УДК 004.02:519.854

О. Жданова, О. Папка, Л. Рибачук, О. Савчук, Г. Соболевський

ЗАДАЧА ФОРМУВАННЯ ЗОН ВІДПОВІДАЛЬНОСТІ НА МНОЖИНІ ОБ’ЄКТІВ ПЛОЩИНИ ЗА КРИТЕРІЄМ МІНІМІЗАЦІЇ РІЗНИЦІ СУМАРНИХ ВАГ

Анотація. Робота присвячена дослідженню оптимізаційної задачі, у якій необхідно розбити множину об’єктів, для яких відомі вага та координати розміщення, на дві підмножини (зони), кожна з яких закріплена за заданими об’єктами–базами (з відомими координатами). Необхідно побудувати розмежувальну лінію, що ділить ділянку на дві зони так, щоб одна зона відповідала одній базі, друга – іншій, і при цьому різниця між зваженими кількостями об’єктів, що попали в різні зони, була мінімальною. Розроблено чотири алгоритми розв’язання задачі: два евристичних і два генетичних. Проведена серія експериментів, метою яких був порівняльний аналіз розроблених алгоритмів за часом роботи та точністю.

Ключові слова: задача розбиття множини на підмножини, зони відповідальності, евристичний алгоритм, генетичний алгоритм

Вступ

Стаття присвячена дослідженню оптимізаційної задачі, у якій необхідно розбити множину об’єктів, для яких відомі координати розміщення, на дві підмножини (зони), кожна з яких закріплена за заданими об’єктами (базами).

Досліджувана задача, по суті є «перетином» двох класичних задач:

- задачі розбиття множини на підмножини;
- задачі кластеризації.

Задача розбиття множини на підмножини (Subset Sum Problem, SSP) – це комбінаторна задача, у якій елементи заданої множини повинні бути розділені на підмножини відповідно до певних правил або обмежень [1]. Ця задача завдяки своїй універсальності знаходить застосування в управлінні, телекомунікаціях, машинному навчанні, обробці зображень, економіці, соціології, біології та інших галузях. Існує декілька типів задач розбиття множини на підмножини, наприклад:

1) розбиття на підмножини за кількістю елементів – розбиття на підмножини фіксованого розміру;

2) розбиття з обмеженнями на кількісні характеристики підмножин – мають місце певні обмеження на розмір підмножин або кількість підмножин;

3) розбиття на підмножини з обмеженнями – задачі, в яких встановлені певні обмеження на самі елементи множини або на структуру розбиття;

4) розбиття на підмножини з визначеними властивостями – у цьому випадку необхідно знайти розбиття множини на підмножини, що має певні характеристики або властивості;

5) розбиття на підмножини з урахуванням порядку елементів у підмножинах.

Залежно від типів задач, наразі розроблені різні методи та алгоритми їх вирішення.

У роботі [2] запропонована схема розв’язання SSP на основі ройових алгоритмів оптимізації з різними формами типу відбору, балансом розвідки та експлуатації, міркуваннями мультимодальності та простором пошуку.

У [3] представлений алгоритм зворотного поширення для обмеження поділу глобальної множини, який, порівняно з іншими підходами в програмуванні з обмеженнями, має значно меншу трудомісткість.

Для задач розбиття на підмножини з урахуванням комбінаторних властивостей важливі комбінаторні аспекти розбиття. В роботах [4] та [5] досліджуються комбінаторні алгоритми для задачі розбиття множини на підмножини.

Робота [6] присвячена дослідженню задачі розбиття множини на підмножини, запропоновано новий наближений алгоритм, що базується на метаевристичному алгоритмі під назвою «Алгоритм арифметичної оптимізації».

У [7] представлено вдосконалений генетичний алгоритм для вирішення SSP. На відміну від звичайного генетичного алгоритму, запропонований алгоритм виконує умовні (і не ймовірні) кросинговер і мутацію.

Евристичний метод для розв'язання задачі розбиття множини на підмножини наведено у [8], де описано кілька евристик для вирішення SSP та запропоновано нову евристику, що заснована на локальному пошуку.

В роботі [9] запропоновано новий алгоритм BalsubLast для задачі розбиття на підмножини, в основу якого покладено збалансоване динамічне програмування Пісінгера.

У [10] розроблено та реалізовано генетичний алгоритм з відповідними операторами мутації та кросинговеру для вирішення задачі розбиття множини на підмножини. Результати роботи алгоритму оцінюються серед різних екземплярів з різною початковою популяцією та часом виконання. Також у роботі наведено традиційний підхід динамічного програмування, що дотримується стратегії «знизу вгору», для вирішення SSP. Експерименти показали, що генетичний алгоритм не був кращим через його довший час виконання.

В роботі [11] пропонується новий масштабований алгоритм розв'язання задачі розбиття на підмножини, який базується на узагальненій моделі молекулярного обчислення.

Кластеризація даних є ефективним і поширеним підходом до їх групування даних за певним шаблоном або властивій подібності даних в одній групі [12]. Важливим аспектом є розробка алгоритмів, які здатні ефективно опрацьовувати різні типи даних і враховувати їхню внутрішню структуру, щоб забезпечити точне та адекватне кластеризаційне розділення. З урахуванням особливостей досліджуваної задачі, стандартні методи кластеризації застосувати доволі важко.

З урахуванням того, що існує кілька типів задач розбиття множини на підмножини, для кожного з них розробляються методи, які враховують особливості задачі.

Постановка задачі

Змістовна постановка задачі

Нехай маємо ділянку, на якій розміщено n об’єктів, координати яких відомі: (x_1, y_1), (x_2, y_2), …, (x_n, y_n). Для об’єкта j $(j = 1, \dots, n)$ відома його вага (пріоритетність) $1 \le w_j \le W$. Є ще два пункти (які будемо називати базовими) з відомими координатами розміщення: $A(x_A, y_A)$, та $B(x_B, y_B)$ $((x_A, y_A) \ne (x_B, y_B))$.

Необхідно побудувати розмежувальну лінію, що ділить ділянку на дві зони так, щоб одна зона відповідала базі A, друга – базі B, з метою мінімізації різниці між зваженими кількостями об’єктів, що попали в різні зони.

На рис. 1 наведено приклад задачі з $n = 8$ (в середині кола, що відповідає об’єкту, вказана відповідна вага).

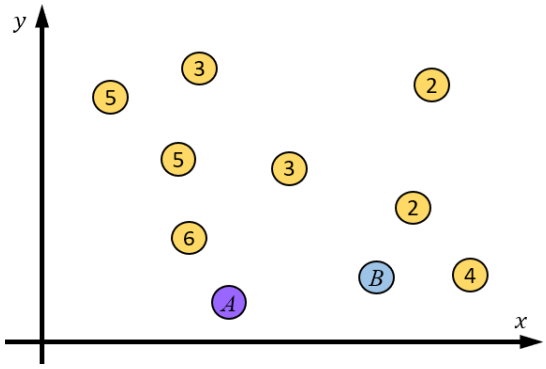
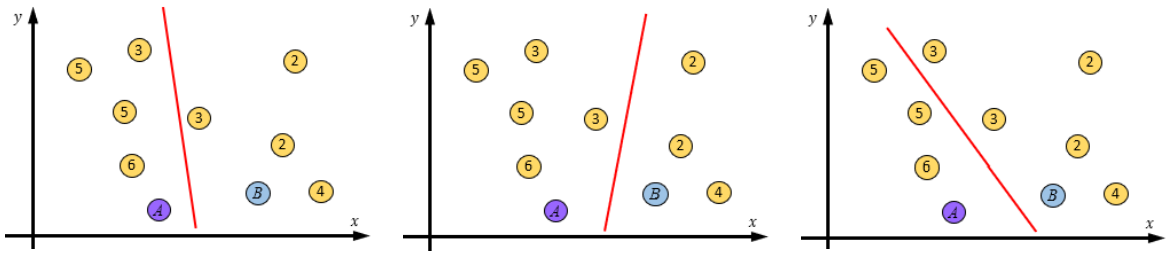


Рис. 1. Приклад задачі з $n = 8$

На рис. 2 наведені три можливі варіанти розташування розмежувальної лінії. Їм відповідають такі значення критерія:

– варіант а): $|19 - 11| = 7$;
– варіант б): $|22 - 8| = 14$;
– варіант в): $|16 - 14| = 2$.



а) б) в)

Рис. 2. Деякі варіанти розташування розмежувальної лінії

Математична постановка задачі

Сформулюємо математичну модель задачі. Оскільки метою задачі є знаходження розмежувальної лінії $ax + by + c = 0$, то шуканими величинами є коефіцієнти цієї прямої: a, b, c.

В основу математичної моделі задачі покладено поняття відхилення [13] точки $M(x_0, y_0)$ від прямої. Відхиленням δ_M точки $M(x_M, y_M)$ від прямої $ax + +by + c = 0$ називається величина

$$\delta_M = \frac{ax_M + by_M + c}{\sqrt{a^2 + b^2}}. \quad (1)$$

Визначимо такі величини:

δ_A – відхилення бази $A(x_A, y_A)$ від прямої $ax + by + c = 0$;

δ_B – відхилення бази $B(x_B, y_B)$ від цієї прямої ;

δ_j – відхилення об’єкта (x_j, y_j) від цієї прямої $(j = 1, \dots, n)$.

Позначимо через z_j булеву змінну, що вказує на те, в якій з двох півплощин лежать бази та об’єкти:

$$z_A = \begin{cases} 1, якщо\ \delta_A \le 0, \\ 0, якщо\ \delta_A > 0, \end{cases}$$

$$z_B = \begin{cases} 1, якщо\ \delta_B \le 0, \\ 0, якщо\ \delta_B > 0, \end{cases}$$

$$z_j = \begin{cases} 1, якщо\ \delta_j \le 0, \\ 0, якщо\ \delta_j > 0, \end{cases} \quad j = 1, \dots, n.$$

Для того, щоб бази A та B лежали по різні сторони розмежувальної прямої, повинно виконуватись обмеження:

$$z_A + z_B = 1.$$

Оскільки необхідно мінімізувати різницю між зваженими кількостями об’єктів, що попали в різні зони, то цільова функція матиме наступний вигляд:

 ISSN 1560-8956

$$\left|\sum_{j=1}^{n} w_j z_j - \sum_{j=1}^{n} w_j (1 - z_j)\right| \to min. \qquad (2)$$

В (2) в різниці під знаком модуля зменшуване відповідає зваженій кількості об’єктів однієї з зон, а від’ємник – протилежної.

Отже, математична модель така.

Знайти такі коефіцієнти прямої $a, b\ ma\ c$, за яких досягає мінімуму функція

$$\left|\sum_{j=1}^{n} w_j z_j - \sum_{j=1}^{n} w_j (1 - z_j)\right| \to min,$$

при обмеженнях

$$z_A + z_B = 1,$$

$$z_A, z_B, z_1, z_2, \dots z_n \in \{0,1\}.$$

Достатні умови оптимальності розв’язку:

– якщо $\sum_{j=1}^{n} w_j$ є парним, то розв’язок, у якого

$$z_w^* = \left|\sum_{j=1}^{n} w_j z_j - \sum_{j=1}^{n} w_j (1 - z_j)\right| = 0,$$

є оптимальним;

– якщо $\sum_{j=1}^{n} w_j$ є непарним, то розв’язок, у якого

$$z_w^* = \left|\sum_{j=1}^{n} w_j z_j - \sum_{j=1}^{n} w_j (1 - z_j)\right| = 1,$$

є оптимальним.

Величина z_w^* – є нижньою межею значень цільової функції задачі.

Алгоритми розв’язання задачі

З урахуванням результатів огляду робіт, присвячених задачі SSP, доцільною є розробка евристичних та генетичних алгоритмів розв’язання поставленої задачі.

Евристичні алгоритми – це прості методи розв'язання оптимізаційних задач, які використовуються для отримання розв’язків за допомогою формальних правил, що базуються на використанні знань, досвіду та інтуїції розробника. До їх переваг відноситься простота та легкість реалізації.

Метаевристичні генетичні алгоритми (Genetic Algorithm) виникли на основі спостереження за природними процесами еволюції та селекції популяцій живих істот – особин певного виду – і моделювання їх принципів [14]. Вони належать до еволюційних пошукових алгоритмів, які використовуються для розв'язання задач оптимізації, моделювання тощо. Ці алгоритми включають в себе послідовний відбір, комбінування та варіації шуканих параметрів шляхом використання механізмів, що схожі на біологічну еволюцію. Алгоритми не гарантують отримання оптимального розв’язку, але їх ефективність можна значно підвищити шляхом налаштування

параметрів та/або операторів. До переваг відноситься невибагливість до виду та властивостей цільових функцій.

Для опису алгоритмів використовуються наступні величини:

$\mathbb{N}$ – множина усіх об’єктів;

$\mathbb{A}$ – множина об’єктів зони A;

$\mathbb{B}$ – множина об’єктів зони B;

w_A – сумарна зважена кількість об’єктів зони A;

w_B – сумарна зважена кількість об’єктів зони B;

$w_{AB} = |w_A - w_B|$.

*Евристичний алгоритм EA*1

Ідея розробленого евристичного алгоритму $EA1$ полягає в тому, що обирається точка C, яка ділить відрізок AB навпіл. Далі ітеративно перебираються точки, що відповідають об’єктам, і на кожній ітерації проводиться пряма через точку C та точку поточного об’єкта. На кожній ітерації визначається w_{AB} та порівнюється з поточним рекордним значенням ЦФ, за необхідності, рекорд оновлюється.

Схема алгоритму

Вхід:

$A(x_A, y_A),\ B(x_B, y_B), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

Вихід:

Рекордний розв’язок:

$\mathbb{A}^*$ – множина об’єктів зони A

$\mathbb{B}^*$ – множина об’єктів зони B

w_{AB}^* – рекордне значення ЦФ

$w_{AB}^* = \infty$ – початкове рекордне значення ЦФ

Розрахувати точку C – середину відрізку AB

for each об’єкту j з $\mathbb{N}$

Побудувати пряму, яка проходить через об’єкт (x_j, y_j) та точку C

Визначити множину $\mathbb{A}$ відносно цієї прямої

Визначити множину $\mathbb{B}$ відносно цієї прямої

Розрахувати $w_{AB} = |w_A - w_B|$

if $w_{AB} < w_{AB}^*$ **then**

0

$\mathbb{A}^* := \mathbb{A}$

1

$\mathbb{B}^* = \mathbb{B}$

2

 ISSN 1560-8956

$w_{AB}^{*} = w_{AB}$

3

endif

4

endfor

5

Складність алгоритму становить $O(n^2)$.

*Евристичний алгоритм EA*2

Цей алгоритм є рекурсивною процедурою виклику описаного вище евристичного алгоритму *EA*1. Порядок рекурсивного перебору наступний. Спочатку обирається точка C, яка ділить відрізок AB навпіл. Далі виконується процедура, що реалізує алгоритм *EA*1. Після цього задача розбивається на дві підзадачі і та ж процедура виконується для відрізків AC *та* CB. І так далі, поки не дійдемо до відрізків, довжини яких менші за 1.

Передбачається, що довжина відрізку AB більша за 1.

Схема евристичного алгоритму EA2

Вхід:

$A(x_A, y_A)$, $B(x_B, y_B)$, (x_1, y_1), (x_2, y_2), …, (x_n, y_n)

Вихід:

$\mathbb{A}^*$, $\mathbb{B}^*$, w_{AB}^{*}

$w_{AB}^{*} = \infty$ – початкове рекордне значення ЦФ

Procedure RA(A, B)

Procedure RA(*L*, *R*) // L – ліва точка відрізку; R – права точка відрізку

if $|LR| < 1$ **then**

return

endif

Розрахувати точку M – середину відрізку LR

for each об’єкту j з $\mathbb{N}$

Побудувати пряму, яка проходить через j та точку M

Визначити множину $\mathbb{A}$ відносно цієї прямої

Визначити множину $\mathbb{B}$ відносно цієї прямої

Розрахувати $w_{AB} = |w_A - w_B|$

if $w_{AB} < w_{AB}^{*}$ **then**

0

$\mathbb{A}^* := \mathbb{A}$

1

$\mathbb{B}^* := \mathbb{B}$

2

$w^*_{AB} = w_{AB}$

3

endif

4

endfor

5

Procedure RA(L, M) //пошук розв’язків на лівому відрізку

6

Procedure RA(M, R) // пошук розв’язків на правому відрізку

7

Складність алгоритму становить $O(ln^2)$, де l – довжина відрізку AB.

Генетичний алгоритм

Особливостями операторів розроблених генетичних алгоритмів є:

– *створення популяції* – початкова популяція не має повторів;

– *оновлення популяції* відбувається за правилом «відбір витисненням»: після додавання нащадка до популяції, з неї видаляється найгірша особина; при цьому уникнення виродженості популяції, на кожній ітерації популяція не повинна мати повторів;

– *вибір батьків* відбувається випадковим чином серед визначеного відсотка найкращих особин;

– *схрещення*: кожен з параметрів прямої нащадка є опуклою лінійною комбінацією відповідних параметрів прямих, що відповідають особинам-батькам;

– *мутація* нащадка: кожен з параметрів прямої нащадка змінюється на випадкове число із заданого діапазону (окіл мутації);

– *умова завершення:* розглядаються дві умови завершення:

  - після виконання заданої кількості ітерацій (*GA*1);
  - після того, як буде виконана задана кількість ітерацій, впродовж яких рекордний розв’язок не поліпшується (*GA2*).

Схема генетичного алгоритму GA1

Вхід:

$A(x_A, y_A), B(x_B, y_B), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

K – кількість ітерацій

I – кількість осіб в популяції

 ISSN 1560-8956

b – відсоток вибірки кращих осіб (частка кращих осіб)

p – ймовірність мутації

Δ – параметр, що описує розмір околу, в межах якого можуть мутувати параметри прямої нащадка

Вихід:

$\mathbb{A}^*$,$\mathbb{B}^*$, w_{AB}^*

$i := 0$

while $i < I$

Побудувати випадкову пряму, яка в довільній точці перетинає відрізок AB

Визначити множину $\mathbb{A}$ відносно цієї прямої

Визначити множину $\mathbb{B}$ відносно цієї прямої

Розрахувати $w_{AB} = |w_A - w_B|$ // значення фітнес-функції

if в популяції немає цієї особини **then**

Додати особину до популяції

0

$i := i + 1$

1

endif

2

endwhile

3

Визначити рекордний розв’язок в поточній популяції: $\mathbb{A}^*$, $\mathbb{B}^*$, w_{AB}^*

4

for $k := 1$ **to** K // вибір батьків, схрещення, мутація

5

Обрати $I \times b$ кращих осіб

6

Серед обраних кращих особин випадковим чином **обрати** дві

7 особини батьків

Створити нащадка, параметри прямої якого є опуклою лінійною

8 комбінацією відповідних параметрів прямих, що відповідають особинам-батькам (коефіцієнт комбінації визначається випадково)

З імовірністю p **виконати** мутацію нащадка шляхом додавання до

9 параметрів прямої випадкового числа, рівномірно розподіленого на інтервалі $[-\Delta;\ +\Delta]$

Оновити популяцію

0

За необхідності **оновити** рекордний розв’язок: $\mathbb{A}^*$, $\mathbb{B}^*$, w_{AB}^*

1

endfor

2

Загальна часова складність генетичного алгоритму становить $O(KI\ log(I))$.

Схема генетичного алгоритму GA2

Вхід:

$A(x_A, y_A)$, $B(x_B, y_B)$, (x_1, y_1), (x_2, y_2), …, (x_n, y_n)

k – максимальна кількість ітерацій, впродовж яких рекордний розв’язок не поліпшується

I – кількість осіб в популяції

b – відсоток вибірки кращих осіб (частка кращих осіб)

p – ймовірність мутації

Δ – параметр, що описує розмір околу, в межах якого можуть мутувати параметри прямої нащадка

Вихід:

$\mathbb{A}^*$,$\mathbb{B}^*$, w_{AB}^*

$i := 0$

while $i < I$

Побудувати випадкову пряму, яка в довільній точці перетинає відрізок AB

Визначити множину $\mathbb{A}$ відносно цієї прямої

Визначити множину $\mathbb{B}$ відносно цієї прямої

Розрахувати $w_{AB} = |w_A - w_B|$ // значення фітнес-функції

if в популяції немає цієї особини **then**

Додати особину до популяції

0

$i := i + 1$

1

endif

2

endwhile

3

Визначити рекордний розв’язок в поточній популяції: $\mathbb{A}^*$, $\mathbb{B}^*$, w_{AB}^*

4

 ISSN 1560-8956

while впродовж k послідовних ітерацій w_{AB}^{*} не поліпшується

Обрати $I \times b$ кращих осіб

Серед обраних кращих особин випадковим чином **обрати** дві особини батьків

Створити нащадка

З імовірністю p **виконати** мутацію нащадка

Оновити популяцію

За необхідності **оновити** рекордний розв’язок: $\mathbb{A}^{*}$, $\mathbb{B}^{*}$, w_{AB}^{*}

endwhile

Дослідження розроблених алгоритмів

На рис. 3-4 наведено результати роботи розроблених алгоритмів для деякої індивідуальної задачі з $n = 20$.

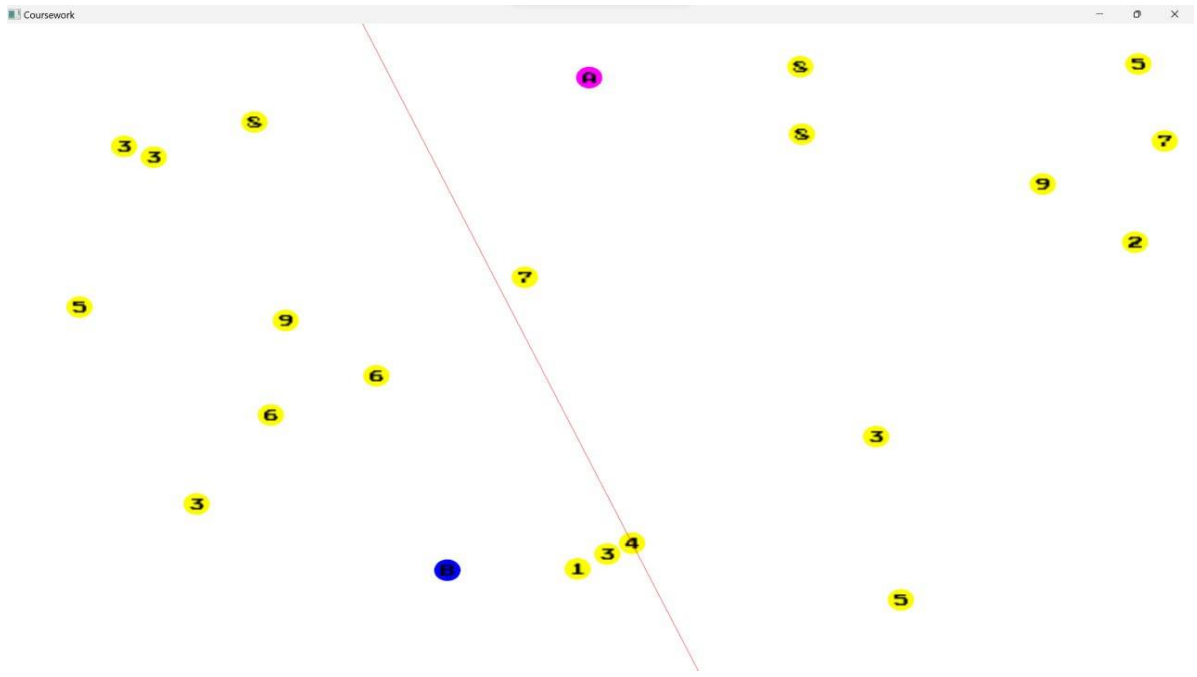


Рис. 3. Результат роботи алгоритму *EA*1 та *GA*1. Значення критерія 3

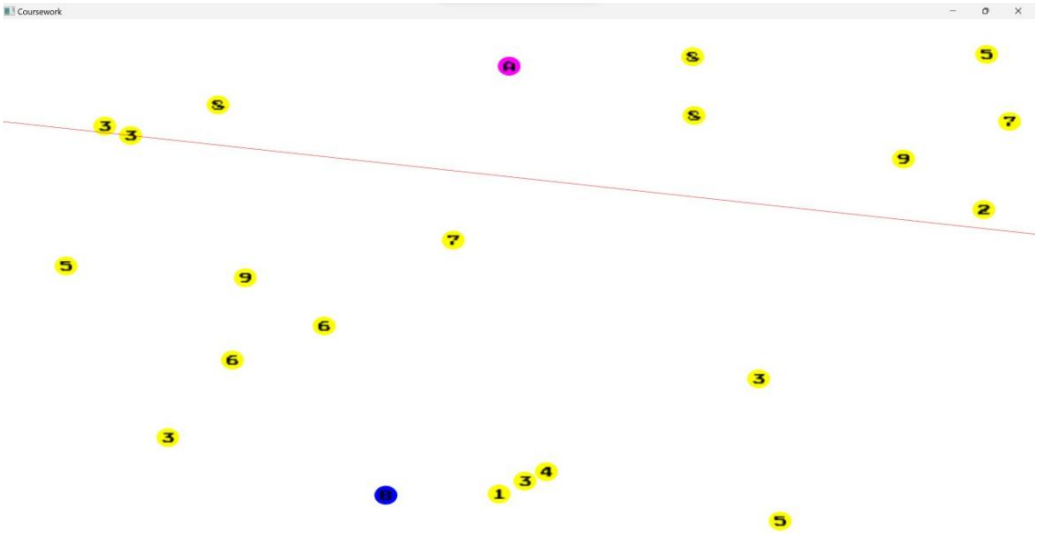


Рис. 4. Результат роботи алгоритму *EA*2 та *GA*2. Значення критерія 1

Під час експериментів розв’язувались задачі розмірностей $n \in\in \{10, 50, 100, 150, 200, 250\}$. Для кожної розмірності генерувалось по 75 індивідуальних задач, у яких W=10. Кожна із задач розв’язувалась кожним із 4-х алгоритмів. Генетичні алгоритми мали такі параметри: $K = 200, k = 100,\ \Delta = 1$.

Для кожної індивідуальної задачі визначався час роботи кожного з алгоритмів та відхилення від нижньої межі значень цільової функції z_w^*.

Графік залежності часу роботи алгоритмів від кількості об’єктів наведено на рис. 5.

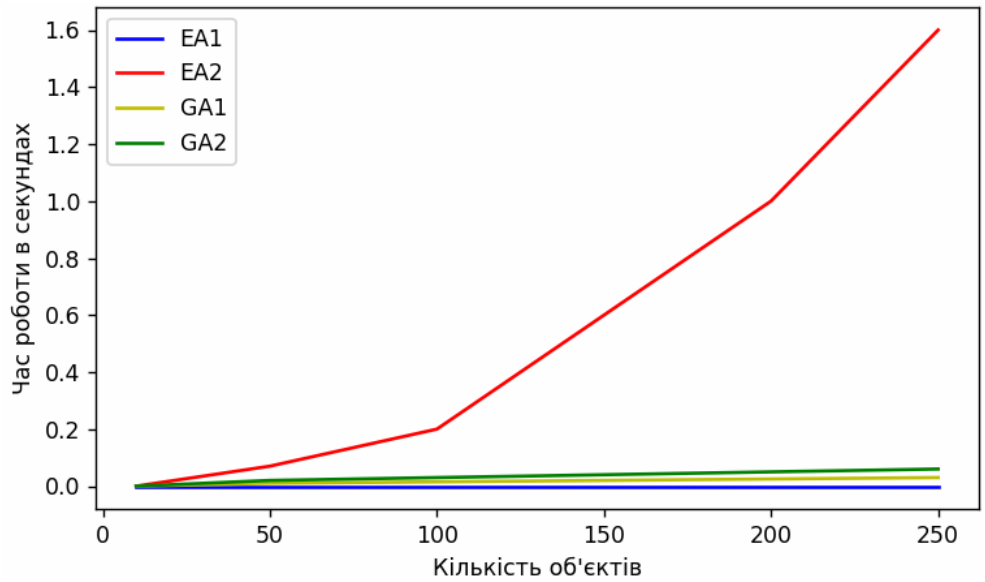


Рис. 5. Графік залежності часу роботи алгоритмів від кількості об’єктів

 ISSN 1560-8956

Графік залежності середньої похибки від кількості об’єктів наведено на рис. 6 (під похибкою розуміється середнє відхилення значення цільової функції від її нижньої границі z_w^*).

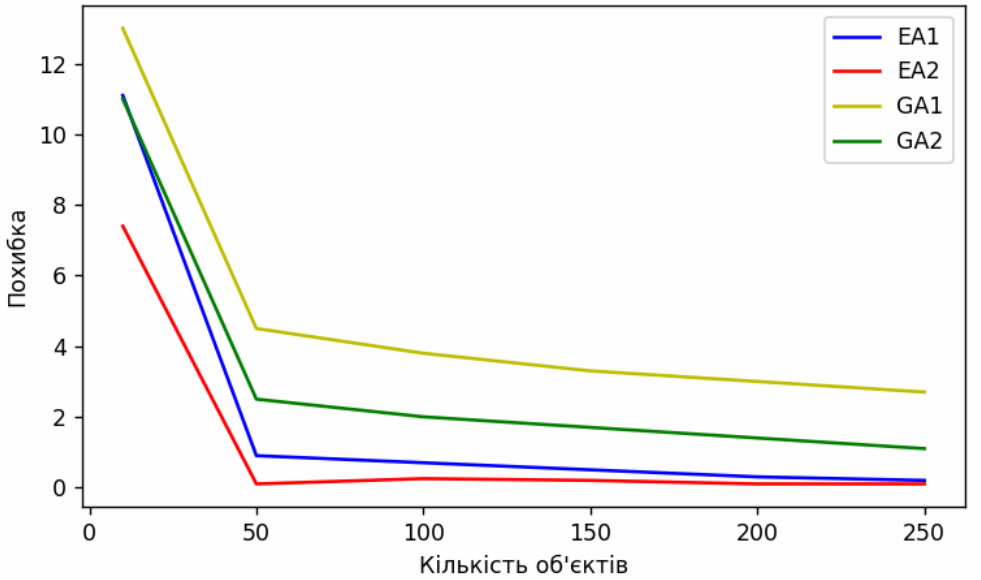


Рис 6. Графік залежності середньої похибки від кількості об’єктів

Відповідно до теоретичної оцінки трудомісткості розроблених алгоритмів, алгоритм *EA*1 виявився найшвидшим серед чотирьох, а рекурсивний *EA*2 – навпаки, найповільнішим. При цьому швидкість алгоритмів *EA*1, *GA*1 та *GA*2 є приблизно однаковою. На відміну від інших, час роботи *EA*2, суттєво збільшується з ростом кількості об’єктів.

Щодо точності, то найкращі результати показав алгоритм *EA*2. Це пояснюється тим, що завдяки використанню рекурсії алгоритм досліджує велику кількість достатньо щільно розташованих прямих, що значно підвищує ймовірність отримання оптимального чи близького до нього розв’язку. На задачі маленької розмірності усі алгоритми показали достатньо велике відхилення від нижньої межі z_w^* значень цільової функції.

Відмітимо, що реальне відхилення значень цільової функції, отриманого кожним з алгоритмів, від невідомого оптимального може бути значно меншим.

Висновки

З урахуванням результатів експериментів, для розв’язання задачі розмірності до 200 об’єктів доцільно використовувати рекурсивний евристичний алгоритм, бо час його роботи залишається прийнятним, а точність – найкращою серед представлених алгоритмів. Підвищення точності генетичних алгоритмів вимагає подальших експериментів з підбору їх параметрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Papadimitriou C.H.* Computational Complexity. – Massachusetts: Addison-Wesley, 1994. – 523 p.

2. *Parque V.* Tackling the Subset Sum Problem with Fixed Size using an Integer Representation Scheme // IEEE Congress on Evolutionary Computation (CEC), Poland, 2021, pp. 1447-1453.

3. *Müller T.* Solving Set Partitioning Problems with Constraint Programming // Computer Science, 2009, pp.313-332.

4. *Ozerov I.* Combinatorial Algorithms for Subset Sum Problems. – Ruhr-Universität Bochum, Diss., 2016. – 128 p.

5. *Aarts E., Lenstra J.K.* Local search in combinatorial optimization. – Princeton University Press, 2003. – 528 p.

6. *Madugula M.K., S.K. Majhi, Panda N.* An Efficient Arithmetic Optimization Algorithm for Solving Subset-sum Problem // International Conference on Connected Systems & Intelligence (CSI), 31 August 2022 - 02 September 2022, India. – IEEE. DOI: 10.1109/CSI54720.2022.9923996

7. *Wang R.L.* A genetic algorithm for subset sum problem // Neurocomputing, vol. 57, 2004, pp. 463-468.

8. *Ghosh D., Chakravarti N.* A competitive local search heuristic for the subset sum problem // Computers & Operations Research, vol. 26, Issue 3, 1999, pp. 271-279.

9. *Curtis V.V., Sanches C.A.A.* An improved balanced algorithm for the subset-sum problem // European Journal of Operational Research, vol. 275, no. 2, 2019, pp. 460-466.

10. *Harsha K., Jeyakumar S.G.* Comparison of Dynamic Programming and Genetic Algorithm Approaches for Solving Subset Sum Problems // Computational Vision and Bio-Inspired Computing, 2020, pp. 472-479.

11. *Gao X., Yu W.* An algorithm for subset sum problem on molecular computation // 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2016, pp. 1072-1076.

12. *Singh S., Srivastava S.* Review of Clustering Techniques in Control System: Review of Clustering Techniques in Control System // Procedia Computer Science, vol. 173, 2020, pp. 272-280.

13. *Дубовик В.П., Юрик І.І.* Вища математика: навч. посібн. – К.: “А.С.К.”, 2006. – 648 с.

14. *Гуляницький Л.Ф. Мулеса О.Ю.* Прикладні методи комбінаторної оптимізації: навч. посібн. – Київ: ВПЦ “Київський університет”, 2016. – 142 с.