*DOI: 10.25140/2411-5363-2025-1(39)-134-150*
*УДК 004.02:519.854*

***Олена Григорівна Жданова[1], Людмила Віталіївна Рибачук[2], Вероніка Сергіївна Рябчун[3], Тарас Васильович Малярчук[4]***

[1] кандидат технічних наук, доцент, доцент кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» (Київ, Україна)
**E-mail:** zhdanova.elena@hotmail.com. **ORCID:** https://orcid.org/0000-0002-8787-846X
**ResearcherID:** K-1193-2016. **Scopus Author ID:** 57208339441

[2] кандидат фізико-математичних наук, доцент, доцент кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» (Київ, Україна)
**E-mail:** rybachuk.liudmyla@lll.kpi.ua. **ORCID:** https://orcid.org/0000-0002-7652-587X
**ResearcherID:** AAE-7644-2019. **Scopus Author ID:** 57226545637

[3] студентка кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» (Київ, Україна)
**E-mail:** nikakyiv696@gmail.com

[4] студент кафедри інформаційних систем та технологій
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» (Київ, Україна)
**E-mail:** tarasmalarcuk8@gmail.com

# ЗАДАЧА МАКСИМІЗАЦІЇ ЗВАЖЕНОЇ КІЛЬКОСТІ ОБ'ЄКТІВ, ОБСТЕЖЕНИХ ДРОНОМ

*У сучасних умовах розвитку безпілотних технологій актуальними є задачі планування траєкторій дронів з максимальним охопленням об'єктів. Задача полягає в побудові прямої, яка перетинає найбільш значущі об'єкти на площині. Через неперервність простору прямих і дискретність розташування цілей застосовано дискретизацію. Метою роботи є розробка та порівняльний аналіз трьох підходів: жадібного алгоритму, часткового перебору секторів і евристичного методу. Проведено математичне обґрунтування, визначено критерії охоплення об'єктів, оцінено складність і ефективність методів. Результати показали: жадібний алгоритм демонструє найменший час виконання, але є менш точним; евристичний забезпечує баланс між точністю й швидкістю; частковий перебір – найточніший, але найповільніший. Задача може бути використана як компонент у складніших моделях планування місій дронів.*

***Ключові слова:*** *дрон; БПЛА; максимізація кількості обстежених об'єктів; планування траєкторії; дискретизація; комбінаторна оптимізація; жадібний алгоритм; частковий перебір; евристичний метод.*

*Рис.: 10. Табл.: 3. Бібл.: 16.*

**Актуальність теми дослідження**. Сучасні військові та цивільні операції дедалі частіше вимагають ефективного застосування дронів для обстеження або ураження цільових об'єктів, витрачаючи при цьому мінімальні ресурси. У військовій сфері це передусім питання оперативного планування й зменшення ризиків, а в цивільному контексті може стосуватися моніторингу великих територій — наприклад, для охорони кордонів, пошуково-рятувальних операцій чи аграрного сектору. В обох випадках завдання максимальної кількості обстежених дронами об'єктів має високу практичну цінність. Водночас проблема ускладнюється обмеженістю часу, ресурсів (наприклад, запасів енергії дрона) та геометричними аспектами його руху.

**Постановка проблеми.** Задача відноситься до класу важкорозв'язуваних оптимізаційних проблем, які неможливо розв'язати за прийнятний час методами перебору, коли розмірність задачі сягає сотень чи тисяч об'єктів. Тож виникає потреба в розробленні ефективних алгоритмічних підходів, здатних за обмежений час знаходити якісні (оптимальні або наближені до них) розв'язки. Дослідження ефективності різних алгоритмів, а також порівняння їхньої продуктивності, точності та масштабованості дозволять обрати найкращий метод під конкретні вимоги, зокрема: швидкодію, ступінь точності, критичність завдання тощо.

**Аналіз останніх досліджень і публікацій.** Проблема оптимізації траєкторії дронів у контексті максимального зваженого покриття об'єктів отримала значну увагу в сучасних наукових дослідженнях. Останні дослідження у сфері планування маршрутів для безпілотних літальних апаратів (БПЛА) спрямовані на розв'язання задачі максимізації покриття території при мінімальних витратах ресурсів.

У роботі [1] представлено огляд методів планування покриття для роботів, виділяючи ефективні стратегії маршрутизації для розвідки та інспекції великих територій. У [2] досліджено методи пошуку та переслідування у мобільній робототехніці, що можуть бути застосовані до задач дронового моніторингу. У дослідженні [3] розглядалися підходи до забезпечення безперервного покриття бездротового зв'язку за допомогою автономних БПЛА, що має значення для управління та передачі даних у реальному часі.

У роботі [4] запропоновано огляд сучасних технологій дронового моніторингу навколишнього середовища, підкреслюючи можливості безпілотників для збору просторових даних. У [5] досліджено проблему одночасного визначення місцеположення та картографування (SLAM), яка є важливою складовою автономної навігації БПЛА.

У дослідженні [6] розглянуто проблему вибору датчиків для завдань спостереження, що є ключовим аспектом для оптимального використання БПЛА в місіях збору даних. У [7] автори запропонували алгоритми семплінгового планування руху, які можуть бути ефективно застосовані до задач оптимального маршруту дрона. У [8] детально розглянуто алгоритми планування, що використовуються для автономного руху безпілотних систем.

У роботі [9] автори застосували змішане-цілочислове лінійне програмування для планування траєкторії літальних апаратів з урахуванням уникнення зіткнень, що є важливим аспектом оптимального маршруту БПЛА. У [10] досліджено використання інтелектуальних роїв БПЛА для спільного пошуку та відстеження об'єктів, що може підвищити ефективність спостереження.

У статті [11] автори запропонували методи планування маршруту БПЛА з урахуванням невизначеностей для максимального збору інформації, що особливо актуально у задачах спостереження, а у [12] оптимізували траєкторії для одночасного відстеження кількох рухомих цілей за допомогою БПЛА.

У дослідженні [13] представлено методи для планування місій БПЛА, використовуючи метаевристичні алгоритми, що дозволяє адаптивно оптимізувати маршрут дрона. У [14] запропоновано алгоритми для оптимізації траєкторій БПЛА у задачах нагляду, що мають практичну значущість для військових і цивільних застосувань. У [15] розглянуто оптимізацію траєкторії дронів для максимального покриття у місіях спостереження, застосовуючи комбінаторні методи оптимізації.

**Виділення недосліджених частин загальної проблеми.** Аналіз літератури свідчить про значну увагу до проблеми оптимізації траєкторії дронів. Незважаючи на значний обсяг досліджень, залишається відкритим питання оптимального планування маршрутів БПЛА для максимізації кількості обстежених об'єктів за обмеженими ресурсами (час польоту, енергія). Існуючі роботи переважно зосереджені або на забезпеченні суцільного покриття, або на оптимізації відстеження рухомих цілей, однак не враховують комбінацію цих аспектів для ефективного використання дронів у реальних умовах. Таким чином, у цій статті пропонується підхід до оптимізації маршрутів БПЛА, спрямований на максимізацію кількості обстежених об'єктів.

**Мета роботи** полягає в розробленні та порівняльному аналізі алгоритмів, призначених для задачі визначення траєкторії польоту за якої досягає максимуму кількість обстежених об'єктів.

## Постановка задачі

**Змістовна постановка задачі.** Маємо прямокутну ділянку з координатами вершин: $(0,0), (0, Y), (X, Y), (X, 0)$. На цій ділянці розміщено $n$ об'єктів, кожен з яких має форму кола радіуса $r$. Для об'єкта $i$ $(i = 1, \dots, n)$ відомі координати центру кола $(x_i, y_i)$ та вага (важливість) $w_i$.

Дрон починає свій маршрут з деякої (довільної) точки границі ділянки і летить по прямій. Необхідно знайти такі точку старту та напрямок польоту дрона, щоб зважена кількість об'єктів, над якими він пролетить, була максимальною.

Передбачається, що польотного ресурсу дрону вистачає для виконання поставленої місії. Ця задача є окремим елементом у складі більш комплексних моделей планування місій дронів. Зокрема, у задачах, де необхідно враховувати обмежений запас енергії, можливість зміни курсу під час польоту, пріоритетність певних зон огляду, а також наявність перешкод. Досліджувана задача може використовуватись як базовий блок для побудови ефективних маршрутів.

**Приклад задачі.** Є прямокутна ділянка з координатами вершин $(0,0), (0,17), (18,17), (18,0)$. На цій ділянці розміщено 12 об'єктів, кожен з яких має форму кола радіусу 0,6 (рис. 1). Для об'єкту $i$ $(i = 1, \dots, 12)$ відомі координати центру кола $(x_i, y_i)$ та вага (важливість) $w_i$ (табл. 1).

Дрон починає свій маршрут з деякої (довільної) точки границі ділянки і летить по прямій. Необхідно знайти такі точку старту та напрямок польоту дрона, щоб зважена кількість об'єктів, над якими він пролетить, була максимальною.

*Таблиця 1 – Об'єкти, їхні координати і ваги*

| Об'єкт № | Координати $(x_i, y_i)$ | Вага $w_i$ | Об'єкт № | Координати $(x_i, y_i)$ | Вага $w_i$ | Об'єкт № | Координати $(x_i, y_i)$ | Вага $w_i$ |
|---|---|---|---|---|---|---|---|---|
| 1 | (2, 15) | 7 | 5 | (2, 9) | 6 | 9 | (5, 3) | 3 |
| 2 | (7, 14) | 5 | 6 | (8, 9) | 7 | 10 | (9, 5) | 2 |
| 3 | (14, 15) | 8 | 7 | (16, 10) | 9 | 11 | (13, 5) | 2 |
| 4 | (11, 12) | 5 | 8 | (2, 5) | 9 | 12 | (15, 2) | 4 |

Джерело: розроблено авторами.

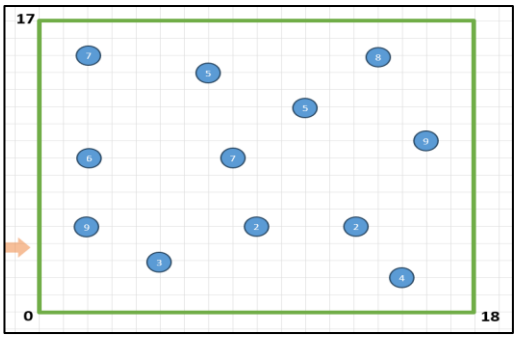
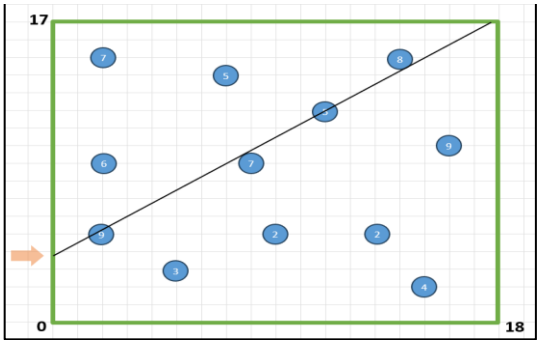



*Рис. 1. Візуалізація об'єктів для постановки задачі*

Джерело: розроблено авторами.

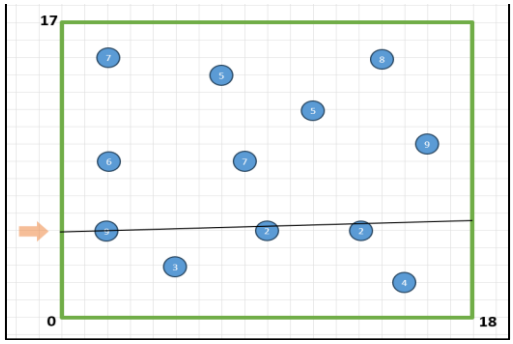
Приклади розв'язків наведені на рис. 2.

Розв'язок 1




Зважена кількість об'єктів, над якими пролетів дрон: 9+7+5+8=29

Розв'язок 2




Зважена кількість об'єктів, над якими пролетів дрон: 9+2+2=13

*Рис. 2. Можливі траєкторії польоту та відповідні зважені кількості об'єктів*

Джерело: розроблено авторами.

**Математична постановка задачі.** Для формалізації задачі побудови оптимальної траєкторії руху дрона необхідно ввести відповідні математичні позначення та сформулювати умови, які визначають шуканий розв'язок.

**Допоміжні математичні відомості.** Метою розв'язання задачі є знаходження параметрів прямої, яка описується рівнянням виду $y = kx + b$, де $k$ — кутовий коефіцієнт, що визначає нахил прямої відносно осі абсцис, а $b$ — ордината точки перетину прямої з віссю $y$. При $k > 0$ пряма піднімається праворуч, при $k < 0$ — опускається праворуч, а при $k = 0$ — є горизонтальною.

В основу математичної моделі задачі покладено умову перетину прямої $y = kx + b$ та кола радіуса $r$ з центром в точці $(x_i, y_i)$, $i = 1, \dots, n$, яке задається формулою $(x - x_i)^2 + (y - y_i)^2 = r^2$.

Підставивши рівняння прямої у рівняння кола, отримаємо вираз:

$$(x - x_i)^2 + (kx + b - y_i)^2 = r^2, \quad i = 1, \dots, n.$$

Аби пряма перетинала коло або доторкалась до нього, дискримінант $D$ цього квадратного рівняння повинен бути не від'ємним:

$$D = (-2x_i + 2kb - 2ky_i)^2 - 4(1 + k^2)(x_i^2 + b^2 - 2by_i + y_i^2 - r^2) \geq 0, \ i = 1, \dots, n.$$

**Вхідні дані.** $X$ — довжина ділянки; $Y$ — ширина ділянки; $n$ — кількість об'єктів; $r$ — радіус об'єктів; $x_i$ — координата центру об'єкту $i$ по осі $x$ $(i = 1, \dots, n)$; $y_i$ — координата центру об'єкту $i$ по осі $y$ $(i = 1, \dots, n)$; $w_i$ — вага об'єкту $i$ $(i = 1, \dots, n)$.

**Змінні.** Необхідно знайти: $b$ — координати точки перетину прямої лінії, вздовж якої летить дрон, з віссю $y$; $k$ — кутовий коефіцієнт прямої. Відповідно до рівняння прямої введемо такі змінні:

$$d_i = \begin{cases} 1, \text{якщо } D \geq 0 \\ 0, \text{якщо інакше} \end{cases} \quad (i = 1, \dots, n).$$

**Обмеження**

$$b \leq> 0, \tag{1}$$

$$k \leq> 0, \tag{2}$$

$$d_i = 0 \text{ або } d_i = 1 \ (i = 1, \dots, n). \tag{3}$$

**Цільова функція.** Оскільки треба знайти таку траєкторію руху дрона аби зважена кількість об'єктів, над якими він пролетить, була максимальною, то цільова функція буде мати вигляд:

$$W = \sum_{i=1}^{n} w_i d_i \to \max.$$

**Алгоритми розв'язання задачі**

Значення параметрів $k$ та $b$, що визначають пряму $y = kx + b$, є неперервними величинами. Однак з огляду на специфіку поставленої задачі — необхідність максимізації сумарної ваги дискретної множини об'єктів (кіл) — існує можливість певним чином дискретизувати простір допустимих розв'язків. Це дозволяє перейти від задачі неперервної оптимізації до задачі дискретної оптимізації, що робить доступними для застосування класичні методи дискретної оптимізації та комбінаторні алгоритми.

Для розв'язання задачі можуть бути застосовані такі підходи.

*Алгоритми перебору — повного або часткового.* Методи, що передбачають систематичний перебір певної дискретної множини можливих розв'язків. Методи повного пере-

бору знаходять оптимальний розв'язок. Якщо перебір є неповним, то методи не гарантують глобальної оптимальності, але зазвичай забезпечують прийнятну точність та стабільність. Головний недолік – висока складність перебору при збільшенні кількості об'єктів.

*Жадібні (Greedy) алгоритми.* Цей підхід ґрунтується на поетапному виборі локально оптимального розв'язку без повернення назад. Перевага жадібних алгоритмів полягає в їх простоті й низькій обчислювальній складності, що забезпечує високу швидкість обчислень. Недоліком є неможливість гарантувати глобальний оптимум.

*Евристичні методи.* Це клас методів оптимізації, які базуються на певних гіпотезах, припущеннях, інтуїтивних ідеях чи практичних спостереженнях. Їх застосовують переважно у випадках, коли задача має високу складність (наприклад, NP-складні задачі), і отримання точного розв'язку за розумний час практично неможливе. На відміну від точних алгоритмів, евристики не гарантують отримання оптимального розв'язку, але дозволяють швидко знаходити прийнятні, наближені розв'язки.

У цій роботі розроблено алгоритми таких методів:

– жадібного (Greedy);

– часткового перебору (далі — будемо називати його Partial Search);

– евристичного (далі — Floating Line).

**Розробка жадібного алгоритму.** Жадібний алгоритм — це стратегія, яка на кожному кроці обирає локально оптимальний вибір, очікуючи, що така послідовність рішень приведе до глобального оптимального розв'язку [16]. Має низьку обчислювальну складність, однак не гарантує глобально оптимального маршруту.

Ідея розробленого алгоритму полягає в рекурсивному поділі вхідного прямокутника на менші прямокутники із подальшим вибором "найбільш зважених" областей для проведення через них прямої.

Для спрощення пояснення припустимо, що всі об'єкти (кола) мають однакову вагу. На першому кроці прямокутна область ділиться на чотири однакові підобласті. З них обираються дві з найбільшою кількістю об'єктів. Припустимо, що області із найбільшою вагою на 1-й ітерації це 2-га і 4-та чверті. На 2-й ітерації ми ділимо кожну з цих областей на 4 менші та з'єднуємо центри цих поділених прямокутників прямою і знову обираємо із 8-ми новоутворених прямокутників два з найбільшими вагами. Далі кожна з цих під-областей також ділиться на чотири, після чого проводиться пряма через центри двох найбільш «зважених» новоутворених прямокутників. Рекурсивний поділ продовжується, поки діагональ новоствореного прямокутника перевищує діаметр $r$ кола. Після досягнення цієї межі повертається остання збудована пряма – вона і є наближеним розв'язком.

*Псевдокод жадібного алгоритму*

**Вхід**
$X$ — довжина ділянки
$Y$ — ширина ділянки
$n$ — кількість об'єктів
$r$ — радіус об'єктів
$x_i, y_i$ — координати центру об'єкту $(i = 1, \dots, n)$
$w_i$ — вага об'єкту $i$ $(i = 1, \dots, n)$
**Вихід**
$k_{best}, b_{best}, W_{best}$ *//Параметри прямої*
Функція **GreedySplit**(*прямокутник*):
  **if** діагональ *прямокутника* $\leq r$
    **then повернути** $W$, $k$, $b$ останньої знайденої прямої
    **else**
      **Поділити** rectangle на 4 рівні *підпрямокутники*
      Для кожного *підпрямокутника* **обчислити** суму ваг кіл, що в ньому знаходяться

**Обрати** два *підпрямокутника* з найбільшими сумарними вагами
**Побудувати** пряму через центри цих двох *підпрямокутників* (знайти відповідні $k, b, W$)
**Сформувати** *новий прямокутник* (можливі два варіанти: обрані центри є кінцями діагоналі прямокутника або кінцями його серединної лінії)
Рекурсивно викликати **GreedySplit**(*новий прямокутник*)
**end if**
$x_{min} = 0, y_{min} = 0, x_{max} = X, y_{max} = Y$ // параметри *вхідного прямокутника*
Викликати **GreedySplit**(*вхідний прямокутник*)
**Повернути** $W_{best}$ , $k_{best}$, $b_{best}$

На кожному кроці алгоритму виконується поділ області на 4 частини і обчислення сумарної ваги кіл у кожній із них. Оскільки для кожної підобласті здійснюється перевірка до $n$ кіл, то складність одного рівня рекурсії становить $O(n)$.

Після кожного поділу лінійні розміри обраної області зменшуються вдвічі, а площа — у чотири рази. Рекурсія триває доти, поки діагональ поточної області $D_l$ ($l$ — номер рівня рекурсії) не стане меншою або рівною діаметру $r$ кола.

Початкова діагональ: $D_0 = \sqrt{X^2 + Y^2} \sim M$ (де $M$ — розмір області). На $l$-му рівні рекурсії діагональ дорівнює $D_l = \frac{D_0}{2^l}$. Рекурсія завершується, коли $D_l \le r$, тобто:

$$\frac{D_0}{2^l} \le r \;\Rightarrow\; 2^l \ge \frac{D_0}{r} \;\Rightarrow\; l \ge \log_2\left(\frac{D_0}{r}\right).$$

Отже, максимальна глибина рекурсії: становить $O(\log M)$, де $M = \frac{D_0}{r}$ — відношення початкової діагоналі до діаметра кола.

Оскільки на кожному рівні виконується $O(n)$ операцій, а таких рівнів не більше ніж $O(\log M)$, то загальна часова складність становить $O(n \log M)$.

**Розробка алгоритму часткового перебору.** Алгоритм повного перебору (Bruteforce) є найпростішим способом пошуку оптимального розв'язку шляхом послідовного перебору всіх можливих варіантів [16]. Як було зазначено вище, для того, щоб застосовувати класичні методи дискретної оптимізації до задачі визначення оптимальних параметрів $k$ та $b$, необхідно певним чином дискретизувати простір допустимих розв'язків. Враховуючи, що параметри прямої є неперервними величинами, повний перебір усіх можливих варіантів є неможливим через нескінченну кількість допустимих розв'язків. Проте в розглянутій задачі оптимальна пряма, яка максимізує сумарну вагу кіл, завжди проходитиме через певні ключові точки — точки дотику або перетину з колами. Це дозволяє значно скоротити множину аналізованих варіантів і перейти до задачі дискретної оптимізації, перебираючи лише прямі, визначені такими точками.

В основу розробленого алгоритму покладено ідею методу часткового перебору, який у цьому випадку не повною мірою перебирає усі можливі розв’язки (надалі цей алгоритм будемо називати Partial Search). Хоча звуження множини допустимих прямих дозволяє суттєво зменшити обчислювальну складність, існує ймовірність, що знайдений таким чином розв’язок не буде абсолютним оптимумом, а лише наближеним до нього.

*Сектором* у контексті вирішення цієї задачі будемо називати нескінченну область між двома прямими, що є дотичними до двох кіл.

Нехай маємо два кола, координатами центрів яких є $(x_i, y_i)$ та $(x_j, y_j)$, $1 \le i, j \le n$, $i \ne j$. Рівняння зовнішніх дотичних до цих кіл такі:

$$y = kx + b_1,\; y = kx + b_2,$$

де $k = \frac{y_j - y_i}{x_j - x_i}$ – кутовий коефіцієнт цих дотичних (які є паралельними);

$b_1 = y_i + rn_y - k(x_i + rn_x)$ – вільний член першої дотичної;

$b_2 = y_i - rn_y - k(x_i - rn_x)$ – вільний член другої дотичної;

$n_x = -\frac{y_j - y_i}{D}$, $n_x = \frac{x_j - x_i}{D}$ – координати одиничного вектора, ортогонального до дотичних;

$d = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ – відстань між центрами кіл.

Нижче наведено приклад того, як виглядає сектор та всі можливі унікальні випадки належності кола до сектора (рис. 3).

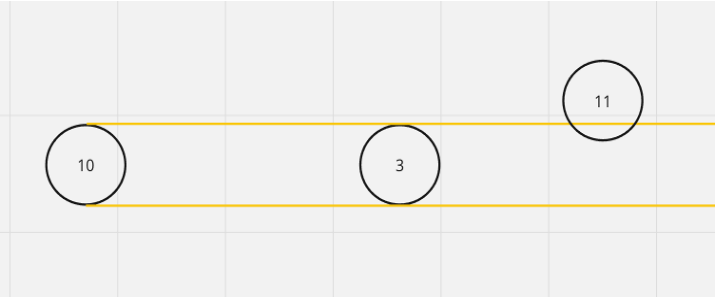



*Рис. 3. Приклад створення сектору алгоритмом Partial Search*
Джерело: розроблено авторами.

У цьому прикладі область між двома жовтими лініями — це сектор. Будемо казати, що коло належить сектору, якщо воно перетинає або дотикається хоча б до однієї прямої, що формує сектор.

Алгоритм перебору для цієї задачі базується на систематичному переборі секторів, утворених усіма можливими парами кіл (об'єктів). Кожна пара визначає сектор, обмежений двома прямими — дотичними або такими, що перетинають відповідні кола. Для кожного такого сектора обчислюється сумарна вага об'єктів, що перетинаються з однією з двох прямих сектора. Серед них обирається та пряма, яка забезпечує максимальне покриття за вагою. Якщо знайдене значення перевищує поточний максимум, рекордний розв'язок оновлюється.

Отже, нам треба мати можливість визначити, чи перетинається певне коло із заданою прямою. Розглянемо умови, коли пряма дотикається чи перетинає коло.

Рівняння кола має вигляд $(x-p)^2 + (y-q)^2 = r^2$, де $(p, q)$ — координати центра кола, $r$ — радіус. Рівняння прямої: $y = kx + b$. Щоб визначити, чи перетинається коло з прямою, підставимо рівняння прямої в рівняння кола й отримаємо систему рівнянь:

$$\begin{cases} (x-p)^2 + (y-q)^2 = r^2, \\ y = kx + b. \end{cases}$$

Підставляючи рівняння прямої в рівняння кола, отримаємо квадратне рівняння відносно $x$:

$$(x-p)^2 + (kx + b - q)^2 = r^2,$$

або, після перетворень:

$$(1+k^2)x^2 + [2k(b-q) - 2p]x + [p^2 + (b-q)^2 - r^2] = 0.$$

Позначимо коефіцієнти цього рівняння як:

$$A = 1 + k^2,\ B = 2k(b-q) - 2p,\ \mathrm{C} = p^2 + (b-q)^2 - r^2,\ D = B^2 - 4AC.$$

Якщо дискримінант $D$ більший за нуль, то маємо два розв'язки — пряма перетинає коло в точках $(x_1, y_1)$, $(x_2, y_2)$:

$$x_1 = \frac{-B + \sqrt{D}}{2A},\ y_1 = kx_1 + b, \quad x_2 = \frac{-B - \sqrt{D}}{2A},\ y_2 = kx_2 + b.$$

Якщо дискримінант рівний нулю, то маємо один розв'язок – пряма дотикається до кола:

$$x_1 = \frac{-B}{2A},\ y_1 = kx_1 + b.$$

*Псевдокод алгоритму Partial Search*

**Вхід**
$X$ — довжина ділянки
$Y$ — ширина ділянки
$n$ — кількість об'єктів
$r$ — радіус об'єктів
$x_i, y_i$ — координати центру об'єкту $(i = 1, \dots, n)$
$w_i$ — вага об'єкту $i$ $(i = 1, \dots, n)$
**Вихід**
$k_{best}, b_{best}, W_{best}$ *//Рекордний розв'язок, рекорд*

**Відсортувати** кола у порядку незростання їх ваг $w$
$W_{best} := 0$
**for** Кожної пари кіл $i, j$, де $1 \le i < j \le n$
    **Побудувати сектор**, що формується двома прямими
    **Визначити** параметри першої прямої: $k_1, b_1$
    **Визначити** параметри першої прямої: $k_2, b_2$
    $W_1 := 0$ *// сума ваг кіл, які перетинаються першою прямою сектора*
    $W_2 := 0$ *// сума ваг кіл, які перетинаються другою прямою сектора*
    **for** $i := 1$ **to** $n$ *//Цикл по об'єктам*
        **if** *(*перша пряма перетинає коло $i$*)*
            **then** $W_1 := W_1 + w_i$
        **if** *(*друга пряма перетинає коло $i$*)*
            **then** $W_2 := W_2 + w_i$
    **end for**
    $W_{cur} := \max\{W_1, W_2\}$
    $(k_{cur}, b_{cur})$ := параметри прямої, що відповідають $W_{cur}$
    *//Перевизначення рекордного розв'язку*
    **if** $W_{cur} > W_{best}$
        **then** $W_{best} := W_{cur}$, $k_{best} := k_{cur}$, $b_{best} := b_{cur}$
**end for**
**Повернути** $W_{best}, k_{best}, b_{best}$

Алгоритм перебирає пари кіл, отже, базова складність перебору пар становить $O(n^2)$. Для кожної пари кіл алгоритм аналізує решту $n$ кіл для визначення належності сектору, що в сукупності дає складність $O(n^3)$.

**Розробка евристичного алгоритму Floating line.** Ідея алгоритму полягає у переборі множини прямих, що проходять через центральну точку ділянки. Починаючи з горизонтальної прямої (кут нахилу $0°$), пряма поступово повертається на деякий невеликий кут ($\Delta$) до повного оберту на $180°$. На кожному кроці алгоритм обчислює параметри нової прямої; перевіряє, з якими колами пряма перетинається або до яких дотикається; обчислює сумарну вагу цих кіл; запам'ятовує поточний рекордний розв'язок.

Розглянемо коло із центром у точці $O(X/2,\ Y/2)$. Пряма задається рівнянням $y = = kx + b$, будемо вважати, що вона проходить через точку $O$. Початково $k = 0, b = Y/2$. Ідея алгоритму полягає у повороті цієї прямої на деякий кут $\Delta > 0$, допоки пряма не розвернеться на $180°$ та перевірці перетину кожного з кіл цією прямою.

Початково кут прямої рівний $0°$. Для того, щоб отримати нову пряму після повороту, потрібно до кута повороту прямої додати певну величину, яку позначимо як $\Delta$. Рівняння

нової прямої $y = k'x + b'$. Отже, новий коефіцієнт $k' = \text{tg}\,(\alpha + \Delta)$, де $\alpha$ — попередній кут нахилу. Точка, через яку завжди проходитиме пряма — це точка $O$. Отже, $b' = \frac{Y}{2} - -k' \cdot \frac{X}{2}$. Тоді рівняння нової прямої набуде вигляду:

$$y = k'x + b' = \text{tg}\,(\alpha + \Delta) \cdot x + \frac{Y}{2} - \text{tg}\,(\alpha + \Delta) \cdot \frac{X}{2}.$$

*Псевдокод алгоритму Floating line*

**Вхід**
$X$ — довжина ділянки.
$Y$ — ширина ділянки
$n$ — кількість об'єктів.
$r$ — радіус об'єктів.
$x_i, y_i$ — координати центру об'єкту $(i = 1, \dots, n)$.
$w_i$ — вага об'єкту $i$ $(i = 1, \dots, n)$.
$\Delta$ — крок повороту прямої *//Параметр алгоритму*
**Вихід**
$k_{best}, b_{best}, W_{best}$ *//Рекордний розв'язок, рекорд*
$J_{max} := \frac{180}{\Delta}$ *//Кількість ітерацій*
$W_{best} := 0,\ k_{best} := 0,\ b_{best} := 0$
$\alpha := 0, k := 0, b := \frac{Y}{2}$
$j = 0$
**while** $(j \le J_{max})$
    $k' := \text{tg}\,(\alpha + \Delta),\ b' := \frac{Y}{2} - k' \cdot \frac{X}{2}$ *//Параметри прямої*
    $W_{cur} := 0$
    **for** $i := 1$ **to** $n$ *//Цикл по об'єктам*
        **if** *(пряма перетинає коло i)*
            **then** $W_{cur} := W_{cur} + w_i$
    **end for**
    *//Перевизначення рекордного розв'язку*
    **if** $W_{cur} > W_{best}$
        **then** $W_{best} := W_{cur}$ , $k_{best} := k'$, $b_{best} := b'$
    *//Перехід до аналізу наступної прямої*
    $i := i + 1,$
    $\alpha := \alpha + \Delta,\ k := \text{tg}(\alpha),\ b := \frac{Y}{2} - k \cdot \frac{X}{2}$
**end while**
**Повернути** $W_{best}$ , $k_{best}$, $b_{best}$

У алгоритмі Floating line пряма повертається від кута $\alpha = 0^\circ$ до $\alpha = 180^\circ$ з фіксованим кроком $\Delta$ (у градусах). Отже, кількість ітерацій, які виконає алгоритм:

$$J_{max} = \frac{180^\circ}{\Delta}.$$

При зменшенні $\Delta$, значення $J_{max}$ зростає обернено пропорційно: якщо $\Delta = 1^\circ$ — 180 ітерацій; якщо $\Delta = 0{,}1^\circ$ — 1800 ітерацій; якщо $\Delta = 0{,}01^\circ$ — 18000 ітерацій.

На кожній ітерації перевіряється, чи перетинає задана пряма кожне з $n$ кіл. Ці перевірки — лінійні по $n$, тобто одна ітерація має складність: $O(n)$. З урахуванням $J_{max} = \frac{180^\circ}{\Delta}$, маємо загальну складність $O\left(\frac{n}{\Delta}\right)$.

Цей метод є евристичним методом пошуку розв'язку (не гарантує знаходження оптимуму) і може ефективно використовуватись як швидкий наближений метод.

Усі три методи мають відносно помірні вимоги до пам'яті. Жадібний алгоритм потребує додаткової пам'яті для зберігання об'єктів у кожній із рекурсивно створених підобластей, а також має витрати на стек викликів. У гіршому випадку його просторова складність становить $O(n \log M)$. Floating Line має просторову складність $O(n)$, оскільки на кожному кроці перевіряє всі об'єкти, не створюючи додаткових структур. Partial Search використовує $O(n^2)$ пам'яті для зберігання пар об'єктів і результатів перевірки належності до прямих.

## Дослідження розроблених алгоритмів

**Класифікація задач.** Точність та ефективність алгоритмів для вирішення задачі залежить від наступних параметрів: $n$ — кількість кіл; $X$ — абсциса площини; $Y$ — ордината площини; $R$ — радіус усіх кіл; circles — масив усіх кіл.

***Приклад розв'язання задачі розмірності*** $\boldsymbol{n = 10}$**.** Задана прямокутна ділянка розмірності 30×30 з колами радіусом 2 та згенеровано 10 кіл з такими координатами центру та вагами: 5,6,30; 11,15,10; 17,23,40; 6,27,5; 14,3,7; 13,8,13; 20,20,20; 26,4,15; 8,19,17; 23,6,13. Отримані результати наведені в табл. 2 та на рис. 4.

*Таблиця 2 – Результат розв'язання задачі розмірності* $n = 10$

| Алгоритм | Рівняння прямої | Вага | Час виконання |
|---|---|---|---|
| Floating line | $y = 3{,}92316x - 43{,}84734$ | 60 | 0,05876 |
| Partial Search | $y = 1{,}416676x + 2{,}38478$ | 80 | 0,00121 |
| Greedy | $y = 1{,}5x + 0{,}46875$ | 80 | 0 |

Джерело: розроблено авторами.

Як можна побачити з результатів, жадібний алгоритм та Partial Search знайшли найкращій результат розв'язку і при цьому виконались за дуже малий час. Час роботи жадібного алгоритму настільки малий, що навіть не відслідкувався. Floating line показав гірший результат та більше затраченого часу на пошук розв'язку.

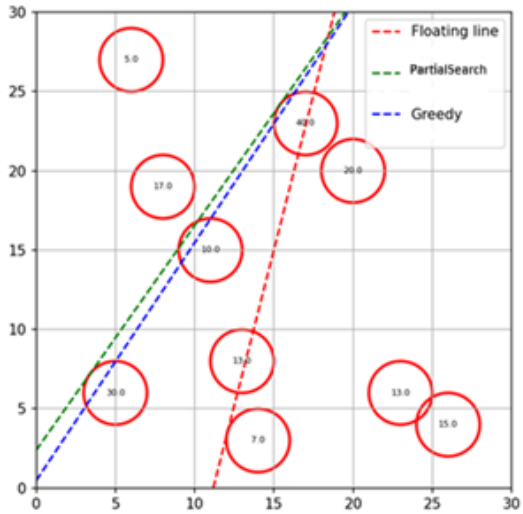



*Рис. 4. Результат розв'язання задачі розмірності* $n = 10$

Джерело: розроблено авторами.

***Приклад розв'язання задачі розмірності*** $\boldsymbol{n = 30}$**.** Задана прямокутна ділянка розмірності 500×300, 30 кіл з $r = 5$ та такими координатами центрів та вагами відповідно: 6,6,30; 1,15,10; 17,23,40; 6,27,5; 14,8,7; 13,8,13; 270,111,20; 147,254,35; 186,193,17; 203,162,13; 372,122,25; 99,99,45; 334,193,45; 254,295,84; 342,29,29; 92,183,49; 129,239,58; 78,23,19; 289,32,65; 111,111,43; 151,211,23; 445,293,7; 116,280,40; 420,100,50; 461,78,8; 324,246,32; 359,266,32; 50,250,50; 178,113,35; 235,94,30. Отримані результати наведені в табл. 3 та на рис. 5.

*Таблиця 3 – Результат розв'язання задачі розмірності* $n = 30$

| Алгоритм | Рівняння прямої | Вага | Час виконання |
|---|---|---|---|
| Floating line | $y = 0{,}56808x + 7{,}98023$ | 160 | 0,22228 |
| Partial Search | y = 0,448x + 186,68683 | 204 | 0,00301 |
| Greedy | $y = 1{,}96x - 6{,}5625$ | 227 | 0,00251 |

*Джерело: розроблено авторами.*

При $n = 30$ найкращі результати показав жадібний алгоритм та витратив на виконання найменшу кількість часу. Partial Search показав середній результат та гірший час виконання. Алгоритм Floating line показав найгірші показники як за часом виконання, так і за значенням цільової функції.

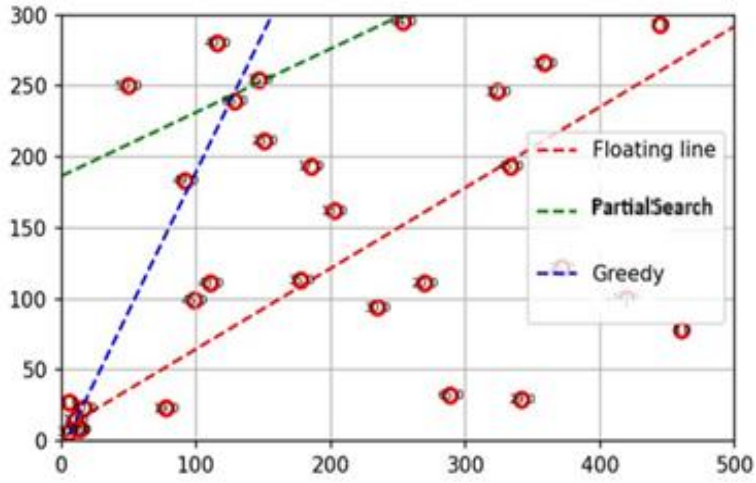



*Рис. 5. Результат розв'язання задачі розмірності* $n = 30$

Джерело: розроблено авторами.

**Результати експериментів.** Метою планування та проведення експериментів є аналіз алгоритмів, а саме дослідження впливу параметрів алгоритмів на їх ефективність. Аналіз відбувався за такими пунктами: 1) вплив параметрів алгоритмів на результат; 2) вплив умов задачі на алгоритми; 3) порівняння алгоритмів за часом та точністю.

*Експеримент 1. Дослідження впливу параметру* $\Delta$ *на результати роботи алгоритму Floating line.* Тільки алгоритм Floating line має параметр (крок зміни кута $\Delta$, на який обертається пряма). Нехай $0{,}01 \leq \Delta \leq 0{,}1$, крок зміни $\Delta$ дорівнює $0{,}01$. Результат експерименту представлено на рис. 6.

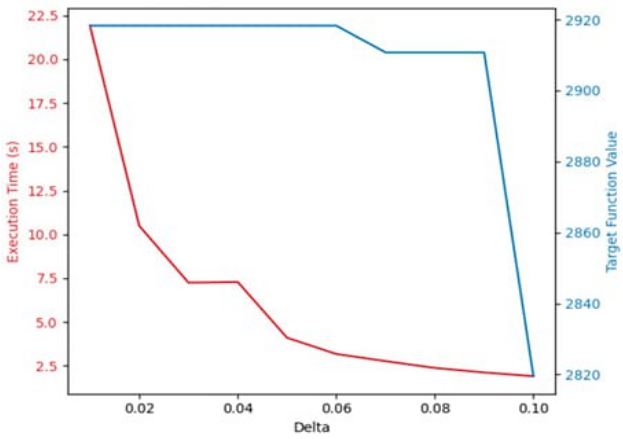



*Рис. 6. Результат експерименту 1*

Джерело: розроблено авторами.

При зміні кута Δ змінюються час виконання та точність. Чим більше Δ, тим менше різних варіантів прямої треба перевірити, тому при збільшенні кута повороту, час виконання зменшується. Чим менший крок повороту, тим більша ймовірність знайти найкращий результат, тому при зменшенні Δ збільшується точність.

*Експеримент 2. Порівняння експериментальної трудомісткості алгоритмів, визначення впливу розмірності задачі на час роботи алгоритмів.* Усі алгоритми досліджувались за часовою складністю.

*Greedy алгоритм.* Результат виконання алгоритму Greedy при $1 \leq n \leq 1000$ представлено на рис. 7.

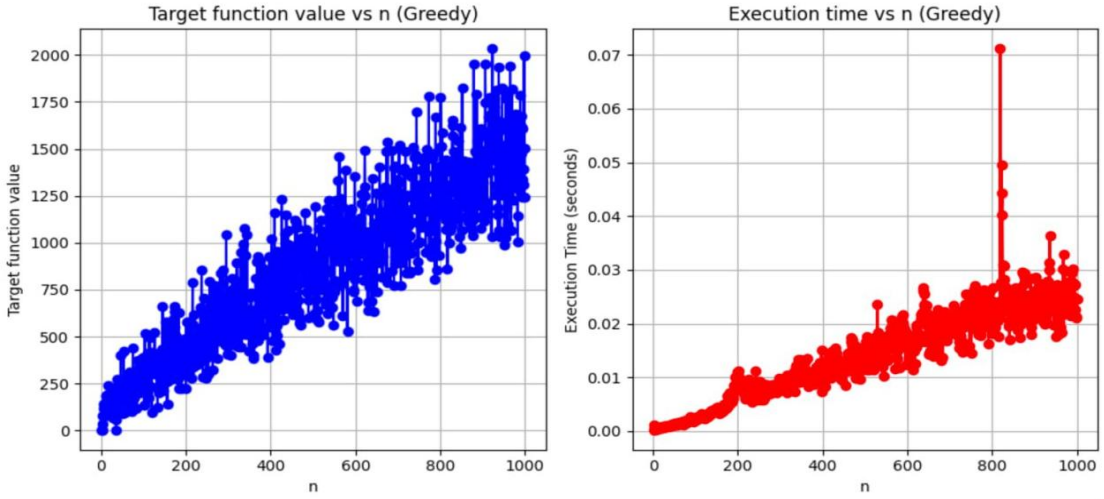



*Рис. 7. Результат експерименту 2 (алгоритм Greedy)*

Джерело: розроблено авторами.

Як можна побачити з графіків, при збільшенні кількості кіл, час роботи алгоритму зростає лінійно, що й відповідає теоретичній складності $O(n)$.

*Partial Search.* Результат виконання алгоритму Partial Search при $1 \leq n \leq 1000$ подано на рис. 8. Час виконання зростає нелінійно, що відповідає теоретичній складності.

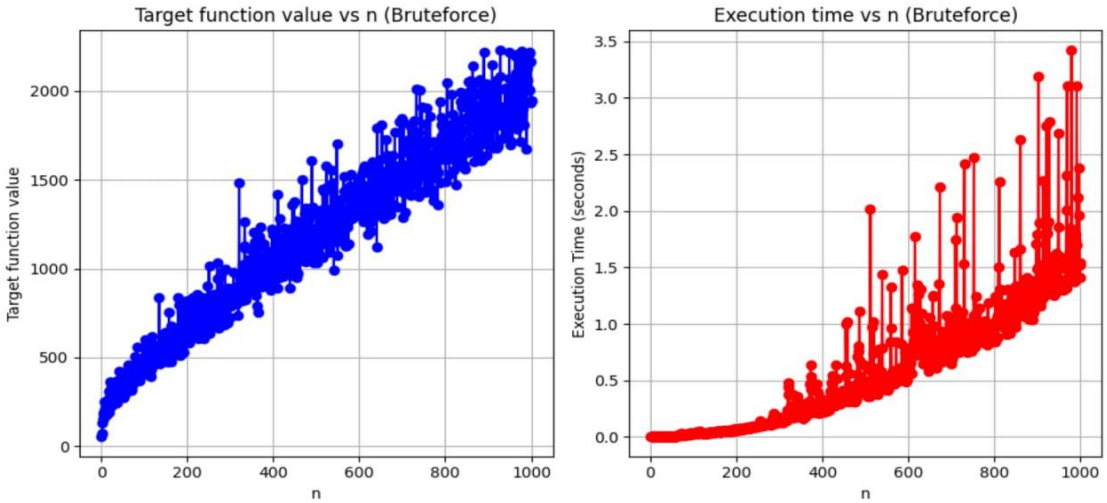



*Рис. 8. Результат експерименту 2 (алгоритм Partial Search)*

Джерело: розроблено авторами.

*Floating line.* Результат виконання алгоритму Floating line при $1 \leq n \leq 1000$ подано на рис. 9. Час виконання зростає лінійно, що відповідає теоретичній складності.

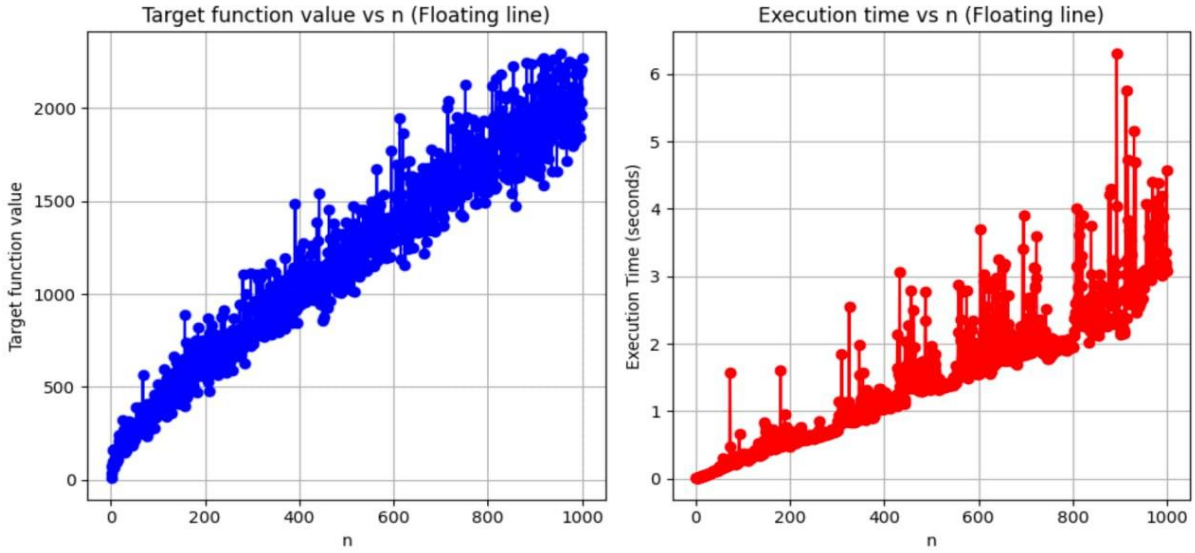



*Рис. 9. Результат експерименту 2 (алгоритм Floating line)*

Джерело: розроблено авторами.

*Експеримент 3. Порівняльний аналіз ефективності розроблених алгоритмів (визначення впливу розмірності задачі на точність алгоритмів).* Нехай $800 \leq n \leq 1000$, кількість задач, які генеруються для кожного $n$ дорівнює 30. Отриманий результат представлено на рис. 10.


*Рис. 10. Результат порівняння алгоритмів за точністю*

Джерело: розроблено авторами.

Як можна побачити з результатів, найкращі результати показують алгоритми Floating line та Partial Search — середня похибка 3-8 % (відхилення від найкращого знайденого). Жадібний алгоритм показав найгірші результати та має середню похибку в 22-39 %.

*Вплив конфігурації розташування об'єктів на ефективність алгоритмів*

Слід зазначити, що різні конфігурації розташування об'єктів по-різному впливають на ефективність алгоритмів.

*Жадібний алгоритм.* Є ефективним у випадку рівномірного або кластеризованого розташування об'єктів, особливо коли в щільніших зонах концентруються об'єкти з вищою вагою; якщо «зважена маса» локалізована в певній ділянці, то алгоритм швидко зійдеться до цієї області. Демонструє нижчу ефективність у випадках нерівномірного розподілу, коли важливі об'єкти розкидані по всій вихідній прямокутній ділянці, у ситуаціях, коли оптимальна пряма проходить через слабо виражену зону, алгоритм може її «не побачити» через локальний характер прийняття рішень.

*Алгоритм Floating Line.* Показує хороші результати при центрально симетричних або кругових конфігураціях розташування об'єктів, коли найбільш вигідна траєкторія проходить поблизу центру області, у таких випадках існує висока ймовірність її знаходження. Менш ефективний при асиметричних або крайових розташуваннях, коли оптимальна пряма проходить далеко від центра — тоді алгоритм із фіксованим центром обертання може не охопити цільову зону.

*Алгоритм Partial Search*. Ефективність цього методу не залежить від конфігурації розташування об'єктів, оскільки перебираються всі пари кіл. Таким чином, гарантовано будуть розглянуті всі ключові дотичні прямі.

**Висновки.** Для розв'язання поставленої задачі максимізації сумарної ваги об'єктів, які які обстежуються дроном, було розроблено три алгоритми: жадібний, часткового перебору (Partial Search) та евристичного (Floating line). Основною складністю при пошуку траєкторії руху прямої була неможливість аналітичного отримання точного розв'язку та оцінювання абсолютної точності розв'язків, що отримуються.

Жадібний алгоритм характеризується вибором локально оптимального розв'язку на кожному кроці, що суттєво знижує час обчислення, однак зумовлює значну похибку розв'язку порівняно з іншими методами. Особливо відчутною є його неточність у задачах малої розмірності. Цей алгоритм ефективний на великих наборах даних, які хаотично розподілені по площі.

Перевагою жадібного алгоритму є висока швидкість виконання, яка дозволяє отримати приблизну оцінку оптимального розв'язку за мінімальний час. Зокрема, під час експериментального дослідження було виявлено, що жадібний алгоритм працює в одному із випадків у 640 разів швидше, ніж алгоритм Partial Search.

Алгоритм Partial Search заснований на переборі скінченної множини прямих, утворених точками дотику або перетину кіл між собою. Це значно скорочує кількість варіантів порівняно з повним перебором усіх можливих прямих, проте не гарантує знаходження абсолютного оптимуму. Перевагою цього алгоритму є стабільність отримуваних розв'язків та відносно висока точність. Водночас суттєвим недоліком є суттєве зростання часу обчислень при збільшенні кількості об'єктів, що може зробити цей алгоритм малоефективним для великих задач.

Алгоритм Floating Line ґрунтується на дискретному переборі прямих, що проходять через задану фіксовану точку (центр площини). Його ефективність визначається співвідношенням між точністю розв'язку та швидкістю обчислень, яке можна регулювати за допомогою параметра $\Delta$ (крок повороту прямої). Перевагою цього алгоритму є його гнучкість та відносно висока точність при помірному часі роботи, особливо коли кола розташовані симетрично навколо центра площини. Однак при зменшенні параметра $\Delta$ точність зменшується, але швидкість збільшується, і навпаки. Це дозволяє користувачу підлаштовувати алгоритм під конкретні вимоги точності та часових ресурсів. Можна підвищити ефективність цього алгоритму, розглядаючи декілька варіантів розміщення точки $O$. Зокрема, одним з напрямків розвитку алгоритму Floating Line є знаходження центру мас (ваг) і обрання цієї точки за центр кола.

Найкращі результати показують алгоритми Floating line та Partial Search — середня похибка 3-8 % (відхилення від найкращого знайденого). Жадібний алгоритм показав найгірші результати та має середню похибку в 22-39 %.

Практичне застосування розроблених алгоритмів залежить від характеристик задачі. Якщо об'єктів багато, а обчислювальні ресурси обмежені, доцільно використовувати жадібний алгоритм, який забезпечує найшвидше отримання результату, хоч і з меншою точністю. У задачах, де важлива висока точність, незалежно від часу виконання, рекомендовано застосовувати метод часткового перебору. Алгоритм Floating Line може виступати як компромісне рішення, оскільки дозволяє регулювати баланс між точністю та швидкістю за допомогою параметра кроку повороту.

Розглянута задача може бути використана як підзадача в межах більш складних задач планування місій дронів, які враховують обмеження на польотний ресурс, можливість зміни напрямку руху, необхідність охоплення заданих пріоритетних зон, а також уникнення перешкод. Такий підхід дозволяє ефективно інтегрувати модуль побудови оптимальної прямої в ширший контекст багатокрокового планування траєкторій.

## Список використаних джерел

1. Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems, 61*(12), 1258–1276. https://doi.org/10.1016/j.robot.2013.09.004.

2. Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots, 31*(4), 299–316. https://doi.org/10.1007/s10514-011-9231-8.

3. Dhillon, S. S., & Smith, J. (2003). Wireless communication coverage using autonomous UAVs. *IEEE Transactions on Aerospace and Electronic Systems, 39*(2), 152–165. https://doi.org/10.1109/TAES.2003.1205972.

4. Hu, J., Zhang, L., & Wang, J. (2020). Drone-based environmental monitoring: An overview of recent developments. *Remote Sensing, 12*(4), 687. https://doi.org/10.3390/rs12040687.

5. Stachniss, C., Leonard, J. J., & Thrun, S. (2008). Simultaneous localization and mapping (SLAM): A survey of current trends. *IEEE Transactions on Robotics, 24*(5), 1050–1067. https://doi.org/10.1109/TRO.2008.2005913.

6. Isler, V., & Bajcsy, R. (2005). The sensor selection problem for surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics, 35*(4), 639–648. https://doi.org/10.1109/TSMCB.2005.850150.

7. Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research, 30*(7), 846–894. https://doi.org/10.1177/0278364911406761.

8. LaValle, S. M. (2006). Planning algorithms. Cambridge University Press. https://doi.org/10.1017/CBO9780511546877.

9. Richards, A., & How, J. P. (2002). Aircraft trajectory planning with collision avoidance using mixed-integer linear programming. *AIAA Guidance, Navigation, and Control Conference, 1*(1), 221–238. https://doi.org/10.2514/6.2002-4588.

10. Lin, L., Wu, Y., & Duan, H. (2018). UAV swarm intelligence for cooperative search and tracking. *IEEE Transactions on Cybernetics, 48*(3), 891–904. https://doi.org/10.1109/TCYB.2017.2671359

11. Evers, L., & Farinelli, A. (2019). UAV path planning under uncertainty for maximum information collection. *Journal of Field Robotics, 36*(2), 125–145. https://doi.org/10.1002/rob.21800.

12. Penicka, R., Saska, M., & Thomas, J. (2017). UAV surveillance of multiple moving targets using trajectory optimization. *Autonomous Robots, 41*(5), 1061–1078. https://doi.org/10.1007/s10514-017-9614-3.

13. Smith, R. G., & Becker, K. (2017). Real-time UAV mission planning using metaheuristic optimization techniques. *Journal of Aerospace Engineering, 31*(3), 04017085. https://doi.org/10.1061/(ASCE)AS.1943-5525.0000875

14. Bazgan, C., Hugot, H., & Vanderpooten, D. (2019). Efficient algorithms for optimizing UAV surveillance trajectories. *Operations Research, 67*(4), 952–969. https://doi.org/10.1287/opre.2019.1887.

15. Altahir, A. A., & Khalid, N. A. (2021). Drone trajectory optimization for maximum coverage in surveillance missions. *IEEE Access, 9*, 124568–124580. https://doi.org/10.1109/ACCESS.2021.3088791.

16. Cormen, T. H., Leiserson, C. E., Rivest, R. R., & Stein, C. (2022). Introduction to algorithms (4th ed.). Massachusetts Institute of Technology. https://surl.li/ltcqew.

## References

1. Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems, 61*(12), 1258–1276. https://doi.org/10.1016/j.robot.2013.09.004.

2. Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots, 31*(4), 299–316. https://doi.org/10.1007/s10514-011-9231-8.

3. Dhillon, S. S., & Smith, J. (2003). Wireless communication coverage using autonomous UAVs. *IEEE Transactions on Aerospace and Electronic Systems, 39*(2), 152–165. https://doi.org/10.1109/TAES.2003.1205972.

4. Hu, J., Zhang, L., & Wang, J. (2020). Drone-based environmental monitoring: An overview of recent developments. *Remote Sensing, 12*(4), 687. https://doi.org/10.3390/rs12040687.

5. Stachniss, C., Leonard, J. J., & Thrun, S. (2008). Simultaneous localization and mapping (SLAM): A survey of current trends. *IEEE Transactions on Robotics, 24*(5), 1050–1067. https://doi.org/10.1109/TRO.2008.2005913.

6. Isler, V., & Bajcsy, R. (2005). The sensor selection problem for surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics, 35*(4), 639–648. https://doi.org/10.1109/TSMCB.2005.850150.

7. Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research, 30*(7), 846–894. https://doi.org/10.1177/0278364911406761.

8. LaValle, S. M. (2006). Planning algorithms. Cambridge University Press. https://doi.org/10.1017/CBO9780511546877.

9. Richards, A., & How, J. P. (2002). Aircraft trajectory planning with collision avoidance using mixed-integer linear programming. *AIAA Guidance, Navigation, and Control Conference, 1*(1), 221–238. https://doi.org/10.2514/6.2002-4588.

10. Lin, L., Wu, Y., & Duan, H. (2018). UAV swarm intelligence for cooperative search and tracking. *IEEE Transactions on Cybernetics, 48*(3), 891–904. https://doi.org/10.1109/TCYB.2017.2671359

11. Evers, L., & Farinelli, A. (2019). UAV path planning under uncertainty for maximum information collection. *Journal of Field Robotics, 36*(2), 125–145. https://doi.org/10.1002/rob.21800.

12. Penicka, R., Saska, M., & Thomas, J. (2017). UAV surveillance of multiple moving targets using trajectory optimization. *Autonomous Robots, 41*(5), 1061–1078. https://doi.org/10.1007/s10514-017-9614-3.

13. Smith, R. G., & Becker, K. (2017). Real-time UAV mission planning using metaheuristic optimization techniques. *Journal of Aerospace Engineering, 31*(3), 04017085. https://doi.org/10.1061/(ASCE)AS.1943-5525.000087.

14. Bazgan, C., Hugot, H., & Vanderpooten, D. (2019). Efficient algorithms for optimizing UAV surveillance trajectories. *Operations Research, 67*(4), 952–969. https://doi.org/10.1287/opre.2019.1887.

15. Altahir, A. A., & Khalid, N. A. (2021). Drone trajectory optimization for maximum coverage in surveillance missions. *IEEE Access, 9*, 124568–124580. https://doi.org/10.1109/ACCESS.2021.3088791.

16. Cormen, T. H., Leiserson, C. E., Rivest, R. R., & Stein, C. (2022). Introduction to algorithms (4th ed.). Massachusetts Institute of Technology. https://surl.li/ltcqew.

Отримано 20.03.2025

*UDC 004.02:519.854*

***Olena Zhdanova[1], Liudmyla Rybachuk[2], Veronika Riabchun[3], Taras Maliarchuk[4]***

[1] PhD in Technical Sciences, Docent, Associate Professor of the Department of Information Systems and Technologies
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)
**E-mail:** zhdanova.elena@hotmail.com. **ORCID:** https://orcid.org/0000-0002-8787-846X
**ResearcherID:** K-1193-2016. **Scopus Author ID:** 57208339441

[2] PhD in Physical and Mathematical Sciences, Associate Professor,
Associate Professor of the Department of Information Systems and Technologies
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)
**E-mail:** rybachuk.liudmyla@lll.kpi.ua. **ORCID ID:** https://orcid.org/0000-0002-7652-587X
**ResearcherID:** AAE-7644-2019. **Scopus Author ID:** 57226545637

[3] Student of the Department of Information Systems and Technologies,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)
**E-mail:** nikakyiv696@gmail.com

[4] Student of the Department of Information Systems and Technologies
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (Kyiv, Ukraine)
**E-mail:** tarasmalarcuk8@gmail.com

# THE PROBLEM OF MAXIMIZING THE WEIGHTED NUMBER OF OBJECTS SURVEYED BY THE DRONE

*In the context of rapid advancements in unmanned technologies, the task of efficient drone trajectory planning is gaining particular relevance. This includes, in particular, optimization of a drone's flight path to maximize coverage of target objects. The research is relevant to both civil and military applications of unmanned aerial vehicles (UAVs), including territory monitoring, search and rescue operations, and patrolling.*

*The problem under consideration consists in maximizing the weighted number of objects that a drone can survey while moving along a straight-line trajectory. Given the discrete distribution of target objects and continuous nature of the parameter space for straight lines, it becomes necessary to apply discretization to use methods of discrete optimization.*

*The aim of this paper is to develop and compare combinatorial methods for constructing a line that maximizes the total weight of covered objects. Three approaches are analyzed: a greedy algorithm based on recursive subdivision of the area, a partial search method that evaluates sectors formed between objects, and a heuristic algorithm called "Floating Line," which gradually rotates a line around the center of the area using a fixed angular step. The paper provides mathematical justification for the discretization of the space of lines and defines criteria for determining whether an object is covered by the trajectory. The computational complexity of the proposed algorithms is analyzed, and their effectiveness is evaluated experimentally.*

*Experimental results show that the greedy algorithm offers high computational speed (tens or even hundreds of times faster), but with reduced accuracy. The "Floating Line" algorithm demonstrates a good balance between accuracy and computation time, whereas the partial search method ensures the highest precision and stability, though it is the most resource-intensive. Depending on the accuracy requirements and available computational resources, the user can choose the most appropriate approach. The proposed algorithms are effective tools for improving drone mission planning in practical scenarios.*

*The problem can serve as a standalone component within more complex drone mission planning models, which additionally account for energy constraints, course corrections, zone prioritization, and the presence of obstacles. Its solution can form the basis for constructing efficient and adaptive flight paths.*

***Keywords:*** *drone; UAV; maximization of surveyed objects; trajectory planning; discretization; combinatorial optimization; greedy algorithm; partial search; heuristic method.*

*Fig.: 10. Table: 3. References: 16.*

Жданова О. Г., Рибачук Л. В., Рябчун В. С., Малярчук Т. В. Задача максимізації зваженої кількості об'єктів, обстежених дроном. *Технічні науки та технології.* 2025. № 1(39). С. 134-150.