6 балів. Найбільші покращення помітні у тестах 22 та 32: 6 балів. Це пов'язано з тим, що круги у цих тестах мають однаковий радіус, де евристичний алгоритм показує себе не так ефективно. Тому для таких тестів завдяки алгоритму з уточненням можна значно покращити результати конкурсних задач в балах для конкурсних задач.

Для 50 конкурсних тестів на рис. 3 показано діаграму для сумарної кількості балів при різній кількості ітерацій евристичного алгоритму: 10, 20, 30, 40 та 50 ітерацій.

Для евристичного алгоритму з рис. 3 видно, що при збільшенні кількості ітерацій кількість балів зростає, це зростання поступово зменшується з кількістю ітерацій. Наприклад, якщо при 10 ітераціях кількість отриманих балів дорівнює 4833, то при 50 – 4851, тобто при збільшенні кількості ітерацій у 5 разів покращення становить лише 18 балів. Враховуючи, що максимум по балах є 5000, то це лише 0,36 %. Проте кращі результати показує алгоритм з уточненням. Так, для прикладу, при 10 ітераціях евристичний алгоритм дає 4833, а після уточнення – 4901, тобто

покращення складає 68 балів, що дорівнює 1,36 %. При збільшенні кількості ітерацій відсоток покращень алгоритму з уточненням буде зменшуватись, тому що евристичний алгоритм буде знаходити кращі результати.

ТАБЛИЦЯ 1. Оцінки в балах для конкурсних тестів: евристичний алгоритм/алгоритм з уточненням

№	Кількість ітерацій евристичного алгоритму					№	Кількість ітерацій евристичного алгоритму				
	10	20	30	40	50		10	20	30	40	50
1	98/99	98/99	98/99	98/99	98/99	26	97	97	97	97/98	97/98
2	96/98	96/98	98	98	100	27	97/98	97/98	97/98	97/98	97/98
3	97/99	98	98	98	98	28	96/98	96/97	96/97	96/97	96/97
4	99	99	99	99	99	29	98	98	98	98	98
5	99	99	99	99	99/100	30	97/98	97/98	97/98	97/98	97/98
6	98	99	99	99	99	31	**94/98**	**94/98**	**94/98**	**94/98**	**94/98**
7	97/98	98	98	98	98	32	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**
8	97	98	98/99	98/99	98/99	33	96/97	96/97	96/97	96/97	96/97
9	98	98	98/99	98/99	98/99	34	98	98	98	98	98
10	**96/99**	98/100	98/100	98	98	35	96/98	96/98	96/98	96/98	96/98
11	98/99	98/99	98/99	98/99	98/99	36	96/97	96/97	96/97	96/97	96/97
12	96/98	96/97	96/97	97	97	37	97/98	97/98	97/98	97/98	97/98
13	97/98	97/98	97/98	97/98	97/98	38	97	97	97	97	97
14	96/98	96/98	97/99	97/98	97/98	39	97/98	97/98	97/98	97/98	97/98
15	96/97	96/97	96/97	96/97	96/98	40	**95/98**	96/98	96/98	97/98	97/98
16	97	97	97	97	97	41	**96/99**	**96/99**	**96/99**	**96/99**	**96/99**
17	96/97	96/97	97/99	97/99	97/99	42	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**
18	96/97	96/97	96/97	96/97	96/97	43	97/98	97/98	97/98	97/98	97/98
19	96/97	97/98	97/98	97/98	97	44	97/98	97/98	97/98	97/98	97/98
20	96/98	97	97	97	97	45	96/98	96/98	96/98	96/98	96/98
21	**95/99**	**96/99**	**96/99**	**96/99**	**96/99**	46	98	98	98	98	98
22	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**	47	97/98	97/98	97/98	97/98	97/98
23	96/97	96/97	96/97	96/97	96/97	48	98	98	98	98	98
24	96/97	96/98	96/98	96/98	96/98	49	98/99	98/99	98/99	98/99	98/99
25	97	97/98	97/98	97/98	97/98	50	98	98	98	98	98

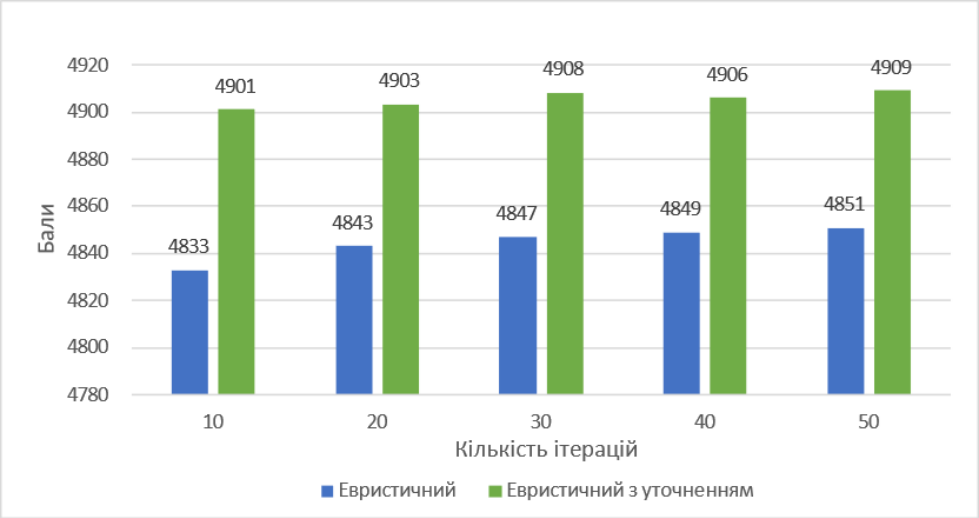


РИС. 3. Сумарна кількість балів за всіма тестами

Важливим аспектом для обох алгоритмів є час роботи їх програмних реалізацій для отримання результатів як для кожного окремого конкурсного тесту та для всіх 50 тестів загалом. У табл. 2 наведено час роботи в секундах для програмних реалізацій евристичного алгоритму та алгоритму з уточненням, де жирним шрифтом виділено часи виконання для кожних десяти тестів з кількостями кругів $N \in \{10, 20, 30, 40, 50\}$. У двох останніх рядках таблиці для кожного з алгоритмів наведено загальний час виконання усіх 50 конкурсних тестів.

ТАБЛИЦЯ 2. Затрати в секундах на виконання конкурсних тестів (I – евристичний алгоритм, II –алгоритм з уточненням)

№	Кількість ітерацій евристичного алгоритму									
	10		20		30		40		50	
	I	II	I	II	I	II	I	II	I	II
1	0,94	4,95	1,99	5,98	3,03	7,33	4,16	7,18	5,19	8,15
2	0,94	3,33	1,9	6,81	2,79	7,5	3,76	8,68	4,76	9,21
3	**0,89**	**3,3**	**1,76**	4,56	**2,58**	**4,3**	**3,56**	**5,32**	**4,41**	**6,15**
4	1,02	6,12	2,06	7,18	3,07	8,06	4,08	9,25	5,21	10,32
5	0,97	3,25	1,99	**4,28**	2,92	5,17	3,94	6,25	4,92	7,84
6	1,08	5,59	2,27	8,77	3,49	9,83	4,63	11,15	5,84	12,28
7	1,04	5,46	2,12	6,28	3,14	7,22	4,2	8,44	5,3	9,43
8	1,14	5,84	2,33	5,69	3,52	7,68	5,25	9,54	6,01	10,22
9	1,13	5,2	2,3	6,42	3,44	7,93	4,63	9,26	5,86	10,44
10	1,25	3,58	2,52	5,26	3,74	6,37	5,02	8,83	6,34	10,14
11	4,9	13,04	9,53	17,89	14,3	22,29	18,8	27,13	24,27	32,46
12	4,38	15,39	9,26	17,34	13,7	21,51	18,35	29,67	22,52	33,46
13	4,73	14,01	9,42	17,53	13,7	21,49	20,6	28,91	24,08	32,08
14	4,29	15,38	8,87	20,14	13,3	19,82	17,6	31,5	22,35	36,17
15	4,3	14,24	9,15	19,08	14	23,75	18,85	28,92	22,97	29,32
16	4,21	18,55	8,86	23,41	13,4	27,61	18,27	33,03	22,2	36,59

Закінчення таблиці 2

№	Кількість ітерацій евристичного алгоритму									
	10		20		30		40		50	
	I	II	I	II	I	II	I	II	I	II
17	4,14	14,04	8,49	18,61	12,9	25,87	17,45	30,86	21,95	35,15
18	4,45	13,33	8,82	17,85	13,4	22,16	17,64	31,02	22,61	35,86
19	**3,78**	**8,7**	8,1	19,4	12,6	21,01	16,88	25,63	21,89	28,41
20	3,89	14,38	**8,05**	**13,06**	**12,1**	**17,01**	**16,31**	**21,37**	**20,71**	**25,7**
21	11	21,36	23,2	39,31	35	50,8	46,63	59,14	59,26	71,66
22	11	24,38	22,3	35,68	33,4	46,46	44,03	57,49	55,16	68,59
23	**8,79**	25,59	19,3	36,11	**29**	**40,15**	40,88	52,52	49,39	**60,74**
24	9,05	22,02	19,3	36,24	29,3	45,69	39,5	56,5	**49,23**	66,09
25	9,25	21,37	**19,2**	33,34	29,6	43,34	**39,05**	53,28	49,34	63,41
26	9,26	**21,16**	19,6	**31,61**	29,8	41,5	40,43	**51,62**	51,31	62,28
27	10,2	22,7	21,3	33,88	33,2	45,43	44,55	57,27	56,01	68,46
28	9,85	22,17	20,8	32,95	32,6	44,34	42,13	54,36	53,74	65,79
29	10,3	23,11	21,1	33,93	31,4	43,95	41,49	54,44	52,05	64,82
30	10,1	23,72	21,1	34,73	31,8	45,13	43,19	57,05	54,54	68,14
31	20,5	39,85	42,9	61,84	65,5	84,03	87,7	107	110,1	130,3
32	23,5	42,61	47,5	65,64	71	89,02	93,59	111,9	118,5	138
33	17,1	33,65	**35,7**	**51,82**	**54,6**	75,25	**72,14**	**88,79**	94,4	**111**
34	17,5	36,32	37,1	55,59	56,3	74,36	74,06	93,93	**94,16**	113,8
35	**17**	33,83	38,1	54,81	57,3	**73,23**	76,58	93,23	96,27	113,6
36	18,5	33,55	39	53,8	59,8	74,39	77,54	93,03	97,08	112,7
37	18,4	35	40,6	57,14	62,7	78,79	84,24	101	108,1	125,3
38	18	**33,12**	39,5	54,3	61,3	75,89	80,32	95,2	104,3	119,8
39	18,8	33,13	39,3	53,48	60,2	73,96	78,55	92,98	103,2	117,7
40	17,3	39,03	39,5	57,81	59,7	77,68	79,99	97,53	102,6	121,8
41	29,4	77,55	66,4	109,5	102	129,8	142,2	171,9	174,7	204,1
42	38,4	66,46	77,2	104,4	115	142,1	155,9	185	192,5	221
43	25,7	56,46	**55,6**	85,62	87	116,6	125,4	157,6	157,7	188,3
44	**25,4**	47,66	56,2	**78,4**	**83,4**	**105,4**	**117,6**	**140,9**	**150,2**	**172,9**
45	26,8	56,82	59,5	89,49	92,6	122,4	128	159	156,4	186,9
46	26	**46,46**	59,4	79,8	91,8	111,9	125,6	147,5	155,2	176,2
47	25,8	48,19	61,5	84,14	98,3	120,3	137,4	161	175,2	198,6
48	27,6	53,34	63,2	88,87	100	125,9	137,4	164	174	200,7
49	29,1	55,37	63,1	89,29	98,4	124,4	137	164,2	170,9	198,2
50	24,5	54,63	60,9	90,97	95,9	125,5	130,9	161,1	171,6	203
Загальний час виконання усіх 50 тестів у секундах										
I	617		1339		2053		2788		3517	
II	1338		2060		2741		3512		4233	

З табл. 2 видно, що алгоритм з уточненням працює майже той самий час, що і евристичний алгоритм, для якого кількість ітерацій на 10 більша. Якщо ж не задаватися знаходженням значення якомога найменшого радіуса зовнішнього круга, то ефективним буде використання алгоритму з уточненням знайденого розміщення кругів за 10 ітерацій евристичного алгоритму. Наприклад, для усіх тестів при 10 ітераціях загальна кількість балів становить 4901, для 50 ітерацій – 4909, що лише на 0,16 % більше, проте потребує в 3,16 разів більше часу, ніж для 10 ітерацій.

Важливо зазначити, що результати, що наведені у табл. 2 були обчислені використовуючи мову програмування Python. Ці ж результати можна отримати за менший час, якщо використати більш швидкі за часом мови програмування: C++, Fortran, Rust та інші. Так наприклад, для 49 та 50 конкурсних тестів реалізація алгоритмів на мові програмування Rust показала прискорення в 50 разів у порівнянні з програмою на Python.

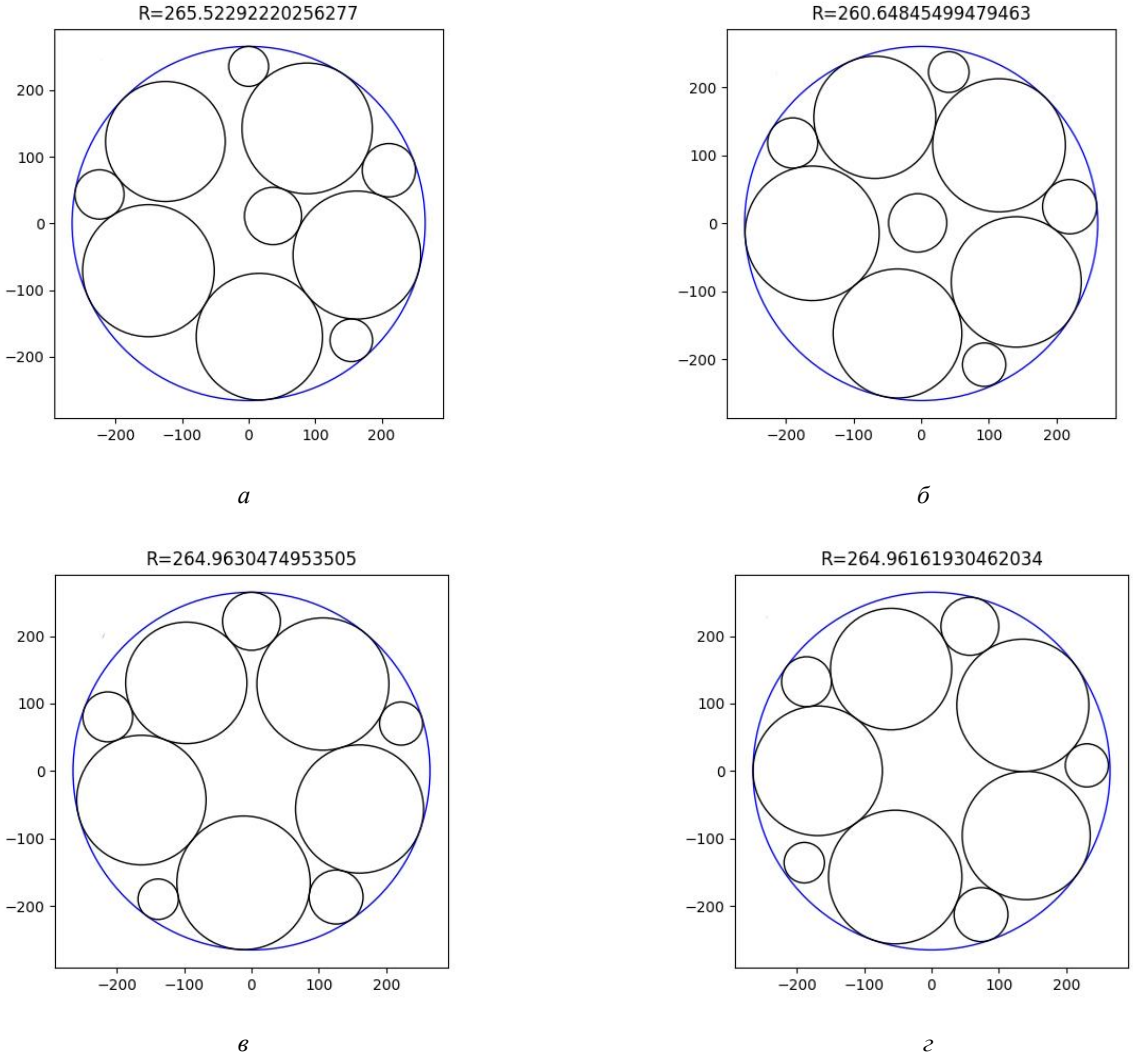
Важлива характеристика евристичного алгоритму – це кількість обмінів кругів на одній ітерації. У табл. 3 для частини конкурсних тестів наведена залежність від цієї характеристики для кількості балів, отриманих за допомогою евристичного алгоритму та алгоритму з уточненням. Тут наведено результати для 10 тестів, на яких спостерігалися найбільші покращення. У табл. 3 жирним шрифтом виділено покращення більше 2 балів. В останніх двох рядках таблиці наведено сумарні покращення в балах для усіх 50 тестів, де найбільша різниця в балах для евристичного алгоритму та для алгоритму з покращенням виділена жирним шрифтом.

ТАБЛИЦЯ 3. Кількість балів в залежності від кількості обмінів кругів на одній ітерації (I – евристичний алгоритм, II – алгоритм з уточненням)

№	Кількість обмінів									
	1	2	3	4	5	6	7	8	9	10
1	98/100	99/100	99	98/99	99	99/100	98/99	99	99/100	98/99
10	98/99	97	99	97/99	98/100	99	98/100	99/100	98/99	98
19	97/98	98	97/99	97/98	98	98	98	98/99	97/98	97
20	97	97	97/98	97/98	96/98	97	97/98	97	96/97	97
21	**95/99**	**96/99**	**96/99**	**96/99**	**96/99**	**96/99**	**96/99**	**96/100**	**95/99**	**96/99**
22	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**	**94/100**
31	**95/99**	**95/99**	**94/99**	**95/99**	**95/99**	**95/99**	**94/99**	**94/99**	**94/99**	**94/98**
32	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**	**93/99**
41	**96/99**	**96/99**	**96/99**	**96/99**	96/98	**96/99**	**96/99**	97/99	97/98	**96/99**
42	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**	**97/100**
Сума балів усіх 50 тестів										
I	**4862**	**4861**	4856	4852	4854	4859	4853	4854	4850	4851
II	**4909**	4905	**4910**	4905	4908	**4911**	4908	4908	4908	**4909**

З табл. 3 видно, що різниця для суми балів по усіх 50 тестах при використанні евристичного алгоритму та алгоритму з уточненням є доволі малою. Найкращі результати отримані для тестів, у яких радіуси кругів рівні, або різниця між ними є доволі малою. Підсумовуючи вищенаведене, можна зробити висновок, що для евристичного алгоритму збільшення кількості обмінів місцями у наборі кругів за одну ітерацію не приводить до суттєвих покращень.

Для алгоритму з уточненням часто вигідніше вибирати стартову точку з більшим радіусом зовнішнього круга, який отриманий при меншій кількості ітерацій евристичного алгоритму. Так, наприклад, для 10 конкурсного тесту на рис. 4, *б* показано найбільше покращення, а на рис. 4, *г* показано найменше покращення, які отримані за допомогою *r*-алгоритму для розміщень кругів з рис. 4, *а* та *в*, знайдених евристичним алгоритмом за 30 та 40 ітерацій, відповідно.



а *б* *в* *г*

РИС. 4. Розміщення кругів для 10-ого конкурсного тесту: *а* – за 30 ітерацій евристичного алгоритму; *б* – знайдене покращення при 30 ітераціях евристичного алгоритму; *в* – за 40 ітерацій евристичного алгоритму; *г* –знайдене покращення при 40 ітераціях евристичного алгоритму

Висновки. Стаття присвячена дослідженню евристичного алгоритму для розв’язання конкурсної задачі «Щільна упаковка кругів в круг мінімального радіусу» та розробці його покращеного варіанту за допомогою *r*-алгоритму Шора з дихотомією кроку. 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

раціонально буде зменшувати величину кроку для того, щоб можна було врахувати локальні екстремуми, які могли бути пропущені.

Алгоритм для покращення евристичного розв'язку реалізовано за наступною схемою. Стартуємо з найкращого розміщення кругів, знайденого евристичним алгоритмом. Далі виконуємо наступні етапи алгоритму.

Етап 1. Встановимо доволі велике значення початкового кроку (визначаємо його в залежності від вхідних даних або ж просто обираємо як великий крок).

Етап 2. Запускаємо *r*-алгоритм зі стартової точки і знаходимо локальний мінімум задачі (4)–(7).

Етап 3. Якщо знайдений локальний мінімум: гарантує менший радіус зовнішнього круга, то точка локального мінімуму стає стартовою точкою; якщо ж знайдений локальний мінімум не гарантує меншого радіуса зовнішнього круга, то величину кроку h зменшуємо вдвічі.

Етап 4. Якщо величина кроку більша за задане значення, то переходимо до етапу 2. Інакше закінчуємо роботу алгоритму з покращення розв'язку.

Для програмної реалізації обох алгоритмів використовувались мова програмування Python 3.9.5 та бібліотека NumPy 1.24.2. Для 50 конкурсних тестів та різної кількості ітерацій евристичного алгоритму (10, 20, 30, 40 та 50 ітерацій) проведено аналіз отриманих результатів за трьома параметрами: кількість отриманих балів, час виконання програми та кількість обмінів кругів на одній ітерації евристичного алгоритму. Для більшої частини конкурсних тестів алгоритм з уточненням дозволяє отримати кращі результати в балах, так у більшості випадків покращення становлять до 2 балів, а у деяких випадках – від 3 до 6 балів. Найбільші покращення (6 балів) помітні у тестах, де круги мають однакові або близькі радіуси. Тут евристичний алгоритм показує себе не так ефективно, тому для таких тестів завдяки алгоритму з уточненням можна значно покращити результати конкурсних задач в балах.

Подяка. Роботу підтримано грантом Volkswagen Foundation № 97775. Автори висловлюють подяку Ользі Хом'як (Інститут кібернетики імені В.М. Глушкова НАН України) за допомогу в підготовці рукопису статті.

Список літератури

1. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наукова думка, 1979. 200 с.
2. Shor N.Z. Non-Differentiable Optimization and Polynomial Problems. Kluwer Academic Publishers, Boston, Dordrecht, London. 1998. 412 p.
3. Стецюк П.И. Теория и программные реализации *r*-алгоритмов Шора. *Кибернетика и системный анализ*. 2017. № 5. С. 43–57.
4. Stetsyuk P., Romanova T., Scheithauer G. On the global minimum in a balanced circular packing problem. *Optimization Letters*. 2016. No. 10. P. 1347–1360.
5. Стецюк П.І., Пилиповський О.В., Хом'як О.М. GNU Octave та Python реалізації *r*-алгоритму Шора з адаптивним регулюванням кроку. *Cybernetics and Computer Technologies*. 2022. 3. С. 98–112.
6. Python 3.9.5:https://www.python.org/downloads/release/python-395 (звернення: 09.08.2023)
7. NumPy 1.24.2 Release Notes https://numpy.org/devdocs/release/1.24.2-notes.html (звернення: 09.08.2023)

Одержано 17.07.2023

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

Міца Олександр Володимирович,
доктор технічних наук, завідувач кафедри інформаційних управляючих систем та технологій
Ужгородського національного університету, Ужгород,
alex.mitsa@gmail.com
https://orcid.org/0000-0002-6958-0870

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

УДК 519.85

Б.О. Задорожний [1 *], О.В. Міца [1], П.І. Стецюк [2]

Про покращення евристичного алгоритму упаковки кругів в круг мінімального радіусу

[1] *Ужгородський національний університет, Ужгород*
[2] *Інститут кібернетики імені В.М. Глушкова НАН України, Київ*
[*] *Листування: bohzador@gmail.com*

Стаття присвячена дослідженню евристичного алгоритму для розв'язання конкурсної задачі «Щільна упаковка кругів в круг мінімального радіусу» та розробці його покращеного варіанту за допомогою *r*-алгоритму Шора з дихотомією кроку. Евристичний алгоритм розроблений студентом третього курсу Ужгородського національного університету Богданом Задорожним. Реалізована на його основі програма посіла друге місце у змаганні та використовувала менше часу за максимальний час, який за умовою змагання відводився на одноразовий запуск програми для 50 конкурсних задач. У статті описано результати дослідження як за допомогою *r*-алгоритму Шора можна покращити ефективність роботи евристичного алгоритму.

Алгоритм для покращення евристичного розв'язку реалізовано за наступною схемою. Стартуємо з найкращого розміщення кругів, знайденого евристичним алгоритмом. Далі виконуємо наступні чотири етапи алгоритму. Етап 1. Встановимо доволі велике значення початкового кроку (визначаємо його в залежності від вхідних даних або ж просто обираємо як великий крок). Етап 2. Запускаємо *r*-алгоритм зі стартової точки і знаходимо локальний мінімум відповідної задачі нелінійного програмування. Етап 3. Якщо знайдений локальний мінімум: гарантує менший радіус зовнішнього круга, то точка локального мінімуму стає стартовою точкою; якщо ж знайдений локальний мінімум не гарантує меншого радіуса зовнішнього круга, то величину кроку h зменшуємо вдвічі. Етап 4. Якщо величина кроку більша за задане значення, то переходимо до Етапу 2. Інакше закінчуємо роботу алгоритму з покращення розв'язку.

Для програмної реалізації обох алгоритмів використовувались мова програмування Python 3.9.5 та бібліотека NumPy 1.24.2. Для 50 конкурсних тестів та різної кількості ітерацій евристичного алгоритму (10, 20, 30, 40 та 50 ітерацій) проведено аналіз отриманих результатів за трьома параметрами: кількість отриманих балів, час виконання програми та кількість обмінів кругів на одній ітерації евристичного алгоритму. Для більшої частини конкурсних тестів алгоритм з уточненням дозволяє отримати кращі результати в балах, так у більшості випадків покращення становлять до 2 балів, а у деяких випадках – від 3 до 6 балів. Найбільші покращення (6 балів) помітні у тестах, де круги мають однакові або близькі радіуси. Тут евристичний алгоритм показує себе не так ефективно, тому для таких тестів завдяки алгоритму з уточненням можна значно покращити результати конкурсних задач в балах.

Ключові слова: пакування кругів, евристика, *r*-алгоритм, Python 3.9.5, NumPy 1.24.2.

UDC 519.85

Bohdan Zadorozhnyi [1 *], Oleksandr Mitsa [1], Petro Stetsyuk [2]

On the Improvement of the Heuristic Algorithm for Packing Circles into a Circle of Minimum Radius

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

The article is devoted to investigation of a heuristic algorithm for solving the competitive problem "Circles Dense packing into a circle of minimum radius" and development of its improved version using Shor's *r*-algorithm with step dichotomy. The heuristic algorithm was developed by Bohdan Zadorozhnyi, a third-year student of Uzhhorod National University. The program implemented on its basis took second place in the competition and used less time than the maximum time, which, according to the competition, was allocated for a one-time launch of the program for 50 competitive tasks. The article describes the research results of how the Shor *r*-algorithm can be used to improve efficiency of the heuristic algorithm.

The algorithm for improving the heuristic solution is implemented according to the following scheme. We start from the best arrangement of circles found by the heuristic algorithm. Then, we perform the following four stages of the algorithm: Stage 1. We set a rather large value of the initial step (we determine it depending on the input data or simply choose it as a large step); Stage 2. We run the *r*-algorithm from the starting point and find the local minimum of the corresponding nonlinear programming problem; Stage 3. If the local minimum found guarantees a smaller radius of the outer circle, then the local minimum point becomes the starting point; the local minimum found does not guarantee a smaller radius of the outer circle, then the size of the step h is reduced by half. Stage 4. If the step value is greater than the specified value, then we proceed to Stage 2. Otherwise, we finish work of the algorithm with solution improvement.

Python 3.9.5 programming language and NumPy 1.24.2 library were used for software implementation of both algorithms. For 50 competitive tests and different number of iterations of the heuristic algorithm (10, 20, 30, 40 and 50 iterations), the analysis of the obtained results was conducted according to three parameters: the number of points obtained, program execution time and the number of exchanges of circles in one iteration of the heuristic algorithm. For most of the competitive tests, the refined algorithm allows you to get better results in points, in particular, in most cases the improvement is up to 2 points, and in some cases – from 3 to 6 points. The biggest improvements (6 points) are seen in tests, where the circles have the same or close radii. Here, the heuristic algorithm works not so effectively, therefore, for such tests, due to the algorithm with refinement, it is possible to significantly improve the results of competitive tasks in terms of points.

Keywords: circle packing, heuristics, *r*-algorithm, Python 3.9.5, NumPy 1.24.2.