DOI: https://doi.org/10.36910/4293-52779-2025-17-02-02
УДК 519.8: 004.6: 004.9

Свирид Ю. В.
магістрант
Гуменюк Л. О.
канд. техн. наук, доцент
ORCID: 0000-0002-7678-7060
Гуменюк П. О.
канд. техн. наук, доцент
ORCID: 0000-0002-6251-8548

Луцький національний технічний університет / Україна

КЛАСИФІКАЦІЯ АЛГОРИТМІВ ПОБУДОВИ ШЛЯХУ РОБОТА З ПЕРЕШКОДАМИ

Анотація: *у роботі складено огляд та класифікацію одинадцяти алгоритмів побудови шляху робота з перешкодами, поділених на шість категорій: класичні підходи на графах (A*, Дейкстри), евристичні підходи для динамічних середовищ (D* Lite, Theta*), метаевристичні стратегії (Генетичний алгоритм, RRT), біонатхненні методи (ACO, GWO), імовірнісні алгоритми (PRM, RRT*) та гібридні алгоритми (A* з потенційними полями). Визначено, що кожен із розглянутих алгоритмів має окремі властивості, які важливі для різних класів задач планування. Описано основні характеристики та переваги кожного типу алгоритмів, що є важливим для їх програмної реалізації та порівняльного аналізу ефективності.*

Ключові слова: *алгоритм, рух з перешкодами, моделювання, аналіз даних.*

ВСТУП, ПОСТАНОВКА ПРОБЛЕМИ

Класифікація методів планування шляху за різними критеріями, такими як використання інтелектуальних технологій, тип навколишнього середовища та повнота інформації про нього, не є лише описовою. Вона відображає основні рушійні сили для розвитку алгоритмів. Наприклад, якщо алгоритм демонструє низьку продуктивність у динамічному середовищі, це створює потребу в розробці нових алгоритмів або модифікацій, які явно враховують динаміку. Аналогічно, обмежена інформація про середовище вимагає переходу від глобального планування до локального або до постійного перепланування. Це вказує на те, що обмеження алгоритмів в одній категорії (наприклад, традиційних методів для Істатичних середовищ з повною інформацією) безпосередньо стимулюють інновації та створення рішень в інших категоріях (наприклад, евристичних методів для динамічних середовищ з неповною інформацією) або гібридних підходів. Такий взаємозв'язок демонструє безперервний цикл досліджень та розробок у робототехніці, де нові виклики в реальних умовах експлуатації (наприклад, невідомі, рухомі перешкоди, вимоги до роботи в реальному часі) призводять до вдосконалення та винаходу більш надійних та адаптивних рішень для планування шляху.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є аналіз та опис класифікації та основних типів алгоритмів побудови шляху робота з перешкодами.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ ДОСЛІДЖЕННЯ

Класичні алгоритми пошуку на графах. Класичні алгоритми пошуку є основою для багатьох методів планування шляху. Вони ефективні в середовищах, де інформація про простір є повністю відомою та статичною, а середовище може бути представлене у вигляді графа або сітки.

Алгоритм Дейкстри призначений для знаходження найкоротшого шляху від однієї початкової вершини до всіх інших вершин у графі з невід'ємними вагами ребер. Його діяльність ґрунтується на ітеративному процесі: на кожному кроці обирається невідвідана вершина з найменшою поточною відстанню від початку, після чого оновлюються відстані до її сусідів. Цей процес триває доти, доки всі вершини не будуть відвідані або поки не буде знайдено шлях до цільової вершини [1].

У контексті планування шляху робота, робоче поле попередньо розбивається на клітини або вузли графа, а ребрам присвоюються значення, що відображають відстань або вартість переміщення. Перешкоди моделюються як недоступні вузли або ребра з нескінченною вагою, що ефективно виключає їх з розгляду під час пошуку шляху. Головною перевагою алгоритму Дейкстри є гарантоване знаходження найкоротшого шляху у статичних середовищах без негативних ваг ребер. Однак, його основним недоліком є висока обчислювальна вартість для великих графів, оскільки він обчислює шлях до всіх вершин, навіть якщо потрібен шлях лише до однієї цілі.

Алгоритм A* – це розширенням алгоритму Дейкстри, що поєднує його переваги з евристичною функцією для прискорення пошуку [1, 2]. Він обчислює вартість шляху від початкової точки до кінцевої, використовуючи евристичну функцію h(n), яка оцінює відстань від поточної вершини n до цільової. Загальна функція вартості f(n) визначається як сума фактичної вартості g(n) (від початкової точки до n) та евристичної оцінки h(n). Алгоритм обирає наступну вершину для розширення, яка має найменше значення f(n).

Подібно до Дейкстри, A* працює на дискретизованій карті (сітці або графі), де перешкоди позначаються як непрохідні. Евристична функція, наприклад, манхеттенська або евклідова відстань, направляє пошук до цілі, уникаючи розширення вузлів у напрямку перешкод, оскільки їх вартість буде значно вищою або нескінченною. Перевагою A* є значно вища швидкість порівняно з Дейкстрою у багатьох випадках завдяки евристиці, при цьому зберігаючи гарантію оптимальності, якщо евристика є допустимою. Проте, A* може мати проблеми з продуктивністю в реальному часі та великим обсягом обчислень на вузол, особливо в складних середовищах.

Алгоритм Флойда-Уоршелла призначений для знаходження найкоротших шляхів між усіма парами вузлів у зваженому орієнтованому графі [3]. Його робота ґрунтується на принципі динамічного програмування, де оцінки

найкоротших шляхів ітеративно покращуються шляхом розгляду можливості використання проміжних вершин.

Хоча алгоритми Дейкстри та A* знаходять шлях від однієї точки до багатьох або однієї цілі, Флойд-Уоршелл є цінним, коли необхідно швидко визначити найкоротший шлях між будь-якими двома точками в мережі. Це особливо корисно в сценаріях з кількома роботами або кількома можливими початковими/кінцевими точками, де потрібно вибрати оптимальну пару (наприклад, який БПЛА найшвидше дістанеться до цілі). Перешкоди інтегруються шляхом встановлення нескінченної ваги для ребер, що перетинають перешкоди, або шляхом початкового побудови графа, який включає лише безперешкодні зв'язки. Алгоритм Флойда-Уоршелла ефективний для задач «всі пари найкоротших шляхів» і може бути використаний для динамічного призначення ваг у моделях глибоких графів для оптимізації шляхів. Однак, його основним недоліком є висока обчислювальна складність, що робить його менш придатним для дуже великих графів або динамічних середовищ, де граф часто змінюється.

Хвильовий алгоритм (Breadth-First Search, BFS) є алгоритмом пошуку на графі, який досліджує всі вершини на поточному рівні глибини, перш ніж переходити до вершин наступного рівня. У контексті планування шляху, особливо в сіткових картах, його можна розуміти як фізичне розповсюдження хвильового фронту [1]. Від початкової клітини хвиля розповсюджується на сусідні клітини, позначаючи їх відстанню від початку.

Перешкоди в сітковій карті позначаються як непрохідні клітини, через які хвильовий фронт не може розповсюджуватися. Це забезпечує, що знайдений шлях буде обходити перешкоди. Після розповсюдження хвилі до цільової клітини, шлях відновлюється зворотним ходом від цілі до початку, слідуючи за градієнтом найменших значень. Перевагою хвильового алгоритму є гарантоване знаходження найкоротшого шляху в незважених графах (якщо всі ребра мають однакову вагу) та простота реалізації. Проте, він не враховує ваги ребер (якщо вони різні) і може бути неефективним для великих просторів, оскільки досліджує всі можливі шляхи на кожному рівні.

Евристичні алгоритми. Такі алгоритми, часто базуючись на класичних методах, впроваджують оптимізації та адаптації для підвищення ефективності, особливо у складних або динамічних середовищах.

Модифікований A* спрямований на подолання обмежень традиційного A*, таких як низька продуктивність у реальному часі, великий обсяг обчислень на вузол, тривалий час обчислення та низька ефективність пошуку. Покращення включають оптимізацію евристичних функцій та коригування відстаней до перешкод на карті [4].

Одним з ключових методів є розширення перешкод (expansion distance), що суттєво змінює діапазон чутливості мобільного робота. Це дозволяє планувати шлях з більшою відстанню до перешкод, забезпечуючи достатній простір для маневрування та підвищуючи безпеку, особливо під час поворотів. Розмір радіуса розширення зазвичай визначається радіусом самого робота. Перевагами модифікованого A* є покращена точність планування шляху, зменшення ризику зіткнень та здатність генерувати більш

плавні та реалістичні траєкторії, особливо у випадку гібридного A*. Однак, це може призвести до збільшення часу виконання та довжини шляху.

Алгоритм D* Lite є ефективним алгоритмом пошуку найкоротшого шляху у зваженому орієнтованому графі, основною особливістю якого є здатність працювати в динамічних середовищах, де структура графа не відома заздалегідь або постійно змінюється [1]. Він є ефективною модифікацією A*, що вимагає менше часу обчислень та скорочує кількість розширених вузлів.

D* Lite дозволяє роботу вільно переміщатися в невідомому та швидко змінюваному середовищі. При виявленні змін (наприклад, появи нових перешкод або зникнення старих), алгоритм ефективно перепланує шлях, адаптуючись до нової обстановки без необхідності повного перерахунку з нуля. Це робить його ідеальним для динамічних середовищ зі змінними перешкодами. Перевагами D* Lite є його ефективність у динамічних та невідомих середовищах, швидке перепланування та менші обчислювальні витрати порівняно з повним перерахунком A*. Проте, його реалізація є складнішою порівняно з A*.

Алгоритм Theta* є алгоритмом планування шляху на основі пошуку на графі, який забезпечує оптимальний шлях з більшою гнучкістю, ніж алгоритм A*, з точки зору маршрутів. Ключова відмінність від A* полягає в тому, що батьківський вузол вершини в Theta* не обов'язково має бути сусіднім, якщо між ними існує пряма видимість (line-of-sight) [5]. Це дозволяє генерувати «будь-які кути» (any-angle) шляхів, що є більш реалістичним для руху роботів, ніж шляхи, обмежені сіткою.

Theta* використовує механізм перевірки видимості, щоб «зламати» обмеження растрового середовища. Це дозволяє роботу планувати шлях, який не проходить через вузли сітки, а замість цього може «зрізати кути» навколо перешкод, якщо пряма лінія до батьківського вузла є вільною. Це допомагає уникнути надмірної кількості точок повороту та гострих кутів, які є проблемою для традиційних сіткових алгоритмів. Перевагами Theta* є генерація плавніших та коротших шляхів порівняно з A* на сіткових картах та більш реалістичні траєкторії для мобільних роботів. Однак, традиційний Theta* може бути складним для одночасного врахування глобальних та детальних аспектів планування, а також може обходити більше вузлів, що призводить до значного обсягу обчислень, особливо зі збільшенням розміру карти [6].

Еволюція від A* до модифікованого A* та Theta* демонструє критичну зміну від чистої оптимальності найкоротшого шляху до якості шляху (плавності, запасу безпеки) як першочергової мети в плануванні шляху роботів. Для фізичних роботів математично «найкоротший» шлях (наприклад, зигзагоподібний на сітці) може бути непрактичним, енергетично неефективним або навіть небезпечним через кінематичні обмеження, граничні можливості приводів або шуми сенсорів. Акцент зміщується від чисто геометричного найкоротшого шляху до «здійсненного та безпечного» шляху, який мінімізує втрати енергії, знос та ризик зіткнень.

Метаевристичні алгоритми. Метаевристичні алгоритми – це оптимізаційні підходи, які не гарантують знаходження глобального оптимуму, але є дуже

ефективними для пошуку «достатньо хороших» рішень у складних, багатовимірних або нелінійних просторах, де традиційні методи можуть бути неефективними.

Генетичні алгоритми (ГА) є метаевристикою, натхненною процесом природного відбору, що належить до ширшого класу еволюційних алгоритмів [7]. Вони працюють шляхом ітеративної модифікації популяції індивідуальних рішень, де кожне рішення (хромосома) представляє можливий шлях для робота. Шляхи оцінюються за допомогою функції вартості (фітнесу), яка враховує такі параметри, як відстань, кількість змін напрямку та зіткнення з перешкодами [8]. Найкращі рішення (з найменшою вартістю) вибираються для «розмноження» за допомогою генетичних операторів, таких як схрещування та мутація, для створення нового покоління [7, 8].

Обробка перешкод є критичною частиною функції вартості. Якщо будь-яка точка запропонованого шляху збігається з позицією перешкоди, вартість цієї хромосоми різко збільшується, що значно зменшує її ймовірність бути обраною для наступного покоління. ГА можуть ефективно планувати шлях як зі статичними, так і з динамічними перешкодами. Перевагами ГА є здатність знаходити оптимальні шляхи у складних, динамічних середовищах з непередбачуваними перешкодами та гнучкість у визначенні функції фітнесу для врахування різних критеріїв (довжина, плавність, безпека). Проте, їх недоліками є повільна збіжність, схильність до локальних оптимумів, невизначена спрямованість мутації. Стохастична природа може призвести до непослідовних результатів, що вимагає багаторазових запусків.

Метод потенційних полів передбачає рух мобільного пристрою вздовж ліній векторного поля [1]. Ціль створює притягуюче потенційне поле, а перешкоди – відштовхуюче. Сила відштовхування зменшується зі збільшенням відстані до перешкоди. Сума цих векторів (притягування до цілі та відштовхування від перешкод) визначає напрямок руху робота.

Перешкоди моделюються як джерела відштовхувальних сил, які «виштовхують» робота з небезпечних зон. Для функціонування в динамічному середовищі, сума векторів перераховується через короткі проміжки часу на основі показань сенсорів, що дозволяє роботу реагувати на рухомі перешкоди. Перевагами алгоритму заливного поля є простота реалізації та ефективність у реальному часі для уникнення перешкод. Однак, його основним недоліком є локальність підходу, що часто робить його непридатним для досягнення комплексних цілей через схильність до потрапляння в локальні мінімуми або зависання в «пастках».

Метод RRT (Rapidly-exploring Random Tree) полягає в швидкому дослідженні простору конфігурацій шляхом побудови дерева опорних точок, яке послідовно розширюється від початкової до кінцевої точки [9]. Процес починається зі стартового дерева, що складається лише з початкового пункту. На кожному кроці генерується випадкова точка в області, і до дерева додається новий вузол, з'єднаний з найближчим існуючим вузлом дерева, якщо шлях між ними вільний від перешкод.

Кожен новий вузол та ребро, що додаються до дерева, проходять перевірку на зіткнення з перешкодами. Якщо виявляється зіткнення, точка

відкидається або ребро замінюється. RRT ефективний для навігації в складних, багатовимірних просторах з перешкодами, оскільки він не вимагає попереднього обчислення всього простору конфігурацій. Він може бути використаний для перепланування в динамічних середовищах, коли перешкоди блокують бажаний шлях. Перевагами RRT є висока ефективність для задач з великою кількістю вимірів, нелінійними динаміками або диференціальними обмеженнями. Він не потребує попереднього обчислення простору конфігурацій та здатний швидко знаходити шлях у складних середовищах. Проте, він не гарантує оптимальності шляху і може бути неефективним у складних середовищах, таких як сади, через рівномірну випадкову вибірку, яка може сповільнювати збіжність.

Біонатхненні алгоритми. Біонатхненні алгоритми черпають натхнення з природних процесів та колективної поведінки організмів для вирішення оптимізаційних задач, включаючи планування шляху.

Мурашиний алгоритм (ACO) імітує поведінку мурах, що шукають їжу. Мурахи залишають феромонні сліди на шляхах, які вони проходять, і інші мурахи з більшою ймовірністю слідують шляхами з вищою концентрацією феромонів [10]. З часом феромони випаровуються, дозволяючи алгоритму адаптуватися до змін у середовищі [11].

У плануванні шляху, «мурахи» досліджують простір, залишаючи «феромони» на пройдених шляхах. Шляхи, що уникають перешкод і є коротшими, отримують більше феромонів, приваблюючи більше «мурах». Покращені версії ACO використовують такі модифікації, як покращена евристична функція (включення штучного потенційного поля для притягання до цілі), адаптивні правила оновлення феромонів та метод обрізки трикутників для усунення зайвих точок повороту, що робить шляхи плавнішими та ефективнішими. Перевагами ACO є ефективність для пошуку оптимальних шляхів у складних графах та здатність адаптуватися до змін у середовищі. Проте, його недоліками є повільна швидкість збіжності, схильність до потрапляння в локальні оптимуми та надмірна кількість точок повороту в традиційній реалізації.

Алгоритм рою часток (PSO) – це оптимізаційний метод, натхненний соціальною поведінкою рою птахів або косяків риб [12]. Кожна «частка» (потенційне рішення) переміщується в просторі пошуку, коригуючи свою швидкість та позицію на основі власного найкращого досвіду (pbest) та найкращого досвіду всього рою (gbest).

У плануванні шляху, частки представляють можливі траєкторії. Якщо позиція частки знаходиться всередині перешкоди, цей шлях відкидається. Перешкоди можуть бути змодельовані як розширені області (збільшені на радіус робота), щоб забезпечити безпечний прохід.PSO використовує функцію фітнесу, яка мінімізує довжину шляху та час подорожі, уникаючи зіткнень. Перевагами PSO є простота, легкість реалізації, надійність, швидка збіжність. Він ефективний для пошуку глобальних оптимумів у нелінійних задачах. Проте, може застрягти в локальних оптимумах у дуже складних ландшафтах фітнесу.

Алгоритм бджолиної колонії (ABC) – це потужна техніка оптимізації,

натхненна поведінкою медоносних бджіл під час пошуку їжі [13]. Він використовує популяційний підхід, де штучні бджоли (розвідники, робітники, спостерігачі) співпрацюють для пошуку оптимальних рішень у просторі пошуку, балансуючи між дослідженням (exploration) та експлуатацією (exploitation).

У плануванні шляху, бджоли шукають «джерела їжі» (потенційні шляхи), оцінюючи їх «якість». Шляхи, що уникають перешкод та є коротшими/ефективнішими, розглядаються як кращі джерела їжі. ABC може оптимізувати траєкторії роботів у складних середовищах з перешкодами, знаходячи найкоротші або найефективніші шляхи, уникаючи зіткнень. Він також може працювати в динамічних середовищах з рухомими перешкодами. Перевагами ABC є ефективність для складних оптимізаційних задач та здатність балансувати дослідження та експлуатацію. Він застосовується для багатоцільового планування шляху. Однак, може мати слабку здатність до експлуатації та низьку точність рішення в деяких випадках, хоча модифікації, такі як FOABC, спрямовані на їх подолання.

Алгоритм сірих вовків (GWO) – це алгоритм ройового інтелекту, що імітує механізм полювання сірих вовків [14]. Він поділяє вовків на ієрархію: альфа (лідер), бета (помічник), дельта (розвідник) та омега (решта зграї). Полювання включає три етапи: оточення, переслідування та напад, де позиції вовків оновлюються на основі позицій альфа, бета та дельта вовків.

У плануванні шляху, «вовки» шукають оптимальну траєкторію. Перешкоди можуть бути розширені для забезпечення безпечної відстані. Покращені версії GWO, такі як Golden Sine Grey Wolf Optimizer (GSGWO), вирішують проблеми повільної збіжності та потрапляння в локальні оптимуми. GSGWO може адаптуватися до функції перетину перешкод, дозволяючи роботу робити вибір між перетином перешкоди (з певною вартістю) та її уникненням. Перевагами GWO є сильна продуктивність збіжності та відносно проста структура алгоритму. Він здатний оптимізувати довжину шляху, кількість точок повороту та плавність. Проте, схильний до потрапляння в локальні оптимуми, має повільну збіжність на пізніх етапах та проблему з єдиною початковою популяцією в оригінальному GWO.

Алгоритм кажана (BA) – це біонатхненний алгоритм, заснований на ехолокаційних (біо-сонарних) здібностях кажанів. Кажани випромінюють звукові імпульси та слухають ехо, що повертається від перешкод або здобичі, використовуючи цю інформацію для навігації [15]. У алгоритмі «штучні кажани» (потенційні позиції робота) переміщуються в просторі пошуку, оновлюючи свою частоту, швидкість та позицію на основі глобального найкращого рішення.

У плануванні шляху, алгоритм кажана шукає найкоротший шлях, уникаючи динамічних перешкод. Модифіковані версії, такі як Modified Frequency Bat (MFB) algorithm, балансують дослідження та експлуатацію. Обробка перешкод відбувається за допомогою віртуальних сенсорів, які виявляють перешкоди, та процедури уникнення перешкод, яка використовує концепцію «вектора проміжку» (gap vector) для вибору вільного шляху. Алгоритм може оцінювати траєкторію рухомих перешкод для їх уникнення. Перевагами є ефективність

для пошуку найкоротшого шляху в динамічних середовищах з рухомими перешкодами та здатність адаптуватися до змін у середовищі. Проте, оригінальний BA може застрягти в локальних оптимумах.

Алгоритм зозулі (CS) – це метаевристичний алгоритм, натхненний стратегією гніздового паразитизму зозуль [16]. Кожна «зозуля» (потенційне рішення) відкладає яйце (пропонує рішення) у випадково вибране гніздо господаря. Рішення покращуються за допомогою механізму Levy flight для глобального пошуку та випадкового блукання для локального дослідження.

У плануванні шляху, початкова популяція позицій генерується випадковим чином. Функція оцінки (фітнесу) враховує довжину шляху та його плавність. Перешкоди моделюються у сітковому середовищі як непрохідні області. Хоча прямих механізмів активного уникнення перешкод у базовому CS не описано, функція фітнесу неявно відштовхує алгоритм від перешкод, штрафуючи шляхи, які їх перетинають. Перевагами є проста структура, мінімальна кількість параметрів, легкість реалізації та надійність. Він ефективний для складних оптимізаційних задач. Проте, має недостатнє покриття простору пошуку, передчасну збіжність, низьку точність пошуку та повільну швидкість пошуку в традиційному CS.

Ймовірнісні алгоритми. Імовірнісні алгоритми, зокрема методи на основі вибірки, ефективні для планування шляху у високовимірних просторах конфігурацій, де побудова явного графа є непрактичною.

PRM (Probabilistic Roadmap) – це метод, що описує зв'язність вільного простору для переміщення за допомогою графів [1]. Він працює шляхом випадкової вибірки точок у просторі конфігурацій робота та з'єднання їх ребрами, якщо пряма лінія між ними вільна від перешкод. Таким чином, будується «дорожня карта» (граф) доступних шляхів.

Перешкоди враховуються на етапі побудови дорожньої карти, коли ребра додаються до графа лише в тому випадку, якщо вони не перетинаються з перешкодами. Це забезпечує, що всі шляхи, знайдені на дорожній карті, є безперешкодними. Метод покладається на повноту інформації про навколишнє середовище, що створює труднощі при його застосуванні в умовах динамічного середовища. Перевагами PRM є ефективність для планування шляху у високовимірних просторах, де явне представлення перешкод є складним, та здатність знаходити шлях у складних середовищах. Проте, він може бути неефективним у динамічних середовищах, оскільки дорожня карта повинна бути перебудована при зміні перешкод, і не гарантує оптимальності шляху.

RRT* (Optimal RRT) є покращеною версією алгоритму RRT, яка, на відміну від базового RRT, гарантує оптимальність шляху при достатній кількості зразків [17, 18]. RRT* працює за принципом RRT, розширюючи дерево від початкової точки шляхом випадкової вибірки точок. Однак, RRT* додає два ключові кроки: перепідключення (rewiring) та вибір найкращого батька (best parent selection). При додаванні нового вузла, RRT* перевіряє, чи можна досягти сусідніх вузлів через новий вузол з меншою вартістю, і якщо так, перепідключає їх.

Як і RRT, RRT* виконує перевірку на зіткнення для кожного нового вузла та

ребра, забезпечуючи, що побудований шлях є вільним від перешкод. Перепідключення також враховує перешкоди, забезпечуючи, що нові, оптимізовані зв'язки також є безперешкодними. Перевагами є гарантована оптимальність шляху при достатній кількості зразків, а також зменшення довжини шляху та покращення плавності порівняно з RRT. Проте, RRT* має вищі обчислювальні витрати порівняно з базовим RRT через додаткові кроки оптимізації і може бути неефективним у дуже складних середовищах, де випадкова вибірка може сповільнювати збіжність.

Сила RRT полягає в його «швидкому дослідженні» та здатності швидко знаходити будь-який здійсненний шлях у високовимірних просторах, де інші методи зазнають невдачі. Це вирішує проблему «повноти» (знаходження шляху). Після того, як шлях знайдено, наступним логічним кроком є його оптимізація. RRT* додає крок «перепідключення», який ітеративно покращує оптимальність шляху. Це передбачає двоетапний підхід до вирішення проблеми: спочатку знайти рішення (здійсненність), потім його оптимізувати.

Гібридні алгоритми. Гібридні алгоритми поєднують сильні сторони декількох підходів для подолання їх індивідуальних обмежень, створюючи більш надійні та ефективні рішення.

A* + потенційні поля. Цей гібридний підхід поєднує глобальні пошукові можливості алгоритму A* з реактивними можливостями уникнення перешкод, які надають потенційні поля [2, 19, 20]. A* використовується для глобального планування шляху, забезпечуючи оптимальний або майже оптимальний маршрут від початку до кінця. Потенційні поля, зі своїми силами притягання до цілі та відштовхування від перешкод, застосовуються для локального уникнення перешкод у реальному часі та згладжування траєкторії.

Гібридні алгоритми, такі як MAAPF (Modified A* and Artificial Potential Field), можуть ефективно працювати в динамічних середовищах. A* планує глобальний шлях, а коли робот виявляє динамічні перешкоди, система переходить до локального планування, використовуючи потенційні поля та прогнозування траєкторії перешкод. Це дозволяє роботу адаптуватися до непередбачених змін у середовищі та уникати зіткнень у реальному часі. Перевагами є подолання обмежень окремих алгоритмів: A* допомагає уникнути локальних мінімумів потенційних полів, а потенційні поля забезпечують плавність та реактивність, яких бракує традиційному A*. Це генерує безпечніші, плавніші та більш реалістичні шляхи. Проте, недоліками є складність інтеграції та налаштування, оскільки необхідно правильно збалансувати вплив глобального планувальника та локального контролера. Гібридний A* може мати довший час виконання у порівнянні з базовим A*.

Поширеність гібридних алгоритмів, особливо тих, що поєднують глобальний пошук (як A*) з локальними реактивними методами (як потенційні поля або DWA), свідчить про перехід до ієрархічних та надійних архітектур планування шляху, які враховують як довгострокову оптимальність, так і адаптивність у реальному часі.

Потенційні поля або підхід динамічного вікна (DWA) відмінно підходять для локального, реактивного уникнення в реальному часі та згладжування шляху, але страждають від локальних мінімумів та відсутності глобальної обізнаності.

Їхнє поєднання створює ієрархічну систему, де A* надає «макро» план, а реактивний метод обробляє «мікро» коригування та уникнення перешкод. Ця архітектурна тенденція є вирішальною для автономних систем, що працюють у складних, непередбачуваних реальних середовищах (наприклад, безпілотні автомобілі, сільськогосподарські роботи, сервісні роботи). Вона свідчить про те, що майбутні рішення для планування шляху все частіше включатимуть складні багатошарові системи, які інтегрують різноманітні алгоритмічні парадигми.

ВИСНОВКИ

Кожен із розглянутих алгоритмів має окремі властивості, які важливі для різних класів задач планування.

A* і алгоритм Дейкстри представляють класичні підходи на графах. A* має здатність працювати з евристиками, що дає змогу зменшити обчислювальні витрати, зберігаючи оптимальність. Алгоритм Дейкстри – це еталонний метод для гарантії точності в умовах фіксованих ваг. Обидва алгоритми є добре формалізованими, що полегшує їх програмну реалізацію в системах зі статичним середовищем.

D* Lite і Theta* демонструють евристичний підхід до задач у динамічних середовищах. D* Lite дозволяє адаптувати маршрут у реальному часі без повного перерахунку, що є ключовим для роботів, які взаємодіють із рухомими об'єктами. Theta* формує більш гладкі траєкторії, що важливо для фізично реалізованих роботів. Обидва методи придатні до інтеграції в робототехнічні системи, де потрібна швидка реакція на зміну оточення.

Генетичний алгоритм і RRT репрезентують метаевристичні стратегії, де ціль – обхід складних або великорозмірних просторових конфігурацій. ГА дозволяє досліджувати простори з багатьма локальними мінімумами, а RRT забезпечує швидке охоплення великих областей. Їхня гнучка структура робить їх придатними до модифікацій для конкретних задач і середовищ.

ACO і GWO як біонатхненні методи показують ефективність у задачах з множинними маршрутами та змінними умовами. ACO моделює розподілену оптимізацію через феромони, а GWO – ієрархічну стратегію пошуку, яка спрощує реалізацію. Обидва алгоритми добре масштабуються і можуть адаптуватися до змін топології середовища.

PRM і RRT* – ймовірнісні алгоритми, що дозволяють працювати у високовимірних просторах. PRM будує граф можливих шляхів, ефективний для багаторазового використання в одному середовищі. RRT* покращує шлях з часом, що цінно при обмеженнях на ресурси. Ці алгоритми легко комбінуються з іншими для досягнення балансу між швидкістю і якістю.

Гібридний алгоритм, який поєднує A* з методом потенційних полів, забезпечує як глобальне планування, так і локальне уникнення перешкод. Така комбінація дозволяє створювати адаптивні маршрути з урахуванням нових перешкод у режимі реального часу.

ЛІТЕРАТУРА

[1] Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of

Mobile Robot. URL: https://pmc.ncbi.nlm.nih.gov/ articles/PMC11175283/ (дата звернення: 15.07.2025).
[2] Path planning diagram. URL: https://www.researchgate.net/figure/Path-planning-diagram_fig1_366333455 (дата звернення: 15.07.2025).
[3] How can i use particle swarm optimisation algorithm for to find optimal path interms of shortest distance between start and goal point to be followed by mobile robot? URL: https://www.mathworks.com/matlabcentral/answers/263094-how-can-i-use-particle-swarm-optimisation-algorithm-for-to-find-optimal-path-interms-of-shortest-dis (дата звернення: 15.07.2025).
[4] Obstacle-Free Path Planning for Autonomous Drones Using Floyd's Algorithm. URL: https://www.arxiv.org/pdf/2409.13149v1 (дата звернення: 15.07.2025).
[5] Theta* algorithm path planning schematic. URL: https://surl.lu/nayurk (дата звернення: 15.07.2025).
[6] Modified A* algorithm for path smoothing and obstacle avoidance. URL: https://www.ewadirect.com/proceedings/ace/article/view/9971/pdf (дата звернення: 15.07.2025).
[7] Genetic Algorithm-based Robot Path Planning. URL: https://repo.pens.ac.id/188/1/c3d14cd2e419cde877993555eeef.pdf (дата звернення: 15.07.2025).
[8] The Path Planning of Mobile Robots Based on an Improved Genetic Algorithm. URL: https://surl.lu/mxhour (дата звернення: 15.07.2025).
[9] HPS-RRT*: An Improved Path Planning Algorithm for a Nonholonomic Orchard Robot in Unstructured Environments. URL: https://www.mdpi.com/2073-4395/15/3/712 (дата звернення: 15.07.2025).
[10] Path-Planning Strategy: Adaptive Ant Colony Optimization Combined with an Enhanced Dynamic Window Approach. URL: https://www.mdpi.com/2079-9292/13/5/825 (дата звернення: 15.07.2025).
[11] Optimizing Robot Path Planning with the Particle Swarm Optimization Algorithm. URL: https://surl.li/ohjkdm (дата звернення: 15.07.2025).
[12] Artificial bee colony algorithm. URL: https://surli.cc/bnzjrg (дата звернення: 15.07.2025).
[13] Multi-objective path planning for mobile robot with an improved artificial bee colony algorithm. URL: https://pubmed.ncbi.nlm.nih.gov/36899544/ (дата звернення: 15.07.2025).
[14] Path Planning and Obstacle Avoidance of a Mobile Robot based on GWO Algorithm. URL: https://surl.li/zmmmpg (дата звернення: 15.07.2025).
[15] Robot Motion Control using Dual Avoidance Scheme. URL: https://www.espublisher.com/journals/articlehtml/engineered-science/10.30919-es1261 (дата звернення: 15.07.2025).
[16] Cuckoo Search Algorithm using Lèvy Flight: A Review. URL: https://www.researchgate.net/publication/269651786_Cuckoo_Search_Algorithm_using_Levy_Flight_A_Review (дата звернення: 15.07.2025).
[17] Wang, H.; Wang, S.; Yu, T. Path Planning of Inspection Robot Based on Improved Ant Colony Algorithm. Appl. Sci. 2024, 14, 9511. URL: https://doi.org/10.3390/app14209511 (дата звернення: 15.07.2025).

[18] Hybrid potential field based control of differential drive mobile robots. URL: https://chaoslab.unm.edu/MaterialDownload/Hybrid%20potential%20Field%20based%20control%20of%20differential%20drive%20mobile%20robots.pdf (дата звернення: 15.07.2025).
[19] Optimal Path Planning using RRT for Dynamic Obstacles. URL: https://nopr.niscpr.res.in/bitstream/123456789/54909/1/JSIR%2079(6)%20513-516.pdf (дата звернення: 15.07.2025).
[20] Path Tracking Hybrid A* For Autonomous Agricultural Vehicles. URL: https://arxiv.org/html/2411.14086v1 (дата звернення: 15.07.2025).
[21] Byari M., Bernoussi A., Jellouli O., Ouardouz M., Amharref M. Multi-scale 3D cellular automata modeling: Application to wildland fire spread. Chaos Solitons Fractals, 2022, V. 164, pp. 112-153.
[22] A Mathematical Model for Predicting Fire Spread in Wildland Fuels by Richard C. Rothermel. URL: https://www.fs.usda.gov/rm/pubs_int/int_rp115.pdf (дата звернення: 15.07.2025).

CLASSIFICATION OF ALGORITHMS FOR CONSTRUCTING A PATH WORKING WITH OBSTACLES

Svyryd, Yu. *Undergraduate*
Gumeniuk, L *PhD, Associate Professor*
Humeniuk, P. *PhD, Associate Professor*
Lutsk National Technical University / Ukraine

Abstract. The paper provides an overview and classification of eleven algorithms for constructing a robot path with obstacles, divided into six categories: classical graph-based approaches (A*, Dijkstra), heuristic approaches for dynamic environments (D* Lite, Theta*), metaheuristic strategies (genetic algorithm, RRT), bio-inspired methods (ACO, GWO), probabilistic algorithms (PRM, RRT*), and hybrid algorithms (A* with potential fields). It has been determined that each of the algorithms considered has distinct properties that are important for different classes of planning problems. The main characteristics and advantages of each type of algorithm are described, which is important for their software implementation and comparative analysis of effectiveness.

Keywords: algorithm, robot path, obstacle avoidance, simulation, data analysis.

Дата першого надходження статті до видання	Дата прийняття статті до друку статті після рецензування	Дата оприлюднення
16.09.2025 р.	16.10.2025 р.	23.12.2025 р.