УДК 004.023

М.М. Глибовець, Н.М. Гулаєва, І.О. Морозов

АНАЛІЗ ГЕНЕТИЧНИХ АЛГОРИТМІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ ДВОВИМІРНОЇ ОРТОГОНАЛЬНОЇ УПАКОВКИ ПРЯМОКУТНИХ ОБ'ЄКТІВ У НАПІВНЕСКІНЧЕННУ СМУГУ

Досліджено клас генетичних алгоритмів вирішення задачі двовимірної ортогональної упаковки прямокутних об'єктів у напівнескінченну смугу фіксованої ширини. Наведено результати теоретичного аналізу складності реалізації декодерів MERA та BLF; запропоновані власні реалізації цих декодерів з низкою евристичних оптимізацій. Запропоновано реалізацію генетичного алгоритму розв'язання задачі упаковки для окремих випадків (із забороною поворотів об'єктів та з поворотами на 90°). Описано результати тестових випробувань розробленого алгоритму за різних конфігурацій основних параметрів з використанням загальновідомих тестових наборів. Наведено результати порівняння отриманого алгоритму з іншими відомими алгоритмами.

Ключові слова: генетичний алгоритм, задача упаковки-розкрою, декодер, Packing problem, MERA, BLF.

Вступ

Розв'язання задачі упаковки-розкрою (Packing problem) полягає у знаходженні оптимального розташування множини предметів меншого розміру (деталей) на об'єктах більшого розміру (заготовках). Цільовою функцією задачі зазвичай є мінімізація втрат заготовок: необхідно розмістити предмети або як можна щільніше, або в як можна меншу кількість контейнерів. Загальновідоме широке практичне застосування задачі.

В цій роботі розглянуто задачу двовимірної ортогональної упаковки прямокутних об'єктів у напівнескінченну смугу (ДОУПОНС) з окремими випадками (забороною поворотів об'єктів та з поворотами на 90°). Тобто, на смузі фіксованої ширини потрібно так розмістити прямокутники, щоб вони не перетинались, не виходили за межі смуги, а довжина використаної частини смуги була мінімальною. Дослідження розв'язків цієї задачі має давню історію [1–3] і привели до її занесення до класу NP-складних задач комбінаторної оптимізації [4].

Тому широке розповсюдження для її розв'язання дістали евристичні та метаевристичні методи. Серед метаевристичних методів виділяють генетичні алгоритми [5–7], дослідженню яких присвячено цю роботу.

Метою роботи є розробка та аналіз генетичного алгоритму (ГА) розв'язання задачі ДОУПОНС для випадків з забороною повороту прямокутників та з дозволеним поворотом на 90°. Особлива увага приділялася теоретичному аналізу складності реалізації декодерів MERA та BLF. Для аналізу роботи ГА використовувався експериментальний підхід (проведення численних прогонів ГА на загальновідомих тестових даних). У статті ми опишемо аналіз розробленого ГА з використанням різних наборів параметрів (методи відбору, оператор кросинговеру, ймовірність мутації, використаний декодер) та виділимо набори параметрів, які дають найкращі результати для різних класів задач, і сформулюємо рекомендації щодо використання певних конфігурацій параметрів ГА.

Формальна постановка задачі

Задача ДОУПОНС фіксованої ширини (*two dimensional orthogonal rectangular strip packing problem*) формулюється так. Вхідну інформацію задають набором даних (W, m, w, l), де $W \in N$ – ширина напівнескінченої смуги; $m \in N$ – кількість прямокутних об'єктів; $(w_1, w_2, \ldots, w_i, \ldots, w_m)$, $w_i \in N$ – ширина i-го прямокутника; $(l_1, l_2, \ldots l_i, \ldots, l_m)$,

$l_i \in N$ – довжина *i*-го прямокутника. Введено прямокутну систему координат *XOY*, таку що осі *OX* та *OY* збігаються відповідно з нижньою необмеженою та боковою сторонами смуги. Позиція *i*-го прямокутника називається *горизонтальною*, якщо його сторона довжини l_i є паралельною до необмеженої сторони смуги, а друга сторона є перпендикулярною до неї. Задамо горизонтальну позицію *i*-го прямокутника парою координат його нижнього лівого кута (x_i, y_i). Набір векторів (x_i, y_i), i=1,2,3,…,m, називається *допустимою прямокутною упаковкою*, якщо виконуються такі умови:

- сторони прямокутників є паралельними до сторін смуги (умова ортогональності);
- прямокутники не перетинаються між собою:

$$\forall i, j = 1,..., m,\ i \neq j:$$

$$((x_i \geq x_j + l_j) \vee (x_j \geq x_i + l_i)) \wedge$$

$$\wedge ((y_i \geq y_j + w_j) \vee (y_j \geq y_i + w_i));$$

- прямокутники не виходять за межі смуги:

$$\forall i = 1,..., m:$$

$$(x_i \geq 0) \wedge (y_i \geq 0) \wedge ((y_i + w_i) \leq W);$$

- дозволені або заборонені повороти прямокутників на кути, кратні 90°.

Прямокутник розташований *щільно*, якщо він дотикається хоча б нижньою та лівою сторонами до інших прямокутників упаковки або до країв смуги. Упаковку називають *щільною*, якщо всі її прямокутники розташовані щільно.

Допустима прямокутна упаковка називається *оптимальною*, якщо довжина зайнятої упакованими прямокутниками частини смуги сягає мінімуму (при цьому втрати невикористаних частин смуги є мінімальними). Нескладно бачити, що оптимальна упаковка є щільною.

Діркою в упаковці називається незаповнена ділянка смуги, з усіх боків оточена прямокутниками або краями смуги.

Упаковка називається *ідеальною*, якщо використана ділянка смуги є прямокутником без дірок з шириною, що дорівнює ширині смуги.

Розв'язком задачі двовимірної ортогональної упаковки прямокутних об'єктів у напівнескінчену смугу є оптимальна упаковка.

Найпоширеніші методи розв'язку

Методи розв'язку задачі двовимірної ортогональної упаковки діляться на точні, евристичні та метаевристичні [2].

Точні методи розв'язку засновані на методі гілок та границь. Запропонований у [8] алгоритм є ні чим іншим, як перебором порядку «приклеювання» прямокутників один до одного, що відсікає гілку тоді, коли не можна приставити новий прямокутник до старих без утворення «дірок». При цьому алгоритм спирається на припущення про існування ідеальної упаковки. Цей метод довів свою ефективність на задачах розміром до 30 об'єктів. Інші точні методи цей результат не покращують, доводячи необхідність евристичних методів.

Перша вдала евристика «Нижній лівий» (Bottom Left) датується 1980 роком [9]. Метод полягав у впорядкуванні прямокутників за спаданням площі. Потім об'єкти розташовувались у верхньому правому куті та зсувались вниз та вліво, наскільки це дозволяли вже встановлені об'єкти. Цей метод був покращений у 1983 р. [10] та названий «Нижній лівий із заповненням» (Bottom Left Fill): кожен об'єкт розташовується в найлівішому та найнижчому можливому місці. Тривалий час ці методи були кращими і давали прийнятні результати. Але в XXI сторіччі інтерес до задачі виріс, а з ним – і вимоги до розв'язків. Тому популярність набрали ймовірнісно-жадібні методи. Вони теж спочатку сортують об'єкти за певним критерієм, але розміщують новий прямокутник тільки з певною ймовірністю. Після одного прогону невстановлені об'єкти знову проходять цю процедуру, і так поки

жодного не залишиться. Прикладом такого алгоритму є GRASP [11].

Описані методи є так званими «евристиками нижнього рівня»; згодом над ними були побудовані метаевристичні алгоритми. Наприклад, BLF-метод був комбінований з генетичним алгоритмом та з методом імітації відпалу; відповідні методи отримали назву GA+BLF та SA+BLF, відповідно.

Кодування розв'язків

Генетичні алгоритми оперують хромосомами – закодованими розв'язками поставленої задачі. Спосіб кодування розв'язків – фундаментальна підзадача алгоритму; огляд способів кодування наведено в [2]. Для даної задачі найочевидніший спосіб кодування упаковки – *прямий код*, або *кодування на основі об'єктів*: у хромосомі закодовано позиції прямокутників (наприклад, координати лівих нижніх кутів на смузі). Такий спосіб має свої переваги: простота реалізації та декодування. Але критичний недолік полягає у тому, що дуже багато отриманих упаковок є недопустимими або нещільними, причому складність виправлення «поганих» упаковок є квадратичною від кількості прямокутників. Тому цей спосіб кодування для задачі двовимірної ортогональної упаковки рідко використовується на практиці, а найпоширенішим є *кодування перестановками прямокутників*. В цьому випадку хромосома містить послідовність номерів прямокутників, яка представляє порядок їх укладки на смугу. Правила укладки визначаються спеціальним алгоритмом – декодером, причому декодери розробляються в такий спосіб, щоб побудовані ними упаковки були допустимими та щільними. Оскільки одну перестановку можна розшифрувати багатьма способами у допустимі щільні упаковки (результуючі упаковки матимуть різну довжину або вимагатимуть різної кількості операцій для побудови), декодери можна порівнювати між собою.

Як правило, в генетичних алгоритмах використовуються однопрохідні онлайн декодери [2]. Найпоширеніші з цих декодерів – «Нижній лівий» (Bottom Left – BL), «Удосконалений нижній лівий» (Improved Bottom Left – IBL), «Нижній лівий із заповненням» (Bottom Left Fill – BLF), «Мінімізація площі оточуючого прямокутника» (Minimization of Enclosing Rectangle Area – MERA) – генерують *BL-компактні* упаковки. Такі упаковки задовольняють *BL-умові*: жоден розташований на смузі прямокутник не може бути зсунутий далі вниз або вліво.

Далі проводиться порівняльний аналіз роботи двох декодерів – MERA та BLF – в контексті генетичного алгоритму. Складність реалізації обох декодерів є кубічною; якість отриманих розв'язків аналізується експериментально.

Декодер «Мінімізація площі оточуючого прямокутника»

Декодер «Мінімізація площі оточуючого прямокутника» (MERA) генерує BL-компактні упаковки та орієнтований на заповнення «дірок», що утворилися в процесі упаковки прямокутників. Алгоритм намагається розташувати кожен наступний прямокутник, зіставляючи кожен його кут з кожним із кутів уже упакованих прямокутників. Серед допустимих розташувань алгоритм обирає таке, за якого координати нижнього лівого кута нового прямокутника є мінімальними (площа оточуючого прямокутника є мінімальною; *оточуючим прямокутником* називають найменший прямокутник, описаний навколо упакованої частини смуги). У роботі [13] стверджується, що складність MERA-алгоритму є $O(m^2)$. Втім, строго кажучи, немає чіткого доведення цього факту: наводяться міркування про проведення аналогії із доведенням складності BL-декодера, хоча таку аналогію проводити не можна. Аналіз наведеного в [13] псевдокоду також викликає серйозні сумніви щодо правильності оцінки складності MERA-декодера. Зокрема, перебір пари прямокутників – нового для вставки та старого, з кутом якого проводиться зіставлення – вже є квадратичним. Додаткова перевірка на наявність перетинів з іншими, вже упакованими прямокутника-

ми, має в такому разі бути виконана за константний час. Задача зробити цю перевірку за константний час – як мінімум, нетривіальна, якщо взагалі розв'язна. Нам не вдалось знайти алгоритм, який би робив таку перевірку зі складністю O(1).

Для проведення експериментального аналізу MERA-декодера нами була розроблена його реалізація складності $O(m^3)$ з такими евристичними оптимізаціями.

Перша оптимізація – запам'ятовування вже декодованих перестановок. Експериментальний аналіз виявив, що у наборі особин за весь прогін алгоритму в середньому міститься лише близько 30 % унікальних хромосом, інші хромосоми – їх копії, що були отримані в результаті невдалого кросинговеру/мутації. Така оптимізація дозволяє більш ніж втричі зменшити об'єм обчислень, запам'ятовуючи значення функції здоров'я для вже обрахованих особин в окремому асоціативному масиві.

Друга оптимізація – відкидання безперспективних та малоперспективних комбінацій кутів. У псевдокоді, запропонованому в [13], розглядаються усі 16 комбінацій зіставлення кутів нового та старого прямокутників. Але 4 з цих комбінацій є апріорі неможливими (наприклад, верхній правий до верхнього правого), ще 4 з високою ймовірністю суперечать BL-умові (наприклад, розташування нового прямокутника повністю зліва або знизу старого). Було пораховано (експериментально), що за описаних комбінацій кутів більш ніж 99.9 % отриманих упаковок не проходили перевірку на допустимість. Тому було прийнято рішення про відкидання таких комбінацій заради прискорення роботи алгоритму.

Третя оптимізація – при пошуку місця розташування нового прямокутника спочатку перевіряти перетин з тими прямокутниками, що були укладені пізніше. Так недопустимість упаковки виявляється раніше. Ця ідея прискорила роботу програми ще приблизно на 20 %.

Щодо константи під асимптотикою – вона повністю визначається кількістю комбінацій кутів зіставлення і дорівнює 8.

Декодер «Нижній лівий із заповненням»

Декодер «Нижній лівий із заповненням» (BLF) був запропонований у [10]. Цей декодер також генерує BL-компактні упаковки та здатен заповнювати дірки, що утворилися під час упаковки. Але алгоритм заповнення дірок є зовсім іншим. BLF-декодер постійно тримає в пам'яті упорядкований зліва направо та знизу догори список точок, які можуть бути координатами лівого нижнього кута чергового прямокутника; послідовно переглядаючи цей список, алгоритм намагається розташувати черговий прямокутник у вільному місці та, у разі успіху, оновлює список точок. Цей список фактично є декартовим добутком списку *X*-координат лівих та правих кінців вже упакованих прямокутників та аналогічного списку *Y*-координат. Саме пріоритет в бік зменшення *X*-координати і дає тенденцію заповнення дірок BLF-декодером.

Наївна складність BLF-алгоритму – $O(m^4)$, покращена – $O(m^3)$. Найкраща існуюча реалізація має складність $O(m^2)$ [10]; втім, ця реалізація використовує специфічний спосіб кодування, є об'ємною та важкою для розуміння. Тому нами був реалізований алгоритм складності $O(m^3)$; псевдокод цього алгоритму показано на рис. 1.

Реалізація алгоритму поєднує дві ідеї для отримання кубічної складності: метод скануючої прямої та метод двох указників.

Перший метод у цій реалізації слугує для перетворення нескінченної кількості варіантів розміщення нового прямокутника на скінченну: по осі *OY* обираються лише координати нижніх та верхніх кінців вже упакованих прямокутників. Ці координати грають роль «подій» для алгоритму – в них або новий прямокутник починається (нижній край), або закінчується (верхній край). У результаті задача перевірки допустимості нового

Алгоритм BLF(permutation, rectangles)

```
if permutation in permutation_buffer:
return permutation_buffer[permutation]
#Список Y-координат початків та кінців прямокутників
#Перший прямокутник встановлюємо в точку (0,0)
events = list((0, open), (rectangles[0].width, close))
for new_rectangle in rectangles:
for left_border in all_possible_left_borders:
opened_rectangles = 0, closed_rectangles = 0
bot_index = 0, top_index = 0
    while events[bot_index].Y + new_rectangle.width<= W:
while events[top_index].Y <= events[bot_index].Y + new_rectangle.width:
if events[top_index].type == open and coincide_by_X:
opened_rectangles = opened_rectangles + 1
top_index = top_index+1
if events[bot_index].Y == close and coincide_by_X:
closed_rectangles = closed_rectangles + 1
if opened_rectangles == closed_rectangles:
new_rectangle.X = left_border
new_rectangle.Y = events[bot_index].Y
break
bot_index = bot_index + 1
events.append(new_rectangle.Y, open)
events.append(new_rectangle.Y + new_rectangle.width, close)
     sort(events)
permutation_buffer[permutation] = placing
return placing
```

Рис. 1. Псевдокод алгоритму BLF-декодера

розташування прямокутника зводиться до перевірки, скільки прямокутників було «відкрито» до верхньої границі нового прямокутника і скільки «завершено» до його нижньої границі. Якщо ці кількості збігаються, розміщувати прямокутник можна, інакше є старі прямокутники, що перетинають новий в його поточному положенні.

Метод двох указників, у свою чергу, робить два прогони по масиву «подій» одночасно замість того, щоб один прогін був вкладений в інший. Ці два указники вказують на нижню границю для укладки нового прямокутника та на останню «подію» перед верхньою границею. Нескладно побачити, що при зсуві вгору нижнього указника верхній також може просунутись тільки вгору. Це й зменшує складність двох циклів while з $O(m^2)$ до $O(m)$, а загальну складність роботи алгоритму – до $O(m^3)$.

Щодо константи під асимптотикою: зовнішній цикл виконується m разів, лівих кінців може бути до $2m$, нижніх і верхніх – так само. В результаті отримуємо $m * 2m * (2m + 2m) = 8m^3$ – верхню границю кількості операцій.

Функція пристосованості

Найочевиднішим вибором функції пристосованості для даної задачі є або мінімізація довжини упакованої частини смуги, або – що майже еквівалентно – максимізація відношення загальної площі упакованих прямокутників до площі зайнятої ними частини смуги.

Така функція пристосованості оцінює лише найважливіший атрибут розв'язку – безпосередню відповідь на задачу. Проте оцінювати пристосованість хромосоми можна не так «сухо», наприклад, ввести доданок «суб'єктивної естетичності» – функцію від площі дірок, утворених при упаковці [6]. Звісно, цей доданок потрібно взяти з меншим ваговим коефіцієнтом, ніж перший, але він буде враховувати «привабливість» розкрою. Для визначення цього доданку наведемо декілька понять, введених в [6].

Розрідженістю гена хромосоми називають величину

$$sparseness\_gene(i) = \frac{free\_space(i)}{block\_width(i)},$$

де $free\_space(i)$ – частина лівої сторони i-го прямокутника, що не дотична до жодного іншого прямокутника або до краю смуги, $block\_width(i)$ – ширина i-го прямокутника.

Відносною розрідженістю хромосоми називають величину

$$sparseness(X) = \sum_{i=1}^{m} sparseness\_gene(i),$$

де m – кількість генів у хромосомі, $sparseness\_gene(i)$ – розрідженість i-го гену.

Функцію пристосованості визначимо таким чином:

$$Fitness(X) =$$

$$= 10 * \frac{S\_shapes}{S} - sparseness(X) \to \max,$$

де S_shapes – загальна площа прямокутників, S – площа використаної частини смуги. Коефіцієнт 10 був підібраний емпірично.

Огляд широко вживаних функцій пристосованості для задачі, що розглядається, наведено в [2].

Відбір

Для аналізу були обрані такі оператори відбору.

Відбір за методом рулетки (Roulette-Wheel Selection, RWS). Згідно з цим методом, ймовірність особини бути обраною в батьківський пул прямо пропорційна значенню її пристосованості:

$$P(X_i) = \frac{f(X_i)}{\sum_{j=1}^{N} f(X_j)},$$

де $f(X_i)$ – коефіцієнт пристосованості хромосоми X_i, N – кількість хромосом в популяції.

Стохастичний універсальний відбір (Stochastic Universal Selection, SUS). Цей оператор відбору має більш детерміновану природу, ніж рулетка, а саме обирає рівномірно розподілені значення з ймовірнісного відрізка. Це дає реальний шанс слабшим особинам вижити та загалом пом'якшує шум відбору [2].

Ці два методи відбору є антиподами і ведуть себе по-різному на різних задачах. З метою дослідження впливу відбору на якість роботи алгоритму реалізовані обидва оператори.

Оператор кросинговеру

Кросинговер є основним оператором ГА. У роботі реалізовано два оператори кросинговеру, що широко використовуються в комбінаторних задачах.

PMX-кросинговер (Partially Mapped Crossover). Робота оператора складається з двох кроків. На першому кроці випадково обираються дві точки схрещування та відбувається обмін генетичним матеріалом між обраними точками. У більшості випадків такий обмін частинами породжує недопустимі хромосоми: різні гени хромосоми можуть набути однакових значень. Тому другий крок

полягає в усуненні дублів. Якщо k значень у кожному протонащадку продубльовані (а кількість дублів буде однакова), то на позиціях поза зонами обміну дублі замінюються на продубльовані значення іншого протонащадка зі збереженням порядку цих значень.

OX-кросинговер (Order Crossover) з одною точкою схрещування. Цей оператор кросинговеру розрахований на збереження не стільки абсолютних позицій елементів, скільки їх відносного порядку в перестановці. Для цього випадково обирається точка схрещування. Значення генів зліва від цієї точки переносяться нащадкам без змін. Значення генів справа від неї переносяться до нащадків у порядку, в якому вони зустрічаються у другій батьківській хромосомі.

Оператор мутації

Оператор мутації у генетичному алгоритмі використовується для урізноманітнення генетичного матеріалу популяції. Для аналізу було обрано *мутацію вставкою (Insertion Mutation)*. В ній випадково обирається позиція в хромосомі, значення гену вилучається з цієї позиції та вставляється у випадково обрану нову позицію. Для задачі двовимірної упаковки з поворотами прямокутників на кут, кратний 90°, використовується мутація, яка змінює знак гена при повороті.

Результати

Для порівняння роботи декодерів MERA та BLF, для аналізу роботи генетичного алгоритму та виявлення оптимальних значень його параметрів було проведено серію обчислювальних експериментів.

Тестові дані

Як тестові дані використані відомі тести Єви Гоппер [14]. Ці тести мають властивість *гільйотинності*: оптимальний розв'язок може бути отриманий в результаті серії розрізів, паралельних осям координат, кожен з яких перетинає усю ширину або довжину прямокутника, що залишився. Більш того, тестові дані мають ідеальну упаковку. Інакше кажучи, щоб отримати n прямокутників, береться один великий прямокутник і n-1 разів розрізається паралельно осям координат. Оптимальна довжина смуги тому відома заздалегідь (як довжина початкового прямокутника).

Тестові приклади розбиті на 7 категорій, в кожній різна кількість прямокутників. З метою прискорення обчислень запуски відбувались на тестах перших чотирьох категорій. Характеристики використаних тестових наборів наведено у табл. 1.

Критерієм оцінювання розв'язку, знайденого генетичним алгоритмом, слугує значення відхилення (у відсотках) знайденої алгоритмом довжини упаковки від оптимальної довжини смуги:

$$gap = \frac{L - L_{opt}}{L_{opt}} \cdot 100\%,$$

де L – найкращий знайдений розв'язок алгоритму, L_{opt} – оптимальна довжина смуги.

Таблиця 1. Характеристика тестів Є. Гоппер

Категорія/приклади	Кількість об'єктів	Ширина смуги	Оптимальна довжина
C1/P1,P2	16,17	20	20
C2/P1	25	40	15
C3/P1	28	60	30
C4/P1	49	60	60

Результати експериментів

У табл. 2 наведено результати експериментів для описаних тестових наборів у випадку задачі із забороною поворотів прямокутників з використанням різних генетичних операторів, описаних вище. Інші параметри генетичного алгоритму задавались так: максимальна кількість ітерацій – 100; кількість особин у популяції – 100; ймовірність застосування оператора кросинговеру – 1; ймовірність застосування оператора мутації – 0.07. Для аналізу не тільки якості знайдених розв'язків, а й стабільності роботи алгоритму виконувалось по 10 прогонів генетичного алгоритму з однаковими параметрами для кожної задачі, найкращий та середній результати по всіх прогонах зберігались.

Таблиця 2. Результати для задачі без поворотів (gap, %)

Параметри								Тестові набори									
decoder		selection		crossover		mutation		C1/P1		C1/P2		C2/P1		C3/P1		C4/P1	
MERA	BLF	RWS	SUS	OX	PMX	Insertion	None	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
*		*		*		*		5.00	8.50	10.00	12.50	20.00	22.00	10.00	16.67	11.67	12.83
*		*		*			*	5.00	11.00	10.00	15.50	20.00	26.00	16.67	20.00	11.67	14.17
*		*			*	*		0.00	8.00	5.00	11.00	20.00	22.00	13.33	18.33	10.00	11.33
*		*			*		*	0.00	8.00	10.00	13.30	20.00	22.00	10.00	16.67	10.00	13.33
*			*	*		*		0.00	6.50	5.00	10.00	20.00	20.00	10.00	16.00	10.00	11.33
*			*	*			*	0.00	7.00	5.00	11.50	13.33	21.33	16.67	18.67	11.67	12.33
*			*		*	*		0.00	4.00	10.00	10.00	20.00	20.00	10.00	15.33	10.00	12.00
*			*		*		*	0.00	6.50	10.00	10.50	20.00	20.00	16.67	17.00	11.67	12.33
	*	*		*		*		0.00	5.50	10.00	10.00	13.33	21.33	23.33	28.00	11.67	12.00
	*	*		*			*	0.00	6.50	5.00	9.50	20.00	22.67	26.67	28.33	11.67	13.00
	*	*			*	*		0.00	1.50	5.00	9.00	20.00	20.00	20.00	26.00	10.00	11.00
	*	*			*		*	0.00	6.00	10.00	10.00	20.00	20.00	23.33	26.67	10.00	11.67
	*		*	*		*		0.00	1.50	5.00	8.00	20.00	20.00	20.00	25.67	11.67	12.00
	*		*	*			*	0.00	2.00	5.00	9.00	20.00	20.00	16.67	26.67	11.67	12.33
	*		*		*	*		0.00	1.50	5.00	8.00	20.00	20.00	16.67	24.67	10.00	11.33
	*		*		*		*	0.00	1.50	5.00	9.00	20.00	20.00	23.33	23.67	11.67	12.33

У табл. 3 наведені середні по прогонах та по задачах результати для кожного набору параметрів. Як видно з цієї таблиці, найкращі результати дає декодер MERA в комбінації зі стохастичним відбором та мутацією вставкою. Водночас, декодер BLF працює стабільніше, з меншою дисперсією. Враховуючи однакову часову складність реалізації алгоритмів, рекомендуємо використовувати декодер MERA із вдало підібраними значеннями інших параметрів.

Аналізуючи табл. 3, також можна констатувати, що застосування мутації є необхідним для успішної роботи алгоритму: за кожного набору параметрів алгоритм без мутації давав гірший результат, ніж з нею. Стохастичний універсальний відбір показав очевидні переваги перед відбором за методом рулетки, що не суперечить теорії: цей метод має менший шум відбору та стабільніший тиск відбору при переході від покоління до покоління [2]. Зазначимо, що тип оператора кросинговеру особливого впливу на результати не має.

Таблиця 3. Середні результати для задачі без поворотів (gap, %)

Параметри								Середні значення					
декодер		відбір		кросинговер		мутація		По всіх наборах		На наборах C1		На наборах C2, C3, C4	
MERA	BLF	RWS	SUS	OX	PMX	Insertion	None	Avg(Best)	Avg(Average)	Avg(Best)	Avg(Average)	Avg(Best)	Avg(Average)
*		*		*		*		11,33%	14,50%	7,50%	10,50%	13,89%	17,17%
*		*		*			*	12,67%	17,33%	7,50%	13,25%	16,11%	20,06%
*		*			*	*		9,67%	14,13%	2,50%	9,50%	14,44%	17,22%
*		*			*		*	10,00%	14,66%	5,00%	10,65%	13,33%	17,33%
*			*	*		*		*9,00%*	*12,77%*	2,50%	8,25%	**13,33%**	**15,78%**
*			*	*			*	9,33%	14,17%	2,50%	9,25%	13,89%	17,44%
*			*		*	*		**10,00%**	**12,27%**	5,00%	7,00%	**13,33%**	**15,78%**
*			*		*		*	11,67%	13,27%	5,00%	8,50%	16,11%	16,44%
	*	*		*		*		11,67%	15,37%	5,00%	7,75%	16,11%	20,44%
	*	*		*			*	12,67%	16,00%	2,50%	8,00%	19,45%	21,33%
	*	*			*	*		11,00%	13,50%	2,50%	5,25%	16,67%	19,00%
	*	*			*		*	12,67%	14,87%	5,00%	8,00%	17,78%	19,45%
	*		*	*		*		11,33%	13,43%	**2,50%**	**4,75%**	17,22%	19,22%
	*		*	*			*	10,67%	14,00%	2,50%	5,50%	16,11%	19,67%
	*		*		*	*		10,33%	13,10%	**2,50%**	**4,75%**	15,56%	18,67%
	*		*		*		*	12,00%	13,30%	2,50%	5,25%	18,33%	18,67%

Таким чином, для розв'язку задачі двовимірної ортогональної упаковки із забороною поворотів прямокутників у загальному випадку можна порекомендувати наступний набір параметрів: декодер MERA, стохастичний універсальний відбір, PMX-кросинговер, мутація вставкою.

Більш глибокий аналіз отриманих результатів дозволяє диференціювати випадки незначної (до 20) та великої (більше 20) кількості прямокутників, що підлягають упаковці. У першому випадку рекомендований набір параметрів: декодер BLF, стохастичний універсальний відбір, PMX-кросинговер або OX-кросинговер, мутація вставкою; в другому випадку – декодер MERA, стохастичний універсальний відбір, PMX-кросинговер або OX-кросинговер, мутація вставкою. Назвемо ГА з такими параметрами GA+MERA та порівняємо його з роботою відомих алгоритмів на цих же тестах. В порівнянні окрім GA+MERA братимуть участь алгоритми GA+IBL ([6]), H-SP та R-GRASP ([16]), а також розроблений в [17] GA+BLF. Алгоритм R-GRASP на сьогодні вважається найкращим для задачі без поворотів прямокутників. Результати порівняння наведені у табл. 4. Як видно з таблиці, розроблений нами генетичний алгоритм GA+MERA поступається навіть GA+IBL, який має меншу часову складність. Водночас GA+MERA поступається розробленому в [17] GA+BLF, але є кращим за розроблений нами GA+BLF. Аналіз табл. 3 показує чітку тенденцію до однакової динаміки поведінки генетичного алгоритму для кожного з розглянутих декодерів (MERA, BLF) при відповідних змінах інших параметрів. Це дозволяє припустити, що параметри алгоритму в цілому були підібрані не дуже вдало, а за більш вдалого набору параметрів (наприклад, набір параметрів у сполученні з евристичними оптимізаціями, що були використані в роботі [17]), алгоритм GA+MERA міг би дати значно кращі результати як у порівнянні з GA+IBL, так і у порівнянні з GA+BLF. Це мета наших подальших досліджень.

Результати для задачі з поворотами

Для задачі з можливістю поворотів були застосовані ті ж значення параметрів, що й для задачі без поворотів. Результати для різних наборів параметрів подані у табл. 5. Висновок, який можна зробити з цієї таблиці, полягає у тому, що самі повороти нічим не покращили якість знайдених розв'язків. Причиною є те, що разом зі збільшенням кількості якісних розв'язків збільшилась і кількість неякісних, а характер пошуку не сильно змінився. Отже, необхідне проведення подальших досліджень з використанням комбінації поворотів та мутації вставкою.

Таблиця 4. Порівняння роботи алгоритму GA+MERA з іншими відомими алгоритмами для задачі із забороною поворотів (gap, %)

Клас	GA+BLF	GA+MERA		GA+IBL		R-GRASP		H-SP	
	Best	Best	Avg	Best	Avg	Best	Avg	Best	Avg
C1	4	2.5	4.75	5	5	0	0	0	1.33
C3	5	10	15	7	8	1.08	1.08	2.22	2.22
C4	3	10	12	8	10	1.64	1.64	1.67	2.11
Avg	4	7.5	10.58	6.66	7.66	0.91	0.91	1.30	1.89

Таблиця 5. Результати для задачі з поворотами (gap, %)

selection		crossover		C1/P1		C1/P2		C2/P1		C3/P1		C4/P1		Total average
RWS	**SUS**	**OX**	**PMX**	**Best**	**Avg**	**Best**	**Avg**	**Best**	**Avg**	**Best**	**Avg**	**Best**	**Avg**	
*		*		5.00	8.00	5.00	7.00	13.33	14.67	10.00	12.67	10.00	11.33	10.73
*			*	5.00	7.50	5.00	7.00	13.33	13.33	10.00	12.00	10.00	11.33	10.23
	*	*		5.00	7.00	5.00	8.00	13.33	13.33	10.00	13.33	10.00	11.67	10.67
	*		*	5.00	8.00	5.00	9.00	13.33	13.33	10.00	11.33	10.00	12.00	10.73

Висновки та рекомендації

У цьому дослідженні проаналізовано вплив параметрів на роботу генетичного алгоритму розв'язку задачі двовимірної ортогональної упаковки прямокутних об'єктів у напівнескінченну смугу фіксованої ширини з можливістю поворотів на 90° та без неї. Особливу увагу приділено аналізу декодерів MERA та BLF. Виділено найкращі набори параметрів. Проведено порівняльний аналіз запропонованого алгоритму з найкращими відомими.

Розроблений алгоритм GA+MERA поступається найкращим відомим сьогодні алгоритмам розв'язку поставленої задачі: найкращі по всіх прогонах результати є гіршими на 5 % – 13 % у випадках з дозволеними поворотами та без них. Цей алгоритм показав навіть гірші результати у порівнянні з GA+IBL, що має меншу часову складність. Втім, беручі до уваги результати роботи [17] та дані з табл. 3, є підстави сподіватись на суттєве покращення роботи алгоритму за умови більш вдалого підбору інших параметрів. Окреслимо можливі шляхи удосконалення.

- Впровадження *оператора упаковки*, що об'єднує прямокутники, які мають однакову довжину/ширину та упаковані поруч, в один метапрямокутник [6].
- Застосування параметрів та евристик, запропонованих в роботі [17].
- Модифікація функції пристосованості.
- Застосування кількох декодерів та автоматичний вибір декодера для розташування чергового об'єкта.

1. *Стоян Ю.Г., Яковлев С.В.* Математические модели и оптимизационные методы геометрического проектирования. – Киев: Наукова думка, 1986. – 268 с.
2. *Глибовець М.М., Гулаєва Н.М.* Еволюційні алгоритми: підручник. – НаУКМА, 2013. – С. 334–357.
3. *Мухачева Э.А., Мухачева А.С.* Л.В. Канторович и задачи раскроя- упаковки: новые подходы для решения комбинаторных задач линейного раскроя и прямоугольной упаковки // Записки научных семинаров ПОМИ. – 2004. – Т. 312. – С. 239–255.
4. *Fowler R.J., Paterson M.S., Tatimoto S.L.* Optimal packing and covering in the plane are NP-complete // Information Processing Letters. – 1981 – N 12. – P. 133–137.
5. *Гребенник И.В., Панкратов А.В., Чугай А.М. и др.* Упаковка n-мерных параллепипедов с возможностью изменения их ортогональной ориентации в n-мерном параллелепипеде // Кибернетика и системный анализ. – 2010. – № 5. – С.122–131.

6. *Гулаєва Н.М., Щур О.П.* Аналіз параметрів генетичного алгоритму розв'язку задачі ортогональної упаковки // Наукові записки НаУКМА. – 2012. – Т. 138: Комп'ютерні науки. – С. 6–14.
7. *Чеканин В.А.* Модифицированные эволюционные алгоритмы и программные решения задачи ортогональной упаковки объектов // Автореф. дис. ... канд. техн. наук. М.: УГАТУ, 2011. – 14 с.
8. *Lesh.N., Marks J., Mc. Mahon A.* Exhaustive approaches to 2D rectangular perfect packings. Information Processing Letters – 2004. – P. 7–14.
9. *Baker B.S., Coffman E.G., Rivest R.L.* Orthogonal packings in two dimensions // SIAM Journal on computing
10. *Chazelle B.* The bottom left bin packing heuristic: an efficient implementation // IEEE Transaction on computers. – 1983 – N 32. – P. 697–707.
11. *Lesh N., Mitzenmacher M.* Bubble search: a simple heuristic for improving priority-based greedy algorithms // Information processing letters. – 2006. – P. 161–169.
12. *Bortfeldt A.* A genetic algorithm for the two-dimensional strip-packing problem with rectangular pieces. European Journal of Operational Research. – 2006. – P. 814–837.
13. *Ahmad, Abdul-Rahim,* An Intelligent Expert System for Decision Analysis and Support in Multi-Attribute Layout Optimization, PhD thesis, University of Waterloo, Ontario, Canada, 2005. – 207 p.
14. *Hopper E., Turton B.C.H.* Problem generators for rectangular packing problems // Computers in Engineering. – 2000. – P. 123–135.
15. *Riff M.C., Bonnaire X., Neveu B.* A revision of recent approaches for two-dimensional strip-packing problems // Engineering application of Artificial Intelligence. – 2009.
16. *Araya I., Neveu B., Riff M.C.* An Efficient Hyperheuristic for Strip Packing problem // Book Adaptive and Multilevel Heuristics, Series Studies in Computational Intelligence. – Springer. – 2008. – P. 61–76.
17. *Hopper E., B. C. H. Turton.* An empirical investigation of metaheuristic and heuristic algorithms for a 2D packing problem. Eur. J. Oper. Res. – 2000. – Vol. 128. – P. 34–57.

References

1. Stoyan Y.G., Yakovlev S.V. Mathematical models and optimization methods of the geometric projection. – Kiev: Scientific thought. – 1986. – 268 p.
2. Glibovets M.M., Gulayeva N.M. Evolutionary algorithms: a textbook. – NaUKMA, 2013. – P. 334–357.
3. Mukhacheva E.A., Mukhacheva A.S. L.V. Kantorovich and problems of cutting-package, new approaches to combinatorial problems of linear cutting and rectangular package // Notes of POMI scientific seminars. – 2004. – Vol. 312. – P. 239–255.
4. Fowler R.J., Paterson M.S., Tatimoto S.L. Optimal packing and covering in the plane are NP-complete // Information Processing Letters. – 1981 – N 12. – P. 133–137.
5. Grebennik I.V., Pankratov A.V., Chugai A.M., Baranov A.V. Packaging n-dimensional parallelepipeds with the ability to change their orthogonal orientation in n-dimensional parallelepiped // Cybernetics and Systems Analysis. – 2010. – N 5. – P. 122–131.
6. Gulayeva N.M. Analysis of the parameters of the genetic algorithm solving the problem of orthogonal package / Gulayeva N.M., Shchur O.P. // Scientific notes NaUKMA. – 2012. – Vol 138: Computer science. – P. 6–14.
7. Chekanin V.A. Modified evolutionary algorithms and programmatic solving of the problem of orthogonal packing of objects // Abstract. Dis. on the scientific degree of Ph.D. : 05.13.17. – Moscow: USATU, 2011. – 14 p.
8. Lesh.N., Marks J., Mc. Mahon A. Exhaustive approaches to 2D rectangular perfect packings. Information Processing Letters – 2004. – P. 7–14.
9. Baker B.S., Coffman E.G., Rivest R.L. Orthogonal packings in two dimensions // SIAM Journal on computing
10. Chazelle B. The bottom left bin packing heuristic: an efficient implementation // IEEE Transaction on computers. – 1983 – N 32. – P. 697–707.
11. Lesh N., Mitzenmacher M. Bubble search: a simple heuristic for improving priority-based greedy algorithms // Information processing letters. – 2006. – P. 161–169.

12. Bortfeldt A. A genetic algorithm for the two-dimensional strip-packing problem with rectangular pieces. European Journal of Operational Research. – 2006. – P. 814–837.
13. Ahmad, Abdul-Rahim, An Intelligent Expert System for Decision Analysis and Support in Multi-Attribute Layout Optimization, PhD thesis, University of Waterloo, Ontario, Canada, 2005. – 207 p.
14. Hopper E., Turton B.C.H. Problem generators for rectangular packing problems // Computers in Engineering. – 2000. – P. 123–135.
15. Riff M.C., Bonnaire X., Neveu B. A revision of recent approaches for two-dimensional strip-packing problems // Engineering application of Artificial Intelligence. – 2009.
16. Araya I., Neveu B., Riff M.C. An Efficient Hyperheuristic for Strip Packing problem // Book Adaptive and Multilevel Heuristics, Series Studies in Computational Intelligence. – Springer. – 2008. – P. 61–76.
17. Hopper E., B. C. H. Turton. An empirical investigation of metaheuristic and heuristic algorithms for a 2D packing problem. Eur. J. Oper. Res. – 2000. – Vol. 128. – P. 34–57.

Одержано 01.07.2016

Про авторів:

Глибовець Микола Миколайович,
доктор фізико-математичних наук,
професор,
декан факультету інформатики НаУКМА.
Кількість наукових публікацій в
українських виданнях – понад 150.
Кількість наукових публікацій в
іноземних виданнях – 11.
Індекс Хірша – 1.
http://orcid.org/0000-0002-3853-2171,

Гулаєва Наталія Михайлівна,
кандидат фізико-математичних наук,
доцент кафедри інформатики НаУКМА.
Кількість наукових публікацій в
українських виданнях – 20.
Кількість наукових публікацій в
іноземних виданнях – 1.
Індекс Хірша – 1.
http://orcid.org/0000-0003-4588-0702,

Морозов Ігор Олегович,
магістр факультету інформатики НаУКМА.
Кількість наукових публікацій в
українських виданнях – 0.
Кількість наукових публікацій в
іноземних виданнях – 0.
Індекс Хірша – 0.
http://orcid.org/0000-0003-4085-170X.

Місце роботи авторів:

Національний університет
«Києво-Могилянська Академія»,
04655, м. Київ,
вул. Сковороди, 2.
Тел.: (044) 463 6985,
факс: (044) 416 4515.
E-mail: glib@ukma.kiev.ua,
ngulayeva@yahoo.com,
hospital-at-home@yandex.ru