DOI: 10.58254/viti.4.2023.02.16

УДК 621.396

Бернацький А. П. ORCID: 0000-0003-0397-075X (ВІТІ ім. Героїв Крут)

МЕТОД ПЛАНУВАННЯ ШЛЯХУ АВТОНОМНОГО НАЗЕМНОГО РОБОТА З ВИКОРИСТАННЯМ МОДИФІКАЦІЇ ДИНАМІЧНОГО ДВОНАПРАВЛЕНОГО RRT-АЛГОРИТМУ

Проблема планування шляху безпілотних автономних наземних засобів завжди була гострою проблемою в галузі досліджень автономних наземних робототехнічних систем. З погляду на складність завдань під час проведення бойових дій в урбанізованому просторі щільної забудови міста з постійною зміною ландшафту, завдання, які покладаються на безпілотні автономні наземні засоби, постійно ускладнюються, а сценарії використання безпілотних автономних наземних засобів демонструють диверсифіковану тенденцію розвитку нових систем прийняття рішень безпілотних автономних наземних засобів. Швидкоплинність ведення сучасних бойових дій в урбанізованому просторі постає як складне і багатопланове завдання. Враховуючи складність процесу руху безпілотних автономних наземних засобів, на сучасному етапі розвитку робототехнічних систем, загальною тенденцією є відмова від дистанційного керування роботизованими комплексами з переходом до автоматичних режимів, що вимагає розробки, впровадження алгоритмів автоматичної взаємодії та руху військових мобільних робототехнічних систем. Задля вирішення проблеми пришвидшення отримання рішень пошуку шляхів без зіткнень у режимі реального часу у двовимірному просторі пропонується застосувати модифікований динамічний двонаправлений алгоритм «швидке дослідження випадкового дерева із зірочкою» з реперними вузлами. Запропонований алгоритм є модифікацією алгоритму «швидке дослідження випадкового дерева із зірочкою» з реперними вузлами, з використанням методу двонаправленого жадібного пошуку для прискорення та вирішення проблеми односпрямованого алгоритму «швидке дослідження випадкового дерева» щодо його повільної швидкості пошуку, а також труднощів прийняття рішення у вузькому середовищі, викликаних сліпою випадковою вибіркою. В разі динамічного переміщення перешкоди така модифікація завдяки перевазі, що реперні вузли не потребують багато обчислень під час планування, у процесі ітераційної оптимізації шляху виконує оновлення інформації карти в режимі реального часу та відновлює пошкоджений вихідний шлях для завершення динамічного планування шляху.

Ключові слова: *автономний наземний робот, вибірковий алгоритм, швидке дослідження випадкового дерева, планування шляху в реальному часі, двонаправлений жадібний пошук, відновлення шляху.*

A. Bernatskyi Method for planning the way of UGV using a modification of dynamic bi-directional RRT algorithm.

*The path planning problem of unmanned autonomous ground vehicles has always been an acute problem in the field of autonomous ground robotic systems research. From the point of view of the complexity of the tasks during the conduct of combat operations in the urbanized space of the densely built-up city with a constantly changing landscape, the tasks entrusted to the UGV are constantly becoming more difficult, and the scenarios of the use of the UGV demonstrate a diversified trend in the development of new management and control systems. The rapidity of conducting modern military operations in an urbanized space appears as a complex and multifaceted task. Considering the complexity of the UGV movement process, at the current stage of the development of robotic systems, the general trend is to abandon remote control of robotic complexes with the transition to automatic modes, which requires the development and implementation of algorithms for automatic interaction and movement of military mobile robotic systems. To solve the problems of the Rapidly-Exploring Random Tree *Fixed Nodes algorithm regarding its low speed for obtaining track junctions and the impossibility of using it in a dynamic environment when planning a UGV path. To solve the problem of accelerating the acquisition of collision-free paths in real time in two-dimensional space, it is proposed to apply a modified dynamic bidirectional RRT* algorithm with reference nodes. The algorithm is a modification of Rapidly-Exploring Random Tree *Fixed Nodes using a bidirectional greedy search method to speed up and solve the problem of the unidirectional Rapidly-Exploring Random Tree algorithm regarding its slow search speed, as well as the difficulties of decision making in a narrow environment caused by blind random sampling. In the case of dynamic disturbance movement, taking advantage that reference nodes do not require much calculation in planning, in the process of iterative path optimization, the proposed algorithm updates the map information in real time and repairs the damaged output path to complete the dynamic path planning.*

Keywords: *UGV, Rapidly-Exploring Random Tree, selective algorithm, way real-time planning, bi-directional greedy search, way recovery.*

Постановка завдання

Планування шляху безпілотних автономних наземних засобів (далі – БАНЗ) завжди було гострою проблемою в галузі досліджень автономних наземних робототехнічних систем [1]. З погляду на складність завдань під час проведення бойових дій в урбанізованому просторі щільної забудови міста з постійною зміною ландшафту, завдання, які покладаються на БАНЗ, постійно ускладнюються, а сценарії використання БАНЗ демонструють диверсифіковану тенденцію розвитку [2], що вимагає розробки військових мобільних робототехнічних систем із системами прийняття рішень, здатними на самостійне (автономне) вирішення проблем пошуку шляху.

Використання математичного апарату алгоритмів планування шляху надає якісний ефект керування системою прийняття рішень БАНЗ. [3]. Проблематика побудови БАНЗ військового призначення передбачає пошук шляху без зіткнень, який з'єднує відомі початкову та цільову точку положення робота у просторовій системі, що складається з одного або групи складних геометричних тіл, задовольняючи при цьому обмеження, накладені складними перешкодами.

Тому актуальною науковою задачею є розробка ефективних алгоритмів пошуку маршруту переміщення робота без зіткнень в умовах динамічно змінних перешкод та обмеженого проходу.

Аналіз останніх публікацій

Відомо багато прикладів досліджень проблематики планування шляху роботів.

Відповідно до тенденції розвитку алгоритмів планування шляху їх можна розділити на два основні класи: алгоритми на основі графів та алгоритми на основі вибірки.

Найбільш відомі серед алгоритмів планування шляху на основі графів містять алгоритм A*, алгоритм Дейкстри, Флойда-Воршелл, Пріма та інші [4]. Традиційні алгоритми планування шляху не мають можливостей коригування шляху в умовах динамічного змінного середовища. Для того, щоб алгоритм пошуку шляху відповідав цілям планування реалізації БАНЗ, Йоав Фройнд и Роберт Шапире [5] одні з перших запропонували динамічний алгоритм A*, якій можна застосовувати в динамічному середовищі у реальному часі. Пізніше Айе Маув [6] розробив оптимізований варіант динамічного алгоритму iADA*-RL. Кадрі та інші [7] представили модифіковану допоміжну функцію керування та посібник з оптимізації функції оцінювання D*.

Завдяки використанню оптимальної роздільної здатності мапи та цілісності картини поля мапи [8], алгоритм пошуку шляху на графах може знайти оптимальне рішення пошуку шляху за відносно малий час. Але такі алгоритми вимагають растрового моделювання карти, що впливає на ступень універсальності і вимагає використання додаткових обчислювальних ресурсів для конвертації. Алгоритми пошуку за графами використовуються в растровому просторі, щоб зменшити загальну кількість станів, а потім визначити мінімальний шлях, складений з набору вузлів без колізій [9]. Враховуючи вище сказане, зауважимо, що *алгоритми планування шляху на основі графа* не підходять для ситуацій, які вимагають планування в реальному часі.

Сучасні методи планування шляху робота на основі вибірки переважно містять такі алгоритми, як ймовірнісні дорожні карти (RPM), алгоритм швидкого дослідження випадкового дерева (Rapidly exploring Random Tree, RRT) [10] та інші. Ці алгоритми засновано на інкрементній випадковій вибірці ефективного планування можливих шляхів. Алгоритм планування шляху, заснований на випадковій вибірці, уникає моделювання простору та швидко отримує шляхи. Він підходить для випадків, коли потрібна продуктивність у реальному часі. Однак спосіб генерації точок вибірки є випадковим. Шлях, отриманий RRT, зазвичай не є оптимальним, тому потрібна процедура оптимізації шляху. Такі алгоритми, як RRT і RRT*, запроваджують методи повторного вибору батьківського вузла, повторного

підключення вузла та асимптотично збільшують цільовий шлях завдяки збільшенню точок вибірки. У випадку коли точки вибірки прагнуть до нескінченності, завдяки оптимізації, може бути отриманий оптимальний шлях [11].

Однак процес додавання точок вибірки для отримання асимптотично оптимального шляху є відносно довгим. Для покращення швидкості збіжності рішення шляху відомо про покращені версії цього алгоритму. Гаммелл та ін. [12] запропонували алгоритм informedRRT*, якій змінює вибір простору вибірки і обмежує глобальну випадкову вибірку еліптичним простором. Тагері та інші [13] запропонували алгоритм пошуку, заснований на нечіткому жадібному RRT (FG-RRT), який генерує вагу вузлів-нащадків шляхом оцінки зміни якості. Адіятов та ін. [14] запропонували алгоритм пошуку Quick-RRT* на основі подвійного дерева. Алгоритм планування шляху вирощує два дерева по черзі від початкової та кінцевої точки для покращення швидкості конвергенції. Оскільки кількість вузлів збільшується під час виконання процесу оптимального алгоритму RRT, обчислювальна складність і час роботи зростатимуть експоненціально до зростання. Щоб вирішити цю проблему, Spanogianopoulos та інші [15] запропонували алгоритм RRT*FN (RRT фіксовані вузли), який може зменшити надлишкові кінцеві вузли на шляху та покращити швидкість планування. Але такий алгоритм можливо використовувати тільки в умовах статичних середовищ, що також унеможливлює повною мірою використання для системи прийняття рішень БАНЗ.

Отже, існуючі рішення (алгоритми) мають певні недоліки, такі як ресурсовимогливість, час виконання завдання або неефективна побудова довжини шляху при виконанні місії в умовах динамічних середовищ..

Метою статті є розробка алгоритму, який можна застосовувати в динамічних середовищах, завдяки використанню двонаправленого жадібного пошуку для покращення часу планування алгоритму, довжини рішення шляху, швидкості конвергенції, форми шляху, задля задоволення вимог планування шляху в реальному часі, дозволяючи БАНЗ швидко отримати оптимальний шлях без зіткнень у динамічних середовищах у режимі реального часу.

Окремим питанням є перевірка енергоефективності запропонованого метода.

Виклад основного матеріалу

1. Теоретична складова модифікації алгоритму RRT*FN.

Подібно до методу модифікації алгоритму RRT, визначеного в літературі [16–18], простір станів $X \subseteq Rn$ у задачі планування шляху визначається за допомогою сенсорів БАНЗ. Інформація про перешкоди, що міститься на цифровій карті, отримується за допомогою лазерного скануючого пристрою відстані (ЛІДАР) та формується як простір перешкод $X_{obs} \subsetneq X$, а доповнення простору перешкод до простору станів визначається як вільний простір X_f, де $X_f = X_{obs} C_U X$, тобто це простір, в якому робот може вільно рухатися.

Припустимо, що початковою точкою БАНЗ у просторі станів є $p_{start} \in X_f$, а рухомою цільовою точкою є $p_{end} \in X_f$. Набір точок шляху визначаємо як $\sigma: [0;\ 1] \subset = p_{end}$. Тоді правило шляху перетворюється на розв'язання множини точок $\sigma[0;\ 1]$ у вільному просторі X_f. Якщо зв'язок між точкою $p \in \sigma[0; 1]$ і точкою p не перетинається з множиною точок $\sigma[0;\ 1]$, то визначаємо функцію вартості шляху $c(\sigma)$, що дорівнює лінії евклідової відстані, яка з'єднує набір точок рішення шляху σ (1).

$$c(\sigma) = \sum_{i=1}^{n} \left\| \sigma\left(\frac{i}{n}\right) - \sigma\left(\frac{i-1}{n}\right) \right\|, \quad (1)$$

де n – кількість елементів у множині σ, а можливий шлях σ у просторі не є єдиним. Визначимо Σ як множину всіх можливих σ, а $C: \sum \to R_{\geqslant 0}$ – множина всіх шляхів.

Для задачі рішення оптимального шляху, вартість шляху використовується як оціночний індекс для оцінки рішень шляху. Чим нижча вартість шляху тим кращий шлях, тоді задача розв'язання оптимального шляху перетворюється на розв'язання σ^*, яке мінімізує вартість шляху у вільному просторі X_f. Тобто ми маємо (2):

$$\sigma^* = argmin\{c(\sigma)|\sigma(0) = p_{start}, \sigma(1) = p_{end}, \forall s \in [0,1], \sigma(s) \in X_f\}. \quad (2)$$

Для ситуації динамічного рішення, коли положення БАНЗ і перешкоди змінюються, p_{start} і p_{end} можуть бути не унікальними, тому набір початкових точок і цільових точок шляху визначається як $P_{end}, \sigma(0) \in P_{start}, \sigma(1) \in p_{end}$. Тоді визначення (2) отримає наступний вигляд (3):

$$\sigma^* = argmin\{c(\sigma)|\sigma(0) = P_{start}, \sigma(1) = P_{end}, \forall s \in [0,1], \sigma(s) \in X_f\}. \quad (3)$$

Алгоритм RRT є найбільш типовим алгоритмом планування шляху на основі вибірки. Цей алгоритм може випадково генерувати точки вибірки з простору, використовувати початкову точку як кореневий вузол і з'єднувати лінії від найближчих вузлів вибірки без зіткнень. Напрямок зростає відповідно до заданого розміру кроку досягнення мети розширення дерева. Як тільки станеться зіткнення між з'єднаннями листового вузла та цільової точки, планування шляху буде завершено.

Основні кроки алгоритму:

Крок 1: Дано простір карти M, де p_{start} – початкова точка, а завдання цільової точки очікуються.

Крок 2: Отримаємо p_{rand} шляхом випадкової вибірки в просторі.

Крок 3: Починаємо зростати із найближчої точки перетину вузла та зростайте до p_{rand} із розміром кроку s. Новостворену точку визначаємо як p_{new}.

Крок 4. Існує певна ймовірність того, що точка вибірки не буде рости вздовж випадкової точки вибірки p_{rand}, а вибере зростання безпосередньо до кінцевої точки p_{end}.

Крок 5: Якщо дерево росте до p_{end} або лінія, що з'єднує p_{new} та p_{end}, не перетинає перешкоди, генерується можливий шлях σ.

Незважаючи на те, що алгоритм RRT має імовірнісну повноту і може швидко отримувати можливі рішення в просторі, його процес пошуку рішень є випадковим, тобто *сліпим*. Це, зокрема, відображається в тому факті, що p_{rand} отримується шляхом випадкової вибірки, роблячи напрямок росту дерева випадковим. До того ж в алгоритмі не вистачає розширення пам'яті вузла, що призводить до надмірності.

*Алгоритм RRT** є оптимізованим алгоритмом RRT. Завдяки якому асимптотично оптимальний шлях можна отримати завдяки отриманню достатньої кількості точок вибірки під час ітерації оновлення. Алгоритм додає дві операції: *повторний вибір батьківського вузла* та *повторне підключення вузла*.

Повторний вибір батьківського вузла: з p_{new} як центром і потенційними батьківськими вузлами в області з радіусом R, обчислюється вартість шляху цих вузлів. Вибираємо вузол з найменшою вартістю шляху як новий батьківський вузол і проводимо поєднання. Якщо на шляху є колізія, вибираємо інший альтернативний батьківський вузол.

Повторне підключення вузла: використовуючи p_{new} як центр, вибираємо діапазон із радіусом R і пробуємо змінити батьківський вузол вузла в межах діапазону на p_{new}. Якщо це може зменшити загальну вартість шляху, від'єднаємо вузол від його батьківського вузла та підключаємо до p_{new}. Якщо в з'єднанні виникне колізія, з'єднання буде розірвано, а інші вузли в межах діапазону продовжуватимуть вибиратися та послідовно перевірятися.

Якщо випадкова точка вибірки p_n береться на оптимальному шляху σ^* від p_{start} до p_{end} у просторі, вартість шляху буде зменшена, оскільки p_n стає батьківським вузлом свого сусіднього вузла. Після достатньої вибірки завжди отримаємо (4):

$$\begin{cases} L\{p_1, p_2, \dots, p_n\} = L^*, \\ \forall p \in L, p_{rand}(x) = p, x \in (0, iter). \end{cases} \quad (4)$$

де $iter$ – кількість ітерацій. Асимптотична оптимальність алгоритму доведена. Оскільки алгоритм розширює масштаб пошуку, операції повторного вибору батьківського вузла та повторного підключення вузла перетинають усі точки в дереві, що призводить до величезного навантаження на пам'ять.

*Алгоритм RRT*FN* є модифікацією алгоритму задля підвищення ефективності пошуку шляхом фіксації кількості вузлів.

Алгоритм RRT*FN (фіксовані вузли RRT*) вводить концепцію максимальної кількості вузлів на основі алгоритму RRT*. Цей алгоритм встановлює максимальну кількість вузлів, дозволену в дереві. Коли кількість вузлів у просторі станів більше попередньо встановленої кількості вузлів ,довільно видаляється бездітний листовий вузол, за винятком кінцевого вузла.

Алгоритм RRT*FN містить такі кроки:

Крок 1: Те саме, що крок 1–3 алгоритму RRT.

Крок 2: Виконується повторний вибір батьківського вузла та повторне підключення вузла.

Крок 3: Кожного разу, коли генерується p_{new}, перевіряємо кількість вузлів у просторі. Якщо кількість вузлів перевищує максимальну $Nodes_{FIX}$, довільно видаляються бездітні листові вузли останнього вузла, що не є шляхом ($p_{delete} \in p_{leaf} exept\ p_{new}$).

Крок 4: Коли дерево розростається до рівня p_{end} або лінія, що з'єднує p_{new} і p_{end}, не перетинається з перешкодою, генерується можливий шлях σ.

Крок 5: Повторяємо кроки 1–3 задля виконання асимптотичної оптимізації рішення шляху.

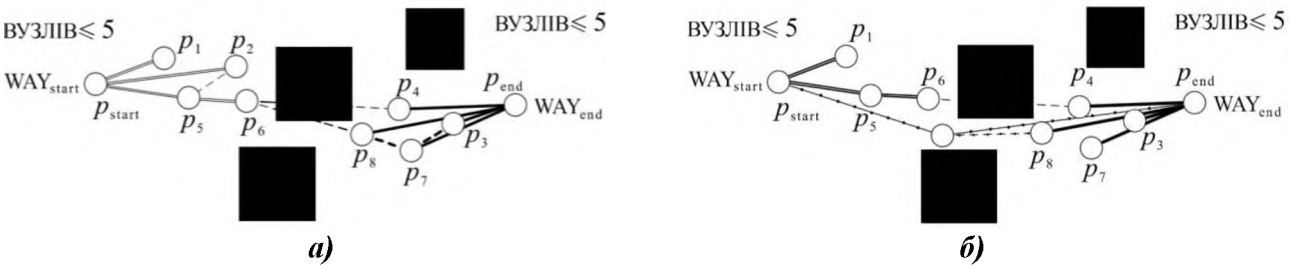
Пропонується удосконалений алгоритм, що поєднує стратегію двонаправленого жадібного пошуку з алгоритмом RRT*FN для вирішення проблеми сліпоти в односторонньому зростанні дерева. Алгоритм RRT*FN у швидкості пошуку не має переваги над алгоритмом RRT* та може лише зменшити надлишкову вибірку точок, яких слід уникати. Надлишкове зростання має ефект обмеження розміру дерева для покращення швидкості роботи програми, коли кількість ітерацій є великою та великий розмір дерева. Після додавання стратегії двостороннього жадібного пошуку, шлях запланований алгоритмом, має більш очевидну спрямованість, а початковий шлях можна отримати відносно швидко.

Жадібний двосторонній пошук вимагає встановлення двох випадкових дерев пошуку, дерева 1 і дерева 2, у початковій і цільовій точках одночасно. Два дерева ростуть назустріч одне одному відповідно, і під час зростання використовується жадібна стратегія.

1) Встановлено, що початкова точка p_{start} і p_{end} двох дерев є кінцевими точками один одного.

2) Після того, як Дерево 1 генерує новий вузол p_{new}, він продовжує рости від p_{new} до вузла, найближчого до себе від Дерева 2 (щоб гарантувати мінімізацію вартості шляху), доки не зіткнеться з іншим або не досягне цілі.

3) Якщо після зростання кількість послідовних вузлів Дерева 1 перевищує кількість вузлів іншого дерева, тоді задача зростання переходить до Дерева 2.



а) *б)*

Рис. 1. Схематична діаграма двостороннього жадібного пошуку:
а – процес зростання; *б* – остаточний шлях

Використання стратегії двостороннього жадібного пошуку з алгоритмом RRT*FN для покращення одностороннього випадкового зростання до цільового вузла дерева є

виправленням недоліка низької продуктивності. Загальний принцип роботи покращеного алгоритму показаний на рисунку 1.

Із початкової точки p_{start} і цільової точки p_{end} будуємо два дерева і жадібно розвиваємо одне до одного. Подвійна лінія є деревом початкової точки, а товста лінія – це кінцеве дерево. Чорні квадрати – перешкоди на шляху. Вузли в просторі генеруються в порядку з нижніми індексами. Якщо вказано кількість вузлів у кожному дереві не більше 5, коли $\mathrm{WAY}_{\mathrm{start}} \cap \mathrm{WAY}_{\mathrm{end}} \neq 0$, означає, що два дерева з'єдналися.

Як видно з рисунка 1, алгоритм BD-RRT*FT, який додає двонаправлену стратегію жадібного пошуку, має більш очевидну спрямованість, ніж алгоритм RRT*FN, рішення шляху також має тенденцію бути оптимальним зі збільшенням процесу ітерації.

Складність фактичного робочого середовища БАНЗ полягає в тому, що початковий запланований шлях може бути пошкоджено під час роботи БАНЗ. Наприклад, під час бойових дій перешкода була зсунута, або з одиничного блока перешкода під впливом вибухової хвилі змінила свій розмір чи перетворилася на групу перешкод в просторі на карті, які не були заздалегідь внесені до списку перешкод.

Одночасно, коли алгоритм динамічного програмування оновлює інформацію про навколишнє середовище, точки відбору проб поступово збільшуватимуться з часом. Чим більший розмір дерева, тим більше навантаження на динамічний алгоритм при оновленні в реальному часі, що сповільнить швидкість роботи

Щоб вирішити вищевказані проблеми алгоритму RRT*FN, впровадимо модифікації алгоритму застосувавши принцип динамічного оновлення та відновлення шляху. А саме під час кожної ітерації алгоритм оновлюватиме інформацію про середовище та виконуватиме оновлення планування на основі вихідних вузлів. Крім того, також проведемо оновлення інформації про розташування БАНЗ. Щоб зменшити розмір дерева під час процесу планування, зазначимо як непотрібні вузли і допоміжні гілки, через які проходить БАНЗ, використовуючи поточне розташування БАНЗ як новий кореневий вузол. Якщо запланований шлях знищено перешкодами, алгоритм відкине вузли на дереві, покриті перешкодами, і використає простір для перебудови дерева з повними вузлами і ефективного відновлення шляху, як показано на рисунку 2.

Етапи відновлення (перебудови) шляху є такими.

Крок 1: Коли шлях зруйновано перешкодою, від'єднаймо зруйнований вузол від інших вузлів, запишіть номер вузла та відкиньте ці вузли.

Крок 2: Виберіть відкинутий вузол, найдальший від кореневого вузла, з його дочірнім вузлом як центром кола, і повторно знайдіть потенційні батьківські вузли для нього в межах області з радіусом R. Потенційний батьківський вузол із найменшою вартістю шляху є новим батьківським вузлом. Виконайте повторне підключення вузла.

Крок 3: Продовжуйте використовувати інші дочірні вузли найдальшого округленого вузла та повторюйте крок 2, доки всі відкинуті вузли, крім кореневого вузла, не знайдуть свої батьківськи вузли. Відновлення шляху завершено.

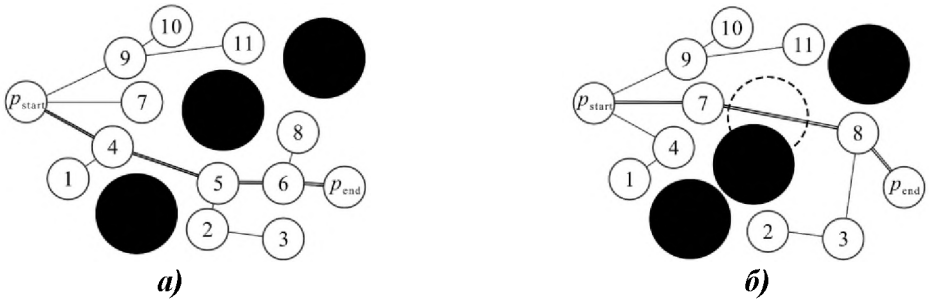


Рис. 2. Схематична діаграма відновлення шляху:
а – до переміщення перешкоди; *б* – після переміщення перешкоди

Аналіз повноти алгоритму та вибір фіксованої кількості вузлів

Перевіримо та доведемо імовірнісну повноту та асимптотичну оптимальність алгоритму RRT*.

Припустимо, що простір станів $X \subseteq R_n$, множина з 2 дерев пошуку $T_1, T_2 \subseteq X$ становить (5):

$$\begin{cases} \forall p \in \sigma, p_{rand}(x) = p, x \in (0, iter1), \\ \forall q \in \sigma, Q_{rand}(x) = q, x \in (0, iter2) \\ \sigma \subseteq T_1 \cup T_2, p_{rand} \subseteq T_1, Q_{rand} \subseteq T_2. \end{cases} \quad (5)$$

Тобто RRT* алгоритм двонаправленого жадібного пошуку все ще має ймовірнісну повноту. Після знаходження рішення шляху σ два дерева об'єднуються, а асимптотична оптимальність подвійного деревного алгоритму така сама, як і у однодеревного RRT* алгоритму. Модифікований алгоритм передбачає максимальну кількість вузлів, якщо кількість фіксованих вузлів $n \to \infty$, імовірнісна повнота та асимптотична оптимальність вдосконаленого алгоритму узгоджуються з алгоритмом RRT*. Якщо кількість фіксованих вузлів n більше числа вузлів у розв'язку шляху m, то алгоритм не має імовірнісної повноти та асимптотичної оптимальності. Розглянемо $m \leqslant n$, коли максимальна кількість вузлів $n = m$ і рішення шляху σ все ще не знайдено. Алгоритм буде відкидати випадкові бездітні листові вузли й продовжувати вибірку та планування, а повторний вибір батьківського вузла та повторне підключення вузла є еквівалентними розподілу випадкових точок p_{rand}, які залишаються у всьому просторі станів. Після достатньої кількості ітерацій маємо (6):

$$\forall p \in \sigma, p_{rand}(x) = p, x \in (0, iter). \quad (6)$$

Якщо шлях існує, кінцевий шлях L має бути перестановкою та комбінацією елементів у наборі σ, тобто повнота ймовірності задовольняється. Так само, через повторний вибір вузла та повторне підключення батьківського вузла, якщо $p \in \sigma$, $p_{rand}(x) = p$, і цю точку остаточно вибрати як об'єкт повторного з'єднання вузла, вартість шляху однозначно зменшиться. Коли кількість ітерацій достатньо велика, все ще буде вірно (7):

$$\begin{cases} L\{p_1, p_2, \dots, p_n\} = \sigma^*, \\ \forall p \in \sigma^*, p_{rand}(x) = p, x \in (0, iter). \end{cases} \quad (7)$$

Тобто покращений алгоритм має імовірнісну повноту та асимптотичну оптимальність. Хоча алгоритм має ймовірнісну повноту, у діапазоні $m \leqslant n < 1$, якщо кількість реперних вузлів встановлена надто малою, ймовірність отримання шляхового рішення майже дорівнює 0. Але якщо значення реперного вузла встановлено занадто великим, воно займатиме забагато місця в пам'яті. Зазвичай кількість реперних вузлів вибирається на основі емпіричних методів. Але для уникнення людського фактору, модифікований алгоритм автоматично перевіряє кілька груп за умови фіксованого простору, станів і розміру кроку.

Послідовність вибору кількості реперних вузлів:

- Отримаємо дані про середній час оптимального рішення.
- Виконаємо апроксимацію кривої 7 разів.
- Вибираємо кількість фіксованих вузлів у мінімальній точці підгонки інтервалу кривої як кількість фіксованих вузлів для експерименту.

Через випадковість точок вибірки, вибраних алгоритмом RRT, метод вибору фіксованої кількості вузлів може бути не оптимальним, але цей метод відбору є відносно розумним і дозволяє уникнути впливу неправильного вибору фіксованої кількості вузлів для експерименту.

*Реалізація алгоритму BD-RRT*FT* полягає в наступному:

Крок 1: Ініціалізуємо карту, дерева пошуку (Дерево 1 та Дерево 2). Додаємо p_{start} до списку вузлів Дерево 1 і p_{end} до списку вузлів Дерево 2.

Крок 2: Використовуємо менше дерево з двох дерев як ціль зростання (якщо кількість вузлів у дереві однакова, вибираємо Дерево 1), випадковим чином вибираємо точку відбору

проб p_{rand} у просторі. Вибираємо найближчий вузол $p_{nearest}$, і починаємо рухатися до p_{rand}, зростаючи з певним кроком, і отримуємо p_{new}.

Крок 3: Виконуємо повторний вибір батьківського вузла та повторне підключення вузла.

Крок 4: З p_{new} як з батьківського вузла, продовжуємо рости до вузла $p'_{nearest}$ дерева, доки він не досягне цієї точки або не зустріне перешкоду.

Крок 5: Після виконання процесу зростання оновлюємо карту. Встановлюємо вузол, де розташований БАНЗ як кореневий вузол. Видаляємо вузол та його гілки в останньому місці розташування БАНЗ, якщо $P \cap X_{obs} \neq 0, P \in \sigma$, видаліть вузли, що покриті перешкодами, а також вузли, лінії яких перетинаються з перешкодами. Від'єднайте ці точки від оточуючих вузлів і виконайте крок 3.

Крок 6: Порівняйте кількість вузлів у списках Дерево 1 і Дерево 2, виберіть менший з них як об'єкт зростання та виконайте кроки 2–5.

Крок 7: Якщо два дерева з'єдналися під час процесу зростання (відстань між двома вузлами в Дереві 1 і Дереві 2 менша за встановлене значення *connectDis*), визначте вузли на з'єднанні Дерева 1 і Дерева 2 як $p_{contree1}$ і $p_{contree2}$ та використовуйте $p_{contree1}$ як кореневий вузол. Обміняйтеся зв'язками «батько-нащадок» вузлів у Дереві 2 і використовуйте $p_{contree1}$ як батьківський вузол $p_{contree2}$ для об'єднання двох дерев у дерево злиття.

Крок 8: використовуючи p_{start} як початкову точку та p_{end} як цільову точку, виберіть найкоротший шлях у дереві злиття як рішення шляху.

Крок 9: виконайте асимптотичну оптимізацію шляху відповідно до кроків 1–3 алгоритму RRT*FN. Після завершення кожного зростання/повторного підключення виконайте крок 5.

2. Експериментальна перевірка алгоритму.

Імітаційне моделювання було здійсненням завдяки розробленій програмі моделювання, з використанням мови програмування Python 3+ та бібліотек PyGame, NumPy. В якості системи обчислювання використовувалась система 2-х сокетона Workstation із системними параметрами Intel® Xeon® Processor E5-2689 v4 (25M Cache, 3.10 GHz) = 2/ 4*64 Gb DDR4 2133/ SSD 250Gb/ VA Asus e GForce GTX 1650 GDDR6 4096Mb.

Для перевірити ефективності запропонованого алгоритму було проведене порівняння алгоритму BD-RRT*FT з іншими алгоритмами за трьома картами з оцінюванням показників продуктивності. Щоб полегшити аналіз симуляції та врахувати раціональність, перешкоди на мапах, що використані як блоки піксельної графіки, БАНЗ розглядається як точка піксельного розміру 4x4, а інші нерелевантні змінні, окрім алгоритму, контролюються, щоб бути узгодженими. Оскільки алгоритм RRT базується на випадковій вибірці, у процесі моделювання є непередбачуваність, і кожен результат вимірювання може мати відмінності, щоб усунути вплив випадковості, під час експерименту проводилось 100 незалежних експериментів для кожної ситуації та формувалися результати для порівняльного аналізу.

Перевірка роботи алгоритмів на карті 1 (карта зі статичними перешкодами).

Використовуємо карту 1, розміром 800 px × 600 px, як звичайну карту перешкод (рис. 3, *а*), яка використовується для перевірки швидкості реакції та індексу шляху алгоритму в нормальному середовищі перешкод. На цій карті верхній лівий кут встановлюється як початок координат [0, 0], а карта розташована в 4-му квадранті, початкова та кінцева позиція БАНЗ [20, 580], [780, 20] позначена значками БАНЗ та надписами START, END відповідно.

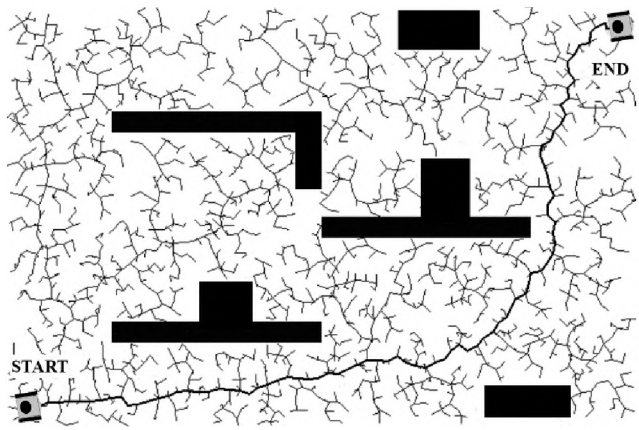
Перевіряються алгоритми RRT, B-RRT*, RRT*FN та BD-RRT*FT. Результати планування шляху показані на рис. 3, *а*. Товста лінія позначає отримане рішення шляху. Інші дерева є ітераційними обчисленнями.

Бачимо, що односторонній алгоритм має занадто багато надлишкових точок вибірки в процесі планування, тоді як двосторонній алгоритм має більш очевидну спрямованість. Для інтуїтивного порівняння середня довжина шляху рішення, отримана зі 100 експериментів. Кількість ітерацій наведено в таблиці 1.

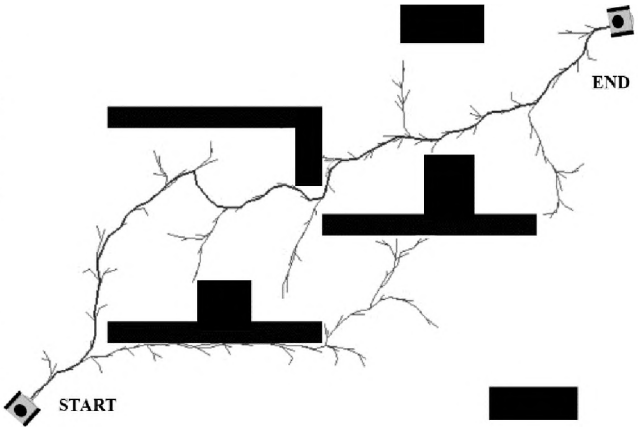
Таблиця 1

Дані порівняння продуктивності алгоритму на карті 1

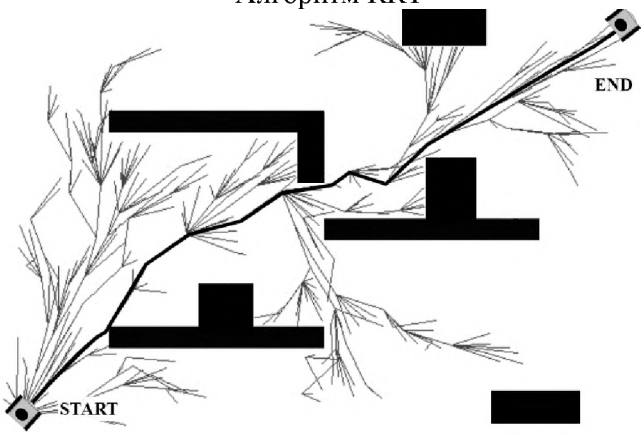
Алгоритм	RRT	B-RRT*	RRT*FN	BD-RRT*FT
Середня довжина шляху (у. о.)	820,662	726,307	754,156	708,778
Середній час роботи, x10^3 (с)	41,596	0,624	2,188	0,1673
Споживана пам'ять (Мб)	329,78	285,52	282,63	246,51



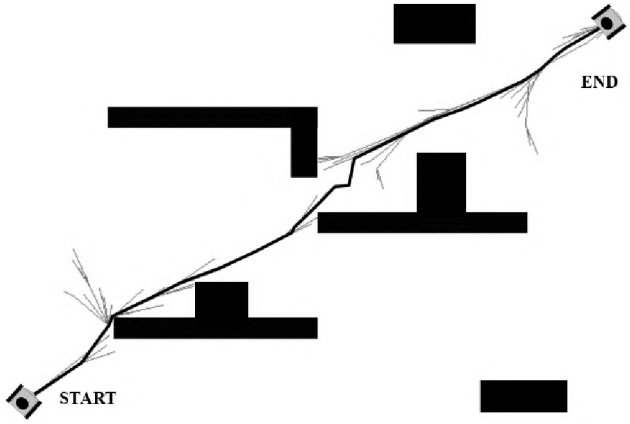
Алгоритм RRT



Алгоритм B-RRT

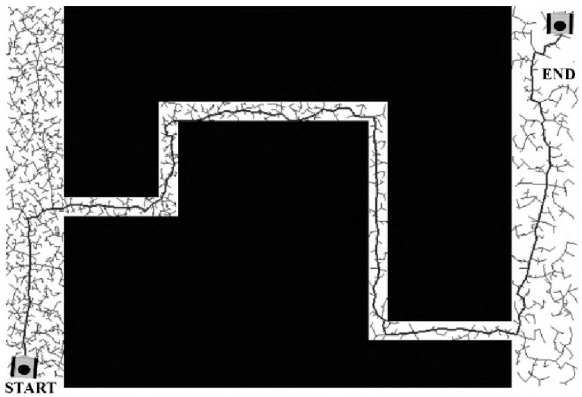


Алгоритм RRT*FN

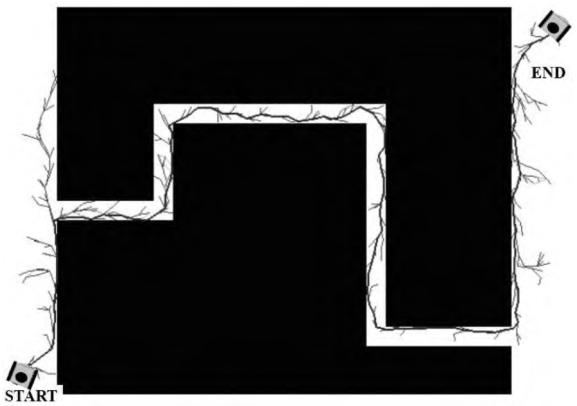


Алгоритм BD-RRT*FT

а)



Алгоритм RRT



Алгоритм B-RRT

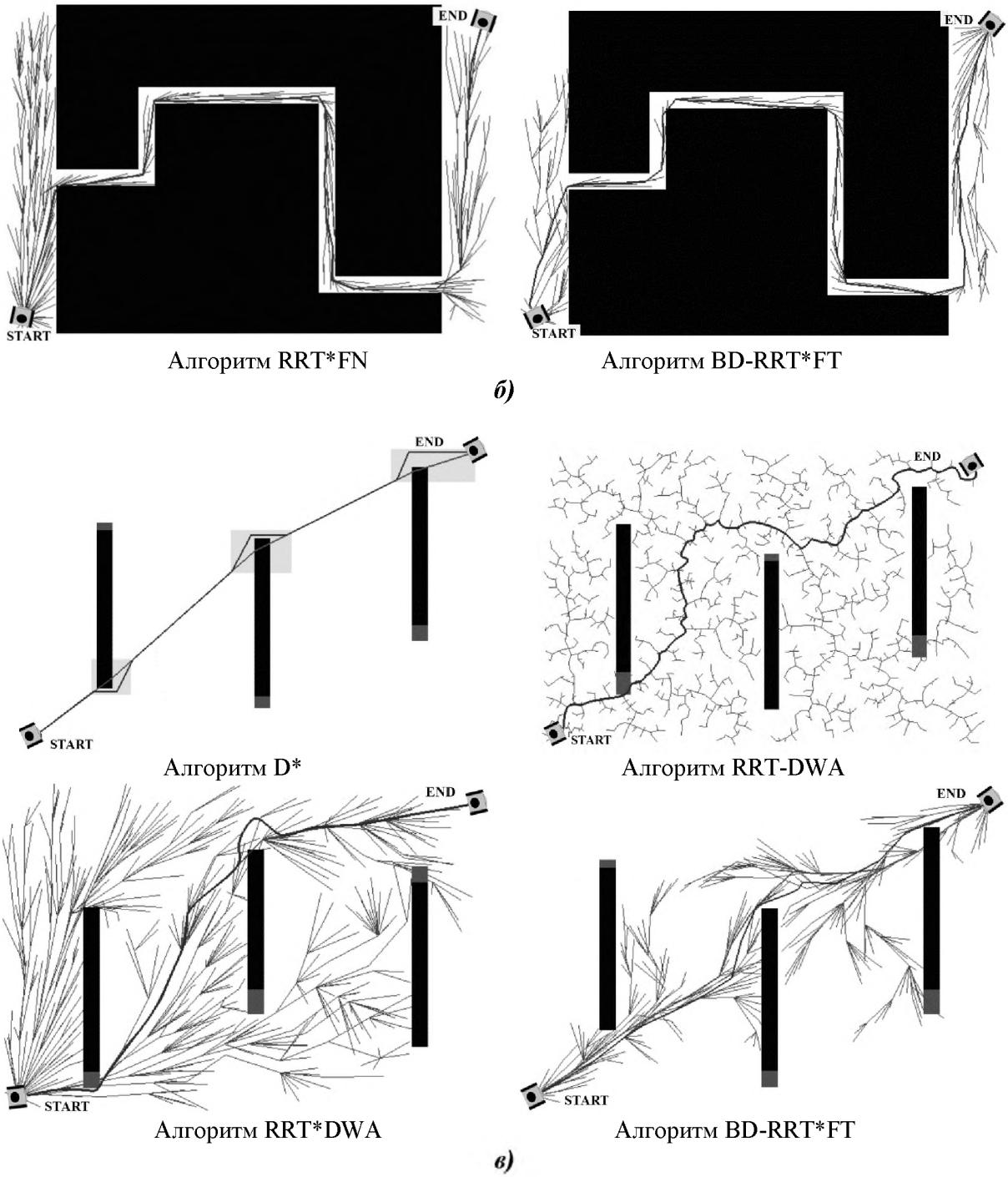
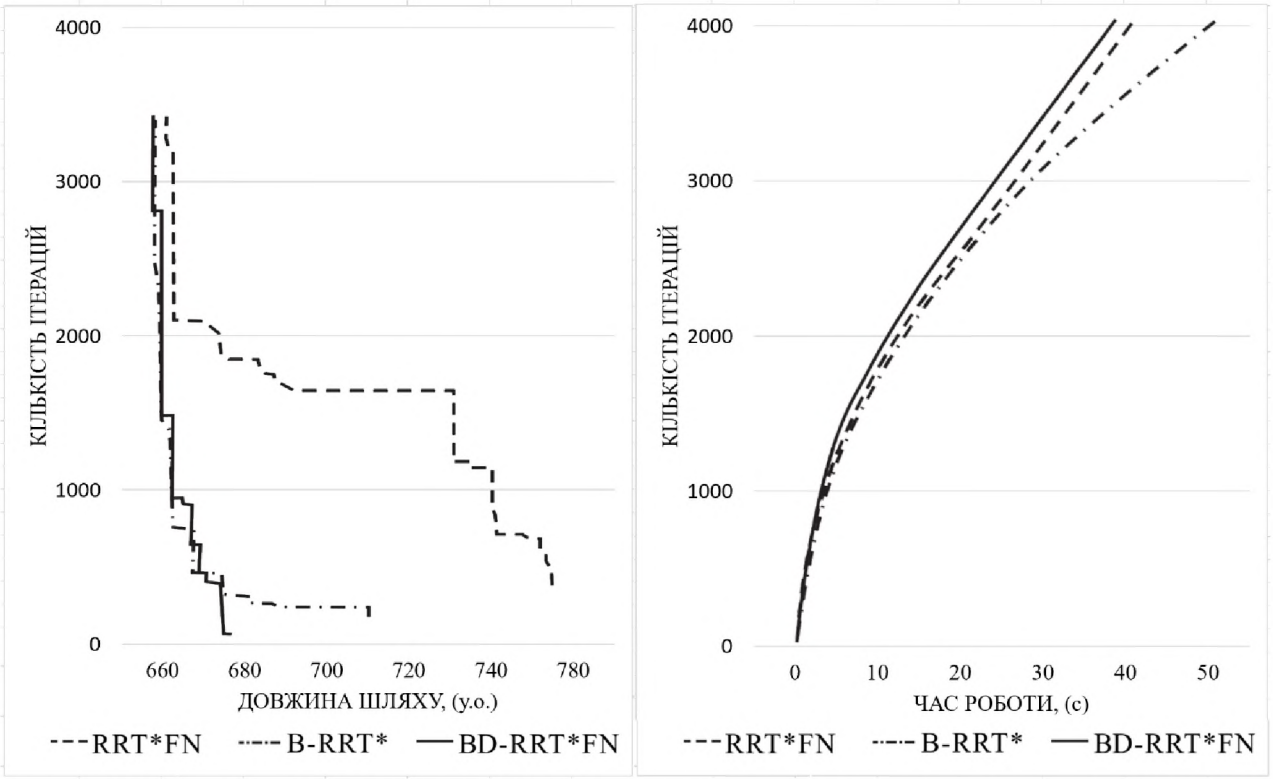


Рис. 3. Візуалізація результату роботи алгоритмів. Планування шляху за трьома типами карт:
а – карта 1: візуалізація результату роботи алгоритмів алгоритму;
б – карта 2: візуалізація результату роботи алгоритмів; *в* – карта 3: візуалізація результату алгоритмів

Як видно з таблиці 1, двонаправлений алгоритм має певні переваги у швидкому пошуку рішення. Враховуючи, що вдосконалений алгоритм використовує стратегію двонаправленого жадібного пошуку, загальна вартість шляху ще більше зменшується. Оскільки алгоритм B-RRT*, алгоритм RRT*FN і алгоритм BD-RRT*FT. Усі належать до алгоритму оптимального рішення. На рисунку 4, *а, б* показано графічний порівняльний аналіз продуктивності кожного алгоритму на карті 1.



а) *б)*

Рис. 4. Порівняльний аналіз продуктивності кожного алгоритму на карті 1:
а – зв'язок між кількістю ітерацій та довжиною шляху;
б – зв'язок між кількістю ітерацій та часом виконання

На рисунках 4, *а* і 4, *б* видно, що довжина рішення шляху та час пошуку алгоритму BD-RRT*FN кращі, ніж у двох інших алгоритмів. Довжина алгоритму B-RRT* після 500 ітерацій така сама, що й для BD-RRT*FN. Розуміємо, що основа алгоритму в основному та сама, але оскільки він не має фіксованої кількості вузлів, час виконання демонструє увігнуту тенденцію до зростання зі збільшенням кількості ітерацій (рис. 4, *б*). Після 1500 ітерацій час роботи вдосконаленого алгоритму становить приблизно лінійну залежність, що пов'язано з кількістю ітерацій. При 3000 ітераціях алгоритму B-RRT* необхідно 36,117 с, а BD-RRT*FN – 29,221 с для виконання завдання зі знаходження шляху.

Перевірка роботи алгоритмів на карті 2 (карта з вузьким проходом).

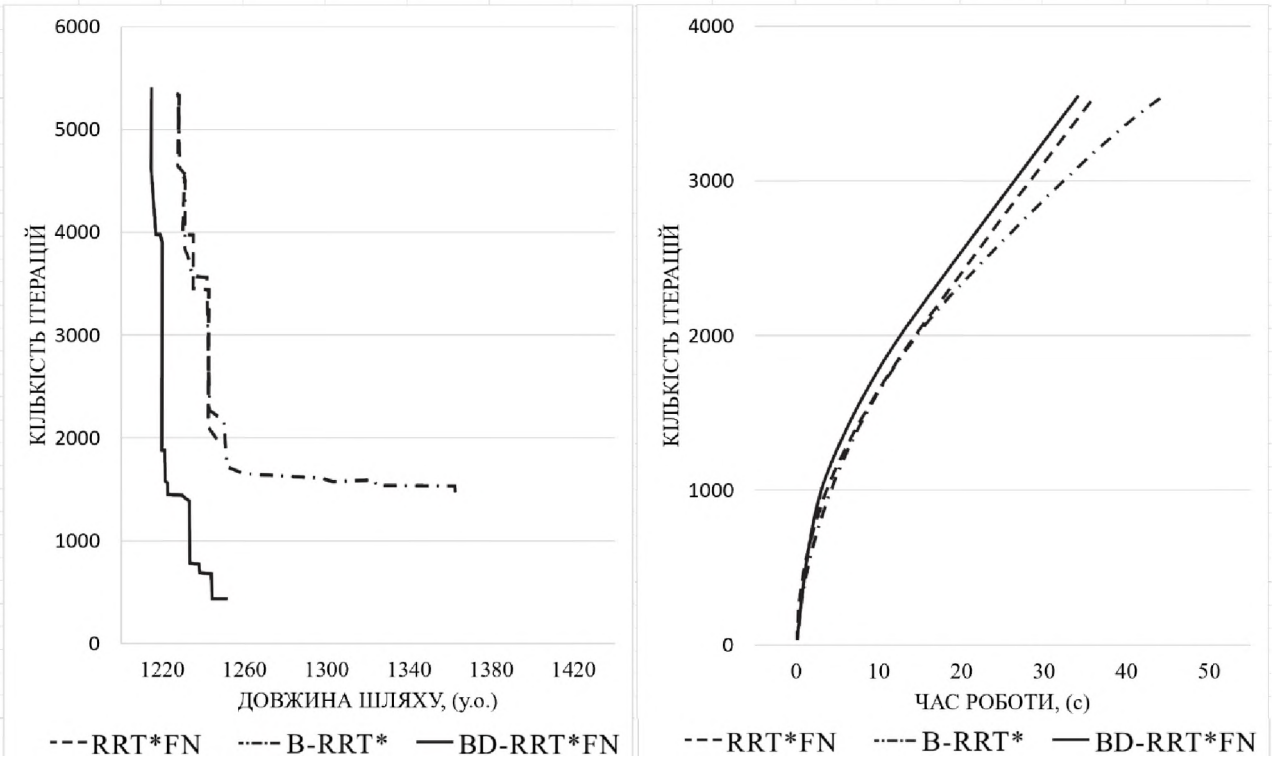
Використовуємо карту 2 розміром 800 px × 600 px як карту з вузькими щілинами (рис. 3, *б*). За винятком конфігурації розташування перешкод на карті, решта налаштувань такі самі, як для карти 1. Результати планування шляху чотирьох алгоритмів показано на рисунку 3, *б*. Останні три є першими отриманими шляхами.

Для карт з вузьким каналом простору, через сліпоту зростання, односторонньому алгоритму важко отримати точки вибірки у вільному просторі в межах вузького каналу, в результаті чого кількість ітерацій і час виконання набагато вищі, ніж у двостороннього алгоритму пошуку шляху, а точки вибірки здебільшого зосереджені в лівому вільному просторі. В таблиці 2 показано отримані під час експерименту середня довжина рішення шляху та час виконання.

Таблиця 2

Порівняння показників ефективності алгоритму на карті 2

Алгоритм	RRT	B-RRT*	RRT*FN	BD-RRT*FT
Середня довжина шляху (у. о.)	1383,031	1355,370	1239,629	1239,042
Середній час роботи, x10^3 (с)	145,188	2,4825	4,919	1,415
Споживана пам'ять (Мб)	320,98	281,12	280,67	245,72



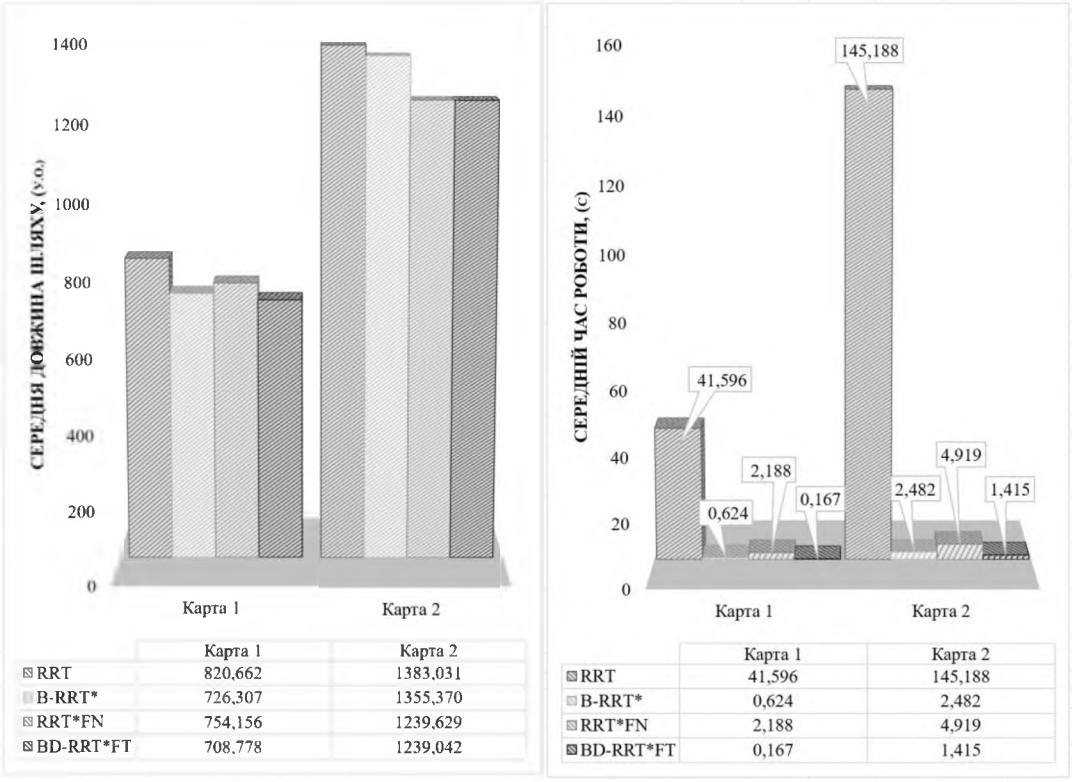
а) *б)*

Рис. 5. Порівняльний аналіз продуктивності кожного алгоритму на карті 2:
а – зв'язок між кількістю ітерацій та довжиною шляху;
б – зв'язок між кількістю ітерацій та часом виконання

Оскільки ймовірність вибору точок вибірки у вузькому каналі зменшується, загальна кількість точок вибірки збільшується. Час роботи алгоритму RRT перевищує 145 с, а кількість ітерацій перевищує 20 000, що створює велике навантаження на пам'ять, що досить сильно уповільнює процес розрахунку.

Порівняння продуктивності алгоритмів оптимального рішення та результати ітераційного процесу з додатковою оптимізацією в 3 алгоритмах показано на рисунку 5, *а*, *б*. З нього видно, що довжина та швидкість ітерації шляху, отриманого алгоритмом BD-RRT*FN, кращі, ніж у алгоритмів B-RRT* і RRT*FN.

При перевірці з використанням карт 1 і 2 зі статично встановленими перешкодами, модифікований алгоритм BD-RRT*FN показав високу продуктивність порівняно з іншими алгоритмами (рис. 6), тому для подальшої перевірки роботи алгоритму з динамічними об'єктами було прийнято рішення на заміну набору алгоритмів для карти 3.



а) *б)*

Рис. 6. Порівняльні діаграми продуктивності алгоритмів на карті 1 і карті 2:
а – зв'язок між кількістю ітерацій та довжиною шляху;
б – зв'язок між кількістю ітерацій та часом виконання

Перевірка роботи алгоритмів на карті 3 (карта з динамічними об'єктами).

Для подальшого вивчення продуктивності алгоритму в динамічному середовищі, створено динамічну карту розміром 600 px × 800 px, що містить рухомі перешкоди. Динамічні перешкоди (чорна частина, розмір 40×200) рухаються вперед і назад уздовж осі Y, а діапазон руху становить $Y = 20 – 580$ px. В якості порівняльних алгоритмів представлено алгоритм D^* (Dynamics A^*), алгоритм RRT-DWA та алгоритм RRT*DWA. Оскільки в динамічному середовищі довжина рішення шляху змінюється нерегулярно, то за для спрощення аналізу приймемо, що лише довжина шляху переміщення динамічної перешкоди буде впливати на глобальний шлях БАНЗ. Середні час виконання побудови шляху та довжина надані в таблиці 3, а результати планування шляху алгоритму показані на рисунку 3, *в* та рисунку 7.

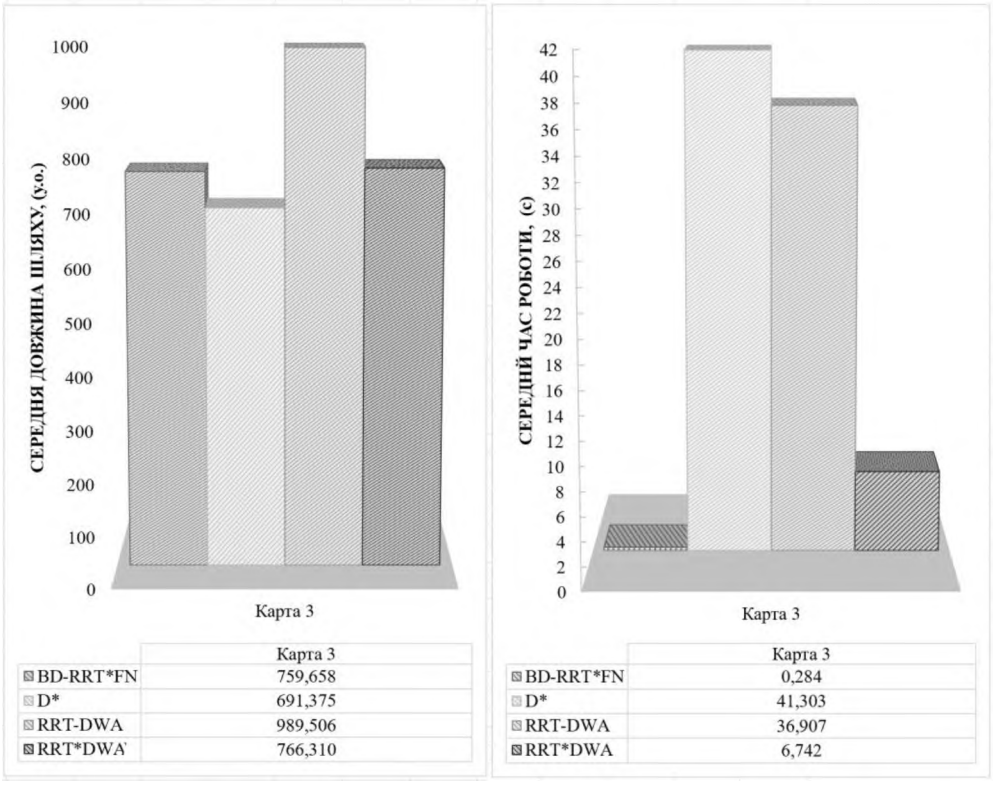
Таблиця 3

Порівняння показників ефективності алгоритму на карті 3

Алгоритм	BD-RRT*FT	D^*	RRT-DWA	RRT*DWA
Середня довжина шляху (у. о.)	759,658	691,375	989,506	766,310
Середній час роботи (с)	0,2844	41,303	36,9078	6,7424
Споживана пам'ять (Мб)	245,72	280,02	284,17	301,42

Як видно з рисунка 3, *в*, усі алгоритми мають можливості планування в реальному часі та можуть уникати динамічних (рухомих) перешкод. Середній шлях, запланований алгоритмом D*, коротший, але оскільки алгоритм D* є евристичним алгоритмом пошуку і базується на графах, його швидкість планування низька, а середній час планування на карті 3

перевищує 40 с. Алгоритми RRT-DWA і RRT*DWA спочатку використовують алгоритми RRT і RRT* для глобального планування шляху, а потім використовують алгоритм DWA під час руху БАНЗ. Локальне планування шляху використовується для досягнення динамічного уникнення перешкод, але через це легко впасти в локальну оптимальність, а середня довжина шляху та глобальний апріорний час планування шляху довший, ніж запропонований модифікований алгоритм BD-RRT*FT. Фактично експеримент показав, що модифікований алгоритм BD-RRT*FT має кращі можливості динамічного планування, але запланований шлях може бути тимчасово поганим через те, що деякі вузли вибірки відкидаються після того, як шлях знищено перешкодами. Ця ситуація покращується після кількох ітерацій, щоб переконатися, що шлях є цілісним.



а) *б)*

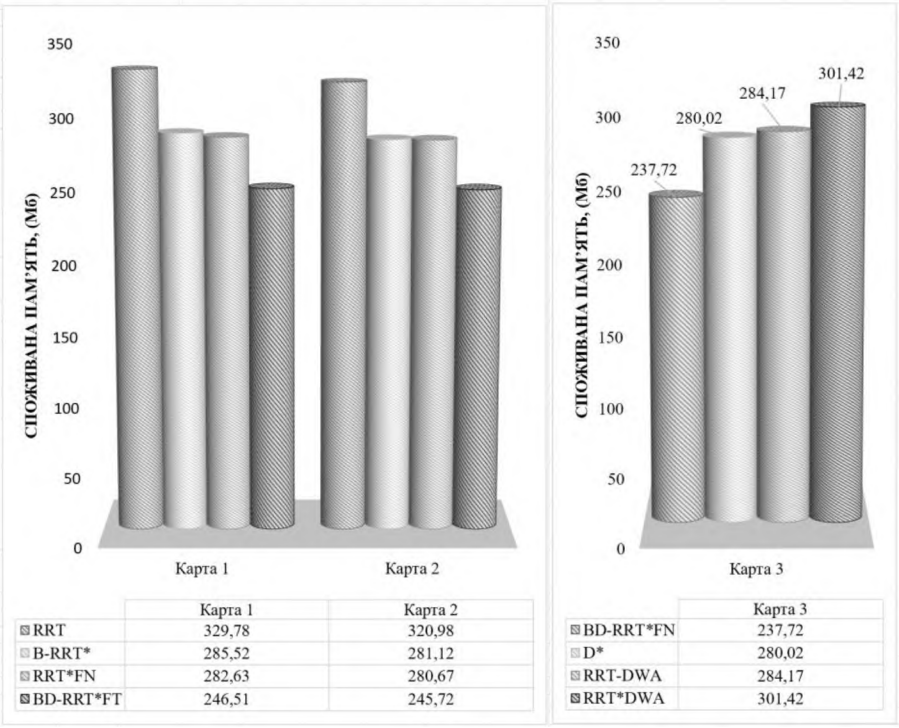
Рис. 7. Порівняльна діаграма продуктивності алгоритмів з динамічними об'єктами на карті 3:
а – зв'язок між кількістю ітерацій та довжиною шляху;
б – зв'язок між кількістю ітерацій та часом виконання

Під час проведення експериментальної частини на всіх етапах перевірки проводився контроль навантаження на пам'ять системи прийняття рішення.

Середні результати споживання при роботі всіх алгоритмів вказано в таблицях 1–3.

Також на рисунку 8 представлено порівняльні діаграми споживання об'єму пам'яті алгоритмами. Ми бачимо пряму залежність від складності та оптимальності виконання алгоритму та шляхів пошуку рішень.

Як і передбачалось, запропонований модифікований алгоритм BD-RRT*FT показав гарну ефективність щодо завантаження пам'яті, внаслідок чого є можливість використання запропонованого алгоритму з менш потужними системами прийняття рішень.



	Карта 1	Карта 2
RRT	329,78	320,98
B-RRT*	285,52	281,12
RRT*FN	282,63	280,67
BD-RRT*FT	246,51	245,72

	Карта 3
BD-RRT*FN	237,72
D*	280,02
RRT-DWA	284,17
RRT*DWA	301,42

а) *б)*

Рис. 8. Порівняльна діаграма ефективності завантаження пам'яті системи прийняття рішень при виконанні пошуку шляху алгоритмами:
а – робота алгоритмів на картах 1 і 2; *б* – робота алгоритмів на картах 3

Висновки

З огляду на існуючу проблему планування шляху роботів у режимі реального часу, яка є центром досліджень у галузі інтелектуальних мобільних роботів, та розуміючи складність завдань під час проведення бойових дій в урбанізованому просторі щільної забудови міста з постійною зміною ландшафту, які покладаються на БАНЗ та мають тенденцію до постійного ускладнення, було запропоновано метод планування шляху БАНЗ з використанням модифікації динамічного двонаправленого RRT-алгоритму. Для чого проведено аналіз існуючих алгоритмів вирішення задачі пошуку шляху зі статичними перешкодами. На основі отриманої аналітики проведено модифікацію асимптотично оптимального алгоритму RRT*FT та розроблено алгоритм BD-RRT*FT, який можна застосовувати в динамічних середовищах, завдяки використанню двонаправленого жадібного пошуку для покращення часу планування алгоритму, довжини шляху та ефективності використання системних ресурсів пристрою прийняття рішення.

Під час досліджень з'ясовано, що для алгоритму RRT і його модифікацій існують різноманітні рішення, які побудовані з використанням сліпої вибірки, що ускладнює навігацію у вузьких ділянках. При цьому запропонований алгоритм завдяки оптимізації таких проблем не має і показав гарні результати.

Експериментально доведено ефективність алгоритму BD-RRT*FT задля задоволення вимог планування шляху в реальному часі, дозволяючи БАНЗ швидко отримати оптимальний шлях без зіткнень у динамічних середовищах у режимі реального часу.

Напрями подальших досліджень

Базуючись на виконаній роботі та отриманих в результаті даних, в подальшому планується створити для БАНЗ реальний прототип системи прийняття рішень, побудований з використанням запропонованого модифікованого алгоритму BD-RRT*FT для подальшого вивчення поведінки алгоритму в реальних умовах в середовищі з динамічними перешкодами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Бернацький А. П., Панченко І. В., Восколович О. І. Розширена математична модель руху автономного наземного робота розвідника в умовах бойових дій в урбанізованому просторі // Озброєння та військова техніка. 2021. № 30 (2). С. 121–129.

2. Winnfield J. A. Unmanned Systems Integrated Roadmap FY2011-2036 / Winnfield J. A. Jr., Kendall F.-Washington, DC: U. S. Department of Defense, March 9, 2012.

3. Бернацький А., Панченко І., Восколович О. Основи робототехніки військового призначення: конспект лекцій. Київ: ВІТІ, 2021.

4. Thomas H. Corman, Charles I. Leiserson, Ronald L. Rivest, Clifford Stein. Algorithms: construction and analysis. Introduction to Algorithms. Michigan U.: Williams, 2006. ISBN 978-0-262-53305-8.

5. Robert Schapire, Yoav Freund. Boosting: Foundations and Algorithms. MIT, 2012. ISBN 10: 0262526034; ISBN 13: 9780262526036.

6. Maw A. A., Tyan M., Nguyen T. A. et al. iADA*-RL: Anytime graph-based path planning with deep reinforcement learning for an autonomous UAV // Applied Sciences. 2021. № 11 (9). P. 1–18. DOI: https://doi.org/10.3390/app11093948.

7. Kadry S., Alferov G., Fedorov V. D-Star Algorithm Modification // International Journal of Online and Biomedical Engineering (iJOE). 2020. Vol. 16. No. 8. P. 108–113. DOI: https://doi.org/10.3991/ijoe.v16i08.14243.

8. Majeed A., Hwang SO. Path planning method for UAVs based on constrained polygonal space and an extremely sparse waypoint graph // Applied Sciences. 2021. № 11 (12). P. 5340. DOI: https://doi.org/10.3390/app11125340.

9. Kagan E. and Ben-Gal I. A Group-Testing Algorithm with Online Informational Learning. IIE Transactions // Institute of Industrial Engineers. № 46 (2). P. 164–184. DOI: https://doi.org/10.1080/0740817X.2013.803639.

10. Jeong I. B., Lee S. J., Kim J. H. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate // Expert Syst. Appl. 2019. № 123. P. 82–90. DOI: https://doi: 10.1016/j.eswa.2019.01.032.

11. Black, Paul E. Greedy algorithm. Dictionary of Algorithms and Data Structures, US National Institute of Standards and Technology* PE Black. URL: http://www.nist.gov/dads/HTML/greedyalgo.html.

12. Gammell J. D., Srinivasa S. S., Barfoot T. D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic; Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems; Chicago, IL, USA. 14–18 September 2014. DOI: https://doi.org/10.1109/IROS.2014.6942976.

13. Taheri E., Ferdowsi M. H., Danesh M. Fuzzy greedy RRT path planning algorithm in a complex configuration space. Int. J. Control. Autom. Syst. 2018. № 16. P. 3026–3035. DOI: https://doi:10.1007/s12555-018-0037-6.

14. Adiyatov, Olzhas; Varol, Huseyin Atakan. A novel RRT-based algorithm for motion planning in Dynamic environments. In Mechatronics and Automation (ICMA), 2017 IEEE International Conference on, 2017. P. 1416–1421. DOI: https://doi:10.1109/ICMA.2017.8016024.

15. Spanogianopoulos, Sotirios and Sirlantzis, Konstantinos Non-holonomic Path Planning of Car-like Robot using RRT*FN. In: 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). 2015. IEEE. P. 53–57. E-ISBN 978-1-4673-7971-7. DOI: https://doi:10.1109/URAI.2015.7358927.

16. Бідюк П. І., Тимощук О. Л. та ін. Системи і методи підтримки прийняття рішень. Київ: КПІ ім. Ігоря Сікорського, 2022.

17. Li B., Chen B. An Adaptive Rapidly-Exploring Random Tree, Journal of Automatica Sinica. 2021. IEEE. DOI: https://doi.org/10.1109/JAS.2021.1004252.

18. Klemm S., Oberländer J., Hermann A., Roennau A., Schamm T., Zollner J. M., Dillmann R. RT-Connect: Faster, asymptotically optimal motion planning / Conference: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO 2015). At: Zhuhai, ChinaVolume: 12. DOI: https://doi.org/10.1109/ROBIO.2015.7419012.