DOI 10.36074/logos-21.06.2024.032

МЕТОДИ ЗБАЛАНСОВАНОГО ЗАКРІПЛЕННЯ ОБ’ЄКТІВ ЗА ЗОНАМИ ВІДПОВІДАЛЬНОСТІ

Сидоров Микита Олександрович[1]
Наукові керівники: Сперкач Майя Олегівна[2]**, Жданова Олена Григорівна**[3]

1. здобувач вищої освіти факультету інформатики та обчислювальної техніки
НТУУ «Київський політехнічний інститут імені Ігоря Сікорського», УКРАЇНА
ORCID ID: 0009-0001-5933-6946

2. канд. техн. наук, доцент, доцент кафедри інформаційних систем та технологій
НТУУ «Київський політехнічний інститут імені Ігоря Сікорського», УКРАЇНА
ORCID ID: 0000-0002-4300-5106

3. канд. техн. наук, доцент, доцент кафедри інформаційних систем та технологій
НТУУ «Київський політехнічний інститут імені Ігоря Сікорського», УКРАЇНА
ORCID ID: 0000-0002-8787-846X

Ефективне закріплення об'єктів за зонами відповідальності є важливим для багатьох галузей, таких як логістика, охорона здоров'я та урбаністика. Сучасні умови вимагають оптимізації розміщення об'єктів для забезпечення ефективного використання ресурсів та мінімізації витрат. Важливість оптимізації закріплення об'єктів зростає завдяки необхідності врахування багатьох параметрів та критеріїв для забезпечення точних і ефективних рішень. Сучасні оптимізаційні методи дозволяють вирішувати складні задачі закріплення, адаптуючись до швидкозмінних умов і великої кількості даних. Це сприяє вдосконаленню методів управління, що є актуальним у сучасному світі.

Робота присвячена оптимізаційній задачі, яка передбачає розбиття множини об’єктів (точок) із відомими координатами та вагами на чотири підмножини, кожній з яких буде відповідати певний ключовий пункт (КП), за критерієм мінімізації різниці сукупних вагових значень елементів між утвореними секторами.

Досліджувана задача включає елементи двох класичних проблем: задача про суму підмножини (SSP) і задача багатошляхового розбиття (MPP).

У задачі суми підмножини (Subset Sum Problem) задано скінченний набір $S\ \{x_1, x_2, \ldots, x_n\}$ натуральних чисел і ціле число $t > 0$. Така задача відповідає на питання, чи існує підмножина $S' \subseteq S$, сума елементів якої дорівнює t. Також ця задача може розглядатися як задача оптимізації, де потрібно знайти підмножину, сума якої є максимально можливою, але не перевищує t. Ця задача є NP-повною [1], тобто не існує відомих поліноміальних алгоритмів для її вирішення у загальному випадку. Для її розв'язання існує декілька підходів.

Експоненційний алгоритм є найпростішим підходом, що передбачає перебір всіх підмножин множини S, обчислення їх сум і порівняння з t [2]. Цей метод має експоненційну складність $O(2^n n)$, що робить його менш ефективним для великих n.

Поліноміальний апроксимаційний алгоритм використовує методи динамічного програмування, які дозволяють знаходити наближені рішення за поліноміальний час [2]. Такі алгоритми корисні для великих задач, де точне рішення є надто дорогим.

Задача про багатошляхове розбиття (Multi-way Partition Problem) є класичною задачею, що стосується розбиття набору точок на кілька підмножин таким чином, щоб різниця між сумами ваг у підмножинах була мінімальною [3]. У цьому також є аспект оптимізації: пошук розбиття підмножини S на k підмножин, де суми максимально збалансовані. До цієї мети оптимізації можна підійти різними способами.

Мінімізація різниці між найбільшою та найменшою сумою. Ця мета часто згадується в роботах, пов'язаних з багатошляховим розбиттям чисел, і знаходить застосування в різних областях [4].

Мінімізація найбільшої суми. Ця мета схожа на планування ідентичних завдань на кількох процесорах, щоб мінімізувати загальний час завершення, відомий як “makespan”.

Максимізація найменшої суми. Ця мета актуальна в таких сценаріях, як “справедливий” розподіл використання предметів, наприклад, підтримка ефективності роботи модульних систем, таких як авіаційні двигуни [5].

Ці цілі збігаються для $k = 2$, але розходяться для $k \geq 3$. Хоча відомо, що ці проблеми є NP-складними, існують ефективні алгоритми, які пропонують рішення для багатьох практичних випадків [6].

Конкретним прикладом проблеми MPP є двошляхове розбиття, де множину ділять на дві підмножини з мінімальною різницею сум ваг. Для вирішення цієї задачі використовуються різні методи.

Один з найпростіших методів, що має назву включення-виключення, передбачає обхід бінарного дерева в глибину, де кожний рівень відповідає різному елементу. Кожний вузол включає елемент на лівій гілці та виключає його на правій гілці. Кінцеві вузли представляють повні підмножини [7].

Підхід, що називається повний жадібний алгоритм, знаходить кращі розв’язки швидше за попередній. Він працює шляхом сортування цілих чисел у порядку спадання та поетапного присвоєння їх до результуючої підмножини на кожному рівні, досліджуючи те саме дерево рішень, що і метод включення-виключення, але змінює порядок гілок [8].

Більш ефективним є повний алгоритм Кармаркара-Карпа, заснований на евристичному наближенні, що розташовує цілі числа в порядку зменшення і на кожному кроці замінює два найбільших цілих числа їхньою різницею [9].

Горовіц і Сахні (HS) запропонували швидший підхід для вирішення проблеми суми підмножини [10]. Їх метод передбачає розбиття заданої множини з n чисел на дві підмножини, після чого генеруються всі можливі підмножини для кожної половини множини.

Шроппел і Шамір (SS) запропонували [11] більш ефективний варіант алгоритму HS. У той час як метод Горовіца і Сахні покладається на попереднє створення, зберігання та сортування підмножин із списків, SS генерує підмножини за потреби та сортує їх на основі сум.

Задачі розбиття множини на підмножини мають різні підходи для вирішення, включаючи експоненційні алгоритми, поліноміальні апроксимаційні методи та спеціалізовані евристики. Для кожного типу задачі використовуються відповідні методи, які враховують її особливості.

Змістовна постановка задачі. Припустимо, існує певна територія з n визначеними об'єктами, кожен з яких має відомі координати: (x_1, y_1), (x_2, y_2), … , (x_n, y_n). Для кожного конкретного об’єкта j зазначено коефіцієнт значимості (вага) w_j, що знаходиться в межах від 1 до W. Визначаються також чотири ключових пункти (КП), кожен з яких має відомі координати: $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$ та $D(x_D, y_D)$.

Завдання полягає у розбитті території на чотири сектори таким чином, щоб кожен сектор був асоційований з відповідним ключовим пунктом (A, B, C, D), а різниця сумарних ваг об’єктів у різних секторах була мінімальною. Отже, завдання зводиться до задачі, в якій потрібно побудувати дві прямі, як границі чотирьох секторів, з метою досягнення рівномірного розподілу значимості об’єктів між секторами.

У даній задачі важливо, щоб ваги об'єктів у секторах були якомога більш рівномірними. Оскільки стандартне відхилення є мірою розкиду значень від середнього, його мінімізація дозволяє досягти рівномірного розподілу ваг.

Розглянемо деяку індивідуальну задачу і для неї наведемо різні припустимі розв’язки. Рисунок 1 демонструє приклад задачі з $n = 16$ (вага кожного елементу позначена всередині відповідного кола).

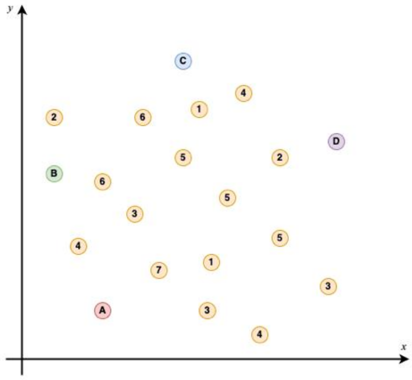
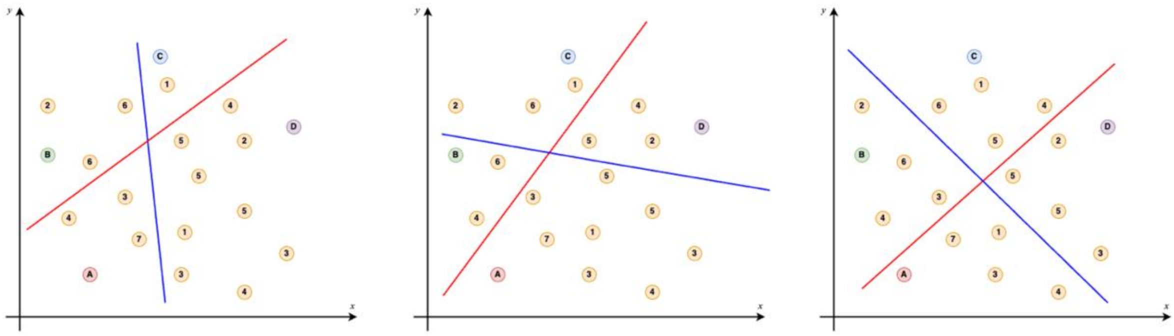


Рис. 1. **Приклад задачі з $n = 16$**

На рисунку 2 представлено три варіанти розміщення прямих для поділу на 4 сектори.



а) б) в)

Рис. 2. **Варіанти розташування розмежувальних прямих**

Кожному з цих варіантів відповідають наступні значення критерію:

– варіант а): A = 14 ; B = 14 ; C = 1 ; D = 32.

$$\text{ЦФ} = stdev([14,\ 14,\ 1,\ 32]) \approx 12.738;$$

– варіант б): A = 31 ; B = 10 ; C = 9 ; D = 11.

$$\text{ЦФ} = stdev([31,\ 10,\ 9,\ 11]) \approx 10.532;$$

– варіант в): A = 15 ; B = 15 ; C = 16 ; D = 15.

$$\text{ЦФ} = stdev([15,\ 15,\ 16,\ 15]) = 0.5.$$

При варіанті розміщення прямих “в” цільова функція досягає свого мінімального значення 0.5, що робить цей розв'язок оптимальним для даного прикладу. Це пояснюється тим, що ЦФ має найнижче можливе значення стандартного відхилення сумарних ваг об'єктів у різних секторах при непарній сумарній вазі всіх об’єктів площини, за умови, що мінімальна вага об’єкта дорівнює 1, а вага об’єкта є цілим числом.

Математична модель. Знайти коефіцієнти m_1, b_1 для першої прямої та m_2, b_2 для другої прямої, за яких досягає мінімуму функція *(1)*:

$$\sqrt{\frac{\sum_{\mathbf{k}\in\{\mathbf{A,B,C,D}\}}(\mathbf{w_k} - \overline{\mathbf{w}})^2}{\mathbf{4}}} \rightarrow \mathbf{min}, \quad (1)$$

при обмеженнях *(2)*:

$$\begin{gathered} \begin{cases} \boldsymbol{y = m_1 x + b_1} \\ \boldsymbol{y = m_2 x + b_2} \end{cases} \neq \emptyset, \\ \mathrm{y_0 - (m_1 x_0 + b_1) \neq 0}, \\ \mathrm{y_0 - (m_2 x_0 + b_2) \neq 0}, \\ index_{k\in\{A,B,C,D\}} = \{0, 1, 2, 3\}, \\ index_A, index_B, index_C, index_D, index_1, \dots, index_n \in \{0,1,2,3\}, \end{gathered} \quad (2)$$

де

$$\begin{gathered} \mathbf{k} = \{\,\mathbf{A, B, C, D}\,\}, \\ \mathrm{j} = \{\,1, \dots, \mathrm{n}\,\}, \\ \mathrm{s_k} = \begin{pmatrix} \mathrm{s_{k_1}} \\ \mathrm{s_{k_2}} \end{pmatrix} = \begin{pmatrix} \begin{cases} 1, \text{якщо } (\mathrm{y_k} - (\mathrm{m_1 x_k + b_1}) > 0) \\ 0, \text{якщо } (\mathrm{y_k} - (\mathrm{m_1 x_k + b_1}) < 0) \end{cases} \\ \begin{cases} 1, \text{якщо } (\mathrm{y_k} - (\mathrm{m_2 x_k + b_2}) > 0) \\ 0, \text{якщо } (\mathrm{y_k} - (\mathrm{m_2 x_k + b_2}) < 0) \end{cases} \end{pmatrix}, \\ \mathrm{s_k} = \begin{pmatrix} \mathrm{s_{k_1}} \\ \mathrm{s_{k_2}} \end{pmatrix} = \begin{pmatrix} \begin{cases} 1, \text{якщо } (\mathrm{y_k} - (\mathrm{m_1 x_k + b_1}) > 0) \\ 0, \text{якщо } (\mathrm{y_k} - (\mathrm{m_1 x_k + b_1}) < 0) \end{cases} \\ \begin{cases} 1, \text{якщо } (\mathrm{y_k} - (\mathrm{m_2 x_k + b_2}) > 0) \\ 0, \text{якщо } (\mathrm{y_k} - (\mathrm{m_2 x_k + b_2}) < 0) \end{cases} \end{pmatrix}, \\ \mathrm{index_k, index_j} = (\mathrm{s_{k_0}} * 2^1 + \mathrm{s_{k_1}} * 2^0, \mathrm{s_{j_0}} * 2^1 + \mathrm{s_{j_1}} * 2^0), \\ \mathrm{w_k} = \sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{w_j} * \begin{cases} 1, \text{якщо } \mathrm{index_j = index_k} \\ 0, \text{якщо } \mathrm{index_j \neq index_k}' \end{cases} \\ \overline{\mathrm{w}} = \frac{\sum_{\mathrm{k}\in\{\mathrm{A,B,C,D}\}} \mathrm{w_k}}{4}. \end{gathered} \quad (3)$$

P – будь-яка точка (КП або об'єкт) з координатами (x_0, y_0).

Алгоритми розв'язання задачі. Враховуючи результати огляду робіт, що стосуються задачі SSP та MPP, розробка евристичних та метаевристичних алгоритмів вирішення поставленої задачі, таких як жадібний алгоритм (Greedy Algorithm) і алгоритм імітації відпалу (Simulated Annealing), є доцільною.

Жадібні алгоритми представляють собою прості конструктивні методи для розв'язання оптимізаційних задач, де розв'язки отримуються шляхом прийняття локально-оптимальних рішень на кожному етапі.

Алгоритм імітації відпалу є метаевристичним методом, який базується на моделюванні процесу відпалу металів. Він використовується для розв'язання задач оптимізації шляхом імітації поступового охолодження матеріалу, при якому атоми досягають стану з мінімальною енергією. Переваги цього методу включають здатність уникати локальних мінімумів та можливість налаштування параметрів для покращення ефективності алгоритму. Алгоритм імітації відпалу є гнучким і може бути ефективним у різних умовах задачі, що робить його корисним інструментом для розв'язання складних оптимізаційних задач.

Жадібний алгоритм (GA)

Ідея розробленого жадібного алгоритму для даної задачі полягає у початковій ініціалізації прямих, які розділяють площину на сектори, шляхом перебору усіх можливих пар точок між ключовими пунктами. Потім обираються ті пари прямих, які утворюють чотири сектори з найкращим значенням ЦФ. Далі алгоритм ітеративно покращує початковий розв'язок, переміщуючи прямі таким чином, щоб мінімізувати значення ЦФ. На кожній ітерації алгоритм розглядає можливі переміщення прямих та обирає ту, яка найбільше зменшує значення цільової функції. Якщо на попередньому кроці не було здійснено переміщень прямих, алгоритм завершує роботу. У іншому випадку процес продовжується до досягнення локально-оптимального розв'язку.

Алгоритм імітації відпалу (SA)

Стандартні параметри алгоритму імітації відпалу включають такі параметри, як початкова температура, кінцева температура, швидкість охолодження та максимальна кількість ітерацій. Додатково, для розв'язання задачі були введені методи збурення, які допомагають алгоритму ефективніше досліджувати простір розв'язків. Налаштування цих параметрів дозволяє забезпечити баланс між швидкістю виконання та точністю розв'язку, що є ключовим аспектом успішного застосування алгоритму.

Для цієї задачі були введені такі методи збурення:

– метод обертання (*rotate*): обертання прямих на випадковий кут до 10 градусів.

– метод зсуву (*shift*): зсув прямих на випадкову величину в діапазоні від -0.15 до 0.15.

– метод випадкових змінних (*random*): внесення випадкових змін у коефіцієнти прямих за допомогою нормальних розподілень.

– комбінований метод (*combined*): поєднання методів обертання, зсуву та випадкових змін для адаптування алгоритму до умов задачі.

Ідея алгоритму імітації відпалу полягає у початковій ініціалізації прямих, що відбувається аналогічно до початкової ініціалізації прямих в GA. Далі визначається початкова температура. Потім алгоритм ітеративно покращує початковий розв'язок, переміщуючи випадковим чином прямі для мінімізації ЦФ. На кожній ітерації алгоритм розглядає можливі переміщення прямих та приймає або відхиляє новий розв'язок на основі температури. Цей процес повторюється до тих пір, поки не буде досягнуто максимальної кількості ітерацій або температура не знизиться до мінімального значення.

Порівняння розроблених алгоритмів. Метою експерименту є порівняння ефективності розроблених алгоритмів в залежності від кількості об’єктів.

У цій роботі для порівняння був використаний підхід повного перебору всіх можливих пар прямих, що розділяють площину, для знаходження оптимального розв’язку. Цей метод надає еталонні значення, з якими можна порівнювати результати інших, менш обчислювально затратних методів.

Для проведення дослідження було згенеровано по 100 задач для кожного з п’яти рівнів розмірності (10, 20, 40, 100, 200 об'єктів). Це дозволить провести детальний аналіз та порівняння ефективності алгоритмів в різних умовах.

Для оцінки якості розроблених алгоритмів було розроблено метод повного перебору, який знаходить оптимальні розв'язки задач. Однак через його високу трудомісткість, цей метод був використаний лише для задач з $n \leq 200$.

Далі для кожного використаного алгоритму визначаються наступні характеристики за результатами розв’язаної сотні задач для певної розмірності:

– середнє відхилення значення цільової функції від оптимального значення;

– найгірше відхилення значення цільової функції від оптимального значення;

– час роботи алгоритму (мінімальний, середній, максимальний) у мілісекундах;

– відсоток знайдених оптимальних розв’язків.

При порівнянні ефективності (точність та відсоток знайдених оптимальних розв’язків) і швидкодії жадібного алгоритму з алгоритмом імітації відпалу, для останнього було обрано по дві найкращі комбінації параметрів, які формуються з використанням значень, зазначених у таблиці 1, для задач кожної розмірності.

Таблиця 1

Значення параметрів алгоритму імітації відпалу

Назва параметру	*Значення параметру*
Початкова температура (it)	$10^3, 10^4, 10^5, 10^6$
Кінцева температура (ft)	$1 * 10^{-6}, 1 * 10^{-7}, 1 * 10^{-8}, 1 * 10^{-9}, 1 * 10^{-10}$
Швидкість охолодження (cr)	0.85, 0.9, 0.95, 0.99
Максимальна кількість ітерацій (mi)	$10^3, 10^4, 10^5, 10^6$
Метод збурення	Random, Rotate, Shift, Combined

[авторська розробка]

Отримані результати експерименту наведені у таблиці 2 та рисунках 3-5.

Таблиця 2

Результати експерименту

Кількість об'єктів	Алгоритм	Відхилення значення ЦФ від оптимального		Час роботи (мілісекунди)			Оптимальний розв'язок (%)
		Серед.	Найгір.	Мін.	Серед.	Макс.	
10	GA	0.42	3.787	4.668	14.97	40.414	69
	SA-it100000-ft1e-07-cr0.99-mi100000-combined	0.69	6.987	281.74	372.43	759.70	64
	SA-it1000000-ft1e-07-cr0.99-mi10000-combined	0.704	6.288	297.51	401.8	766.25	64
20	GA	0.697	5.406	16.215	69.601	208.46	56
	SA-it1000000-ft1e-08-cr0.99-mi100000-combined	0.801	6.694	367.45	491.372	751.81	55
	SA-it1000000-ft1e-09-cr0.99-mi10000-combined	0.824	5.496	348.40	532.86	768.67	50
40	GA	1.646	13.28	110.02	330.083	851.75	39
	SA-it1000000-ft1e-09-cr0.99-mi10000-combined	1.144	10.738	491.29	741.628	1051.9	41
	SA-it1000000-ft1e-09-cr0.99-mi1000000-random	1.264	10.173	408.64	624.812	947.24	42
100	GA	3.297	31.406	1057.1	3654.48	16690	24
	SA-it100000-ft1e-09-cr0.99-mi100000-combined	3.26	49.036	880.00	1290.05	1823.7	20
	SA-it1000000-ft1e-08-cr0.99-mi100000-random	3.292	24.499	679.25	1094.15	2491.4	15
200	GA	7.128	81.378	7234.0	27497.2	77272	21
	SA-it1000-ft1e-07-cr0.99-mi10000-random	5.228	62.691	698.11	1151.89	1660.8	12
	SA-it10000-ft1e-10-cr0.99-mi100000-combined	5.458	66.587	1200.3	1878.72	2971.5	18

[авторська розробка]

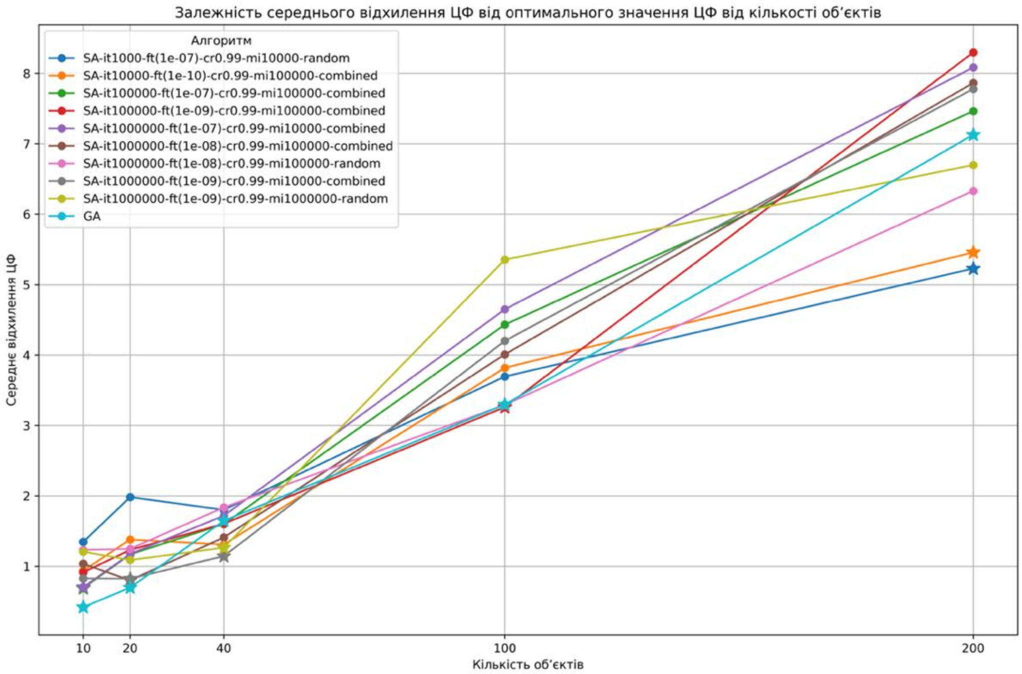


Рис. 3. **Залежність середнього відхилення ЦФ від кількості об'єктів**

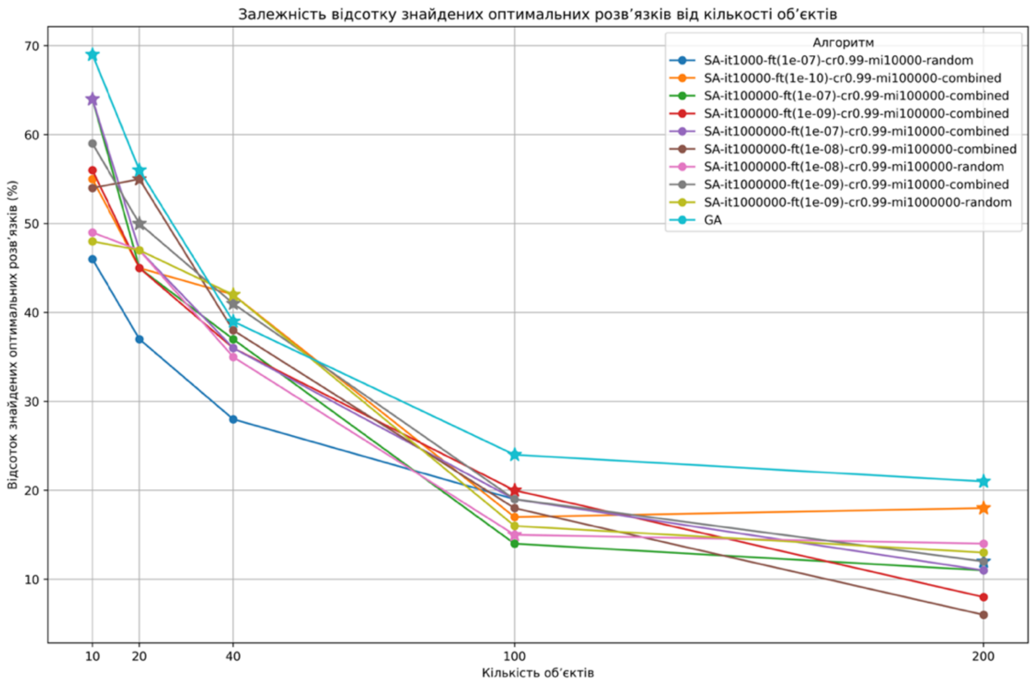


Рис. 4. **Залежність відсотку знайдених оптимальних розв'язків від кількості об'єктів**

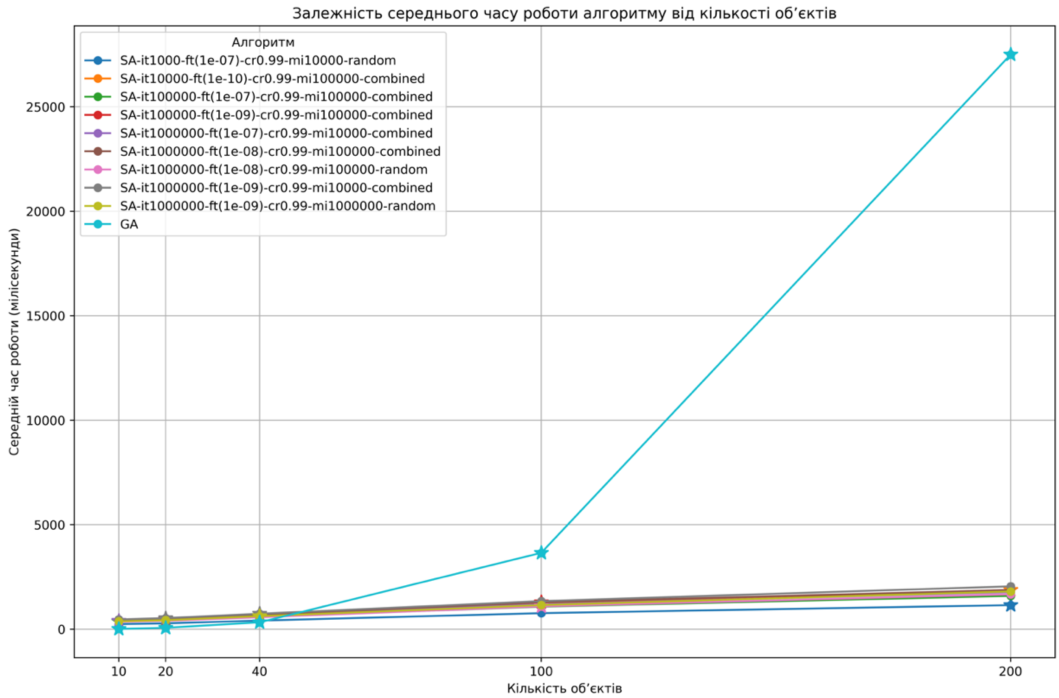


Рис. 5. **Залежність середнього часу роботи алгоритму від кількості об’єктів**

Висновки. Аналіз результату експерименту показує, що для задач з невеликою кількістю об'єктів (10-20) жадібний алгоритм є найефективнішим. Він відзначається найменшим середнім відхиленням значення ЦФ від оптимального значення цільової функції та найшвидшим часом роботи. Крім того, жадібний алгоритм знаходить більший відсоток оптимальних розв'язків порівняно з алгоритмом імітації відпалу, зокрема, 69% та 56% оптимальних розв'язків для задач з 10 та 20 об'єктами відповідно. Однак, зі збільшенням кількості об'єктів (40 і більше), жадібний алгоритм починає втрачати ефективність, а алгоритм імітації відпалу демонструє більш стабільні результати для великих задач, маючи менше середнє відхилення значення ЦФ від оптимального значення цільової функції і значно коротший час роботи. Таким чином, вибір між розробленими алгоритмами залежить від розміру задачі та балансу між точністю, часом роботи та ймовірністю знаходження оптимального розв'язку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

[1] Garey, M. R., & Johnson, D. S. (1979). Computers and intractability a guide to the theory of NP-completeness garey, M. R.; Johnson, D. S. Freeman & Company, New York.

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. The MIT Press.

[3] Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. SIAM Journal on Applied Mathematics, 17(2), 416–429. https://doi.org/10.1137/0117039
[4] Mertens, S. (2005). The easiest hard problem: Number partitioning. Computational Complexity and Statistical Physics. https://doi.org/10.1093/oso/9780195177374.003.0012
[5] Friesen, D. K., & Deuermeyer, B. L. (1981). Analysis of Greedy Solutions for a replacement part sequencing problem. Mathematics of Operations Research, 6(1), 74–87. https://doi.org/10.1287/moor.6.1.74
[6] Walter, R. (2011). Comparing the minimum completion times of two longest-first scheduling-heuristics. Central European Journal of Operations Research, 21(1), 125–139. https://doi.org/10.1007/s10100-011-0217-4
[7] Korf, R.E., Schreiber, E.L., & Moffitt, M.D. (2014). Optimal Sequential Multi-Way Number Partitioning. International Symposium on Artificial Intelligence and Mathematics.
[8] Korf, R.E. (2009) Multi-way number partitioning. In Proceedings of the 21st International Joint Conference on Artificial Intelligence.
[9] Karmarkar, N. & Karp, R. (1982) The differencing method of set partitioning. Computer Science Division (EECS).
[10] Horowitz, E., & Sahni, S. (1974). Computing partitions with applications to the Knapsack problem. Journal of the ACM, 21(2), 277–292. https://doi.org/10.1145/321812.321823
[11] Shamir, A. & Schroeppel, R. (1981) A T=O(2n/2), S=O(2n/4) algorithm for certain NP-complete problems. SIAM Journal of Computing, 10(3), 456–464. https://doi.org/10.1137/0210033

