HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

BÀI TẬP LỚN TOÁN RỜI RẠC

Đề tài:
"Nghiên cứu thuật toán Batch Informed Trees (BIT*) trong lập kế hoạch chuyển động"

Giảng viên hướng dẫn: TS. Nguyễn Kiều Linh
Nhóm Sinh Viên Thực Hiện : Nhóm 8
Lớp: INT1359-20242-11
Niên khóa: 2024–2025
Hệ đào tạo: Đại học chính quy

Hà Nội, 04/2025

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

Điểm: **(Bằng chữ:**)

Hà Nội, ngày tháng năm 20...

Giảng viên phản biện

Mục lục

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn cô bộ môn Toán Rời Rạc đã tạo điều kiện cho nhóm được thực hiện báo cáo này. Chúng em cũng cảm ơn các bạn trong nhóm đã cùng nhau phối hợp, chia sẻ kiến thức và hoàn thành báo cáo về thuật toán BIT* một cách hiệu quả.

Mặc dù đã cố gắng hoàn thiện, nhưng do kiến thức còn hạn chế, báo cáo khó tránh khỏi thiếu sót. Chúng em rất mong nhận được sự góp ý của thầy cô để cải thiện trong những lần sau.

Hà Nội, tháng 04 năm 2025

Nhóm 8

Trưởng Nhóm: Kiều Tiến Đạt

Danh sách hình vẽ

Mở đầu

Trong lĩnh vực robot di động và robot thao tác, bài toán lập kế hoạch đường đi (path planning) đóng vai trò quan trọng để đảm bảo robot di chuyển an toàn và hiệu quả trong không gian làm việc. Với sự phát triển của các hệ thống robot có nhiều bậc tự do (DOF), bài toán này ngày càng trở nên phức tạp do không gian trạng thái có kích thước lớn.

Có hai nhóm phương pháp chính được áp dụng: thuật toán dựa trên đồ thị (graph-based) và thuật toán dựa trên mẫu ngẫu nhiên (sampling-based). Mỗi phương pháp đều có ưu điểm và hạn chế riêng, đặc biệt là khi áp dụng vào các không gian có số chiều cao.

Trong bối cảnh đó, thuật toán Batch Informed Trees (BIT*) ra đời như một sự kết hợp giữa tính có thứ tự của tìm kiếm đồ thị (A*) và khả năng bao phủ không gian của phương pháp sampling-based như RRT*. BIT* được đánh giá là phương pháp tối ưu tiệm cận, có hiệu quả cao trong các không gian phức tạp.

Mục tiêu của bài tập lớn này là tìm hiểu lý thuyết, cài đặt và đánh giá hiệu quả của thuật toán BIT* trong các bài toán lập kế hoạch chuyển động.

Trong báo cáo này chúng em sẽ tập trung trình bày một số nội dung chính như sau:

Chương 1: Giới thiệu tổng quan về các thuật toán lập kế hoạch đường đi và sự ra đời của BIT* Nội dung chương 1 sẽ khái quát các vấn đề và phương pháp nhận dạng giọng nói, khảo sát về các phương pháp học máy đang được sử dụng cho ba nhiệm vụ con, và trình bày về phạm vi của đồ án.

Chương 2: So sánh thuật toán BIT* với các thuật toán tìm đường khác Nội dung của chương 2 sẽ trình bày tổng quan về các thuật toán tìm đường truyền thống như A*, D*, RRT*, cùng với thuật toán BIT*. Chương này sẽ tập trung phân tích và so sánh hiệu quả của thuật toán BIT* với các phương pháp khác dựa trên các tiêu chí như độ dài đường đi, thời gian tính toán, khả năng hội tụ, và tính tối ưu. Thông qua việc đánh giá thực nghiệm trên các môi trường mô phỏng khác nhau, chương này làm rõ ưu điểm của BIT* trong việc kết hợp giữa khả năng tối ưu hóa theo mẫu của RRT* và tính hiệu quả trong quá trình mở rộng không gian tìm kiếm.

Chương 3: Cơ sở lý thuyết Nội dung của chương 3 trình bày các kiến thức nền

tảng liên quan đến thuật toán BIT*

Chương 4: Phân tích chi tiết thuật toán Batch Informed Trees (BIT*) Phân tích chi tiết thuật toán, các mô hình toán học. **Chương 5: Mã nguồn thuật toán** Toàn bộ mã nguồn thuật toán **Chương 6: Demo thuật toán và kiểm nghiệm** Chương này sẽ trình bày quy trình triển khai thực tế thuật toán BIT* trong một số môi trường mô phỏng cụ thể nhằm kiểm nghiệm hiệu quả hoạt động của thuật toán.

Chương 7: Kết luận Trong chương này, chúng em sẽ tổng kết lại các nội dung chính đã trình bày trong báo cáo, đồng thời rút ra những nhận xét và định hướng phát triển trong tương lai.

Chương 1

Giới thiệu tổng quan về các thuật toán lập kế hoạch đường đi và sự ra đời của BIT*

1.1 Bối cảnh và tầm quan trọng của lập kế hoạch đường đi

Lập kế hoạch đường đi (path planning) là một bài toán quan trọng trong lĩnh vực robot, đặc biệt khi robot cần tìm đường đi từ vị trí khởi đầu đến đích mà không va chạm với các vật cản. Bài toán trở nên ngày càng phức tạp khi làm việc trong các không gian có số chiều lớn, như trong các hệ thống robot thao tác có nhiều bậc tự do.

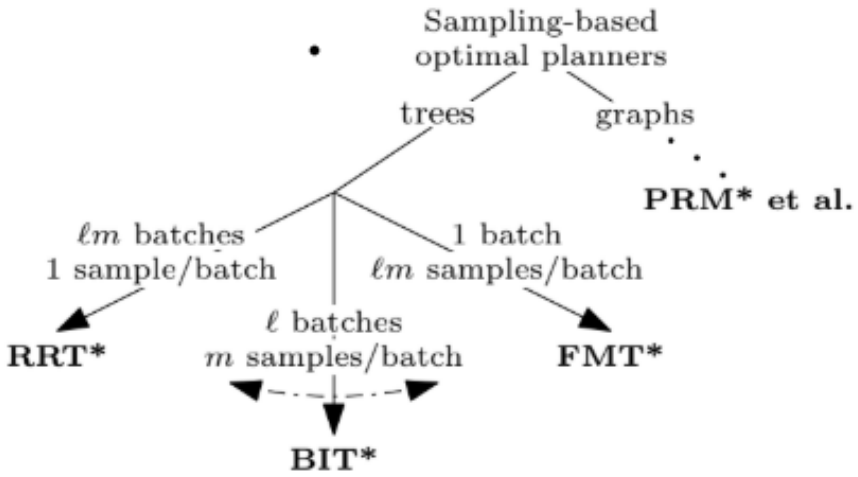
Hiện nay có hai hướng tiếp cận chính để giải quyết bài toán này: **thuật toán tìm kiếm trên đồ thị** và **thuật toán dựa trên lấy mẫu ngẫu nhiên**. Mỗi nhóm thuật toán đều có ưu và nhược điểm riêng, và đã có nhiều nghiên cứu nhằm kết hợp các ưu điểm để tạo ra thuật toán hiệu quả hơn.

1.2 Tổng quan về các thuật toán lập kế hoạch đường đi

Thuật toán dựa trên đồ thị (graph-search methods) như Dijkstra và A* sử dụng phương pháp quy hoạch động để tìm lời giải chính xác trên không gian rời rạc. Những thuật toán này đảm bảo tính hoàn thiện và tối ưu ở mức độ phân giải cố định. Đặc biệt, thuật toán A* được cải tiến để mở rộng hiệu quả hơn nhờ vào hàm heuristic, giúp tìm ra lời giải tối ưu một cách nhanh chóng. Tuy nhiên, trong không gian nhiều chiều, độ phức tạp của các thuật toán tìm kiếm trên đồ thị tăng theo cấp số mũ do vấn đề "lời nguyền chiều không gian" (curse of dimensionality), làm hạn chế khả năng áp dụng cho các bài toán lập kế hoạch trong các không gian lớn.

Thuật toán dựa trên mẫu ngẫu nhiên (sampling-based methods) như PRM (Probabilistic Roadmaps), RRT (Rapidly-exploring Random Trees) và các biến thể của

chúng như RRT*, Informed RRT*, FMT* hoạt động trực tiếp trên không gian liên tục bằng cách lấy mẫu ngẫu nhiên. Các thuật toán này giúp mở rộng tốt hơn trong các không gian có nhiều chiều, đặc biệt là khi không gian trạng thái là liên tục. Những thuật toán này có tính hoàn thiện xác suất (probabilistic completeness), nghĩa là có xác suất tìm được lời giải (nếu tồn tại), và một số biến thể như RRT* đảm bảo tính tối ưu tiệm cận. Tuy nhiên, việc tìm kiếm trong các thuật toán này không có thứ tự, và tốc độ hội tụ phụ thuộc nhiều vào cách thức lấy mẫu.



Hình 1: BIT* là sự kết hợp của RRT* và FMT* – hai hướng tiếp cận phổ biến trong lập kế hoạch dựa trên cây. Cách tiếp cận này giúp cải thiện hiệu suất trong không gian trạng thái phức tạp.

1.3 Sự ra đời của thuật toán BIT*

Thuật toán Batch Informed RRT* được phát triển bởi Jonathan D. Gammell, cùng với các cộng sự là:

- **Timothy D. Barfoot**
- **Siddhartha S. Srinivasa**

Jonathan D. Gammell

Timothy D. Barfoot

Siddhartha S. Srinivasa

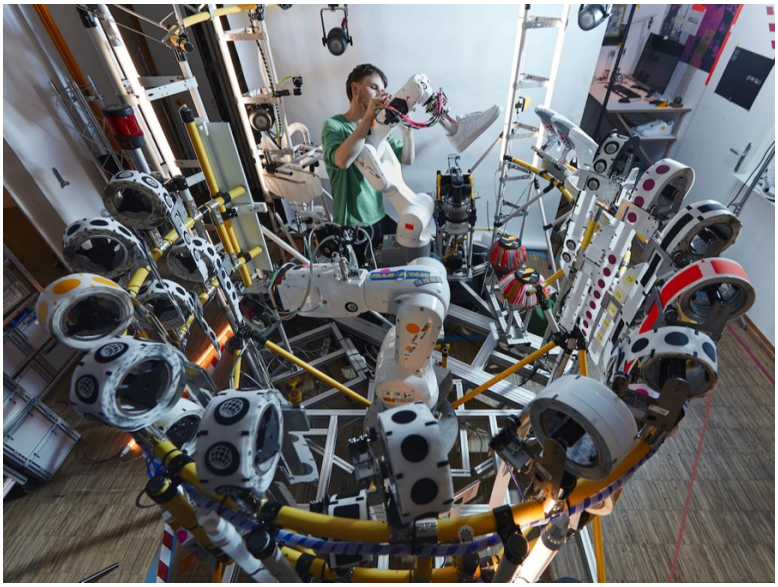
Trước những hạn chế của các thuật toán hiện tại, **thuật toán Batch Informed Trees (BIT*)** ra đời nhằm kết hợp thế mạnh của cả hai nhóm tiếp cận. BIT* sử dụng các lô mẫu (batches of samples) để tạo nên một đồ thị hình học ngầm, từ đó thực hiện tìm kiếm có thứ tự như A*, nhưng vẫn duy trì tính mở rộng bất cứ lúc nào như RRT*. Việc xử lý mẫu theo từng lô giúp thuật toán tập trung vào các vùng tiềm năng có thể chứa lời giải tốt hơn, đồng thời giữ được tính *"anytime"*, nghĩa là có thể cải thiện lời giải theo thời gian tính toán mà không yêu cầu phải hoàn tất quá trình tìm kiếm.

BIT* áp dụng các kỹ thuật tìm kiếm gia tăng như **Lifelong Planning A* (LPA*)** để tận dụng thông tin từ các lần tìm kiếm trước, giúp giảm thiểu công sức tính toán trong các lần tìm kiếm sau. Đồng thời, thuật toán này sử dụng **heuristic** để ưu tiên tìm kiếm các vùng có khả năng chứa lời giải tốt hơn. Với cách tiếp cận này, BIT* không chỉ hội tụ nhanh hơn đến lời giải tối ưu, mà còn có thể xử lý các bài toán có độ phức tạp cao như robot có nhiều bậc tự do.

Các kết quả thực nghiệm cho thấy BIT* có xác suất tìm được lời giải cao hơn và thời gian hội tụ nhanh hơn so với các thuật toán như RRT, RRT*, Informed RRT* và FMT*, đặc biệt trong các tình huống mà việc kiểm tra va chạm tốn kém về mặt tính toán. Điều này chứng tỏ hiệu quả vượt trội của BIT* trong việc giải quyết bài toán lập kế hoạch đường đi cho robot, đặc biệt là trong các không gian trạng thái có độ phức tạp cao và các bài toán cần tính toán kiểm tra va chạm tốn kém.

1.4 Một số ứng dụng của thuật toán BIT* trong thực tế

- **Tối ưu hóa trong Học máy:** Thuật toán BIT* có thể được sử dụng để tối ưu hóa quá trình học trong các mô hình học máy, đặc biệt là trong các bài toán có không gian trạng thái rất lớn. Phương pháp này giúp tối ưu hóa các siêu tham số và cải thiện hiệu suất tính toán.



Hình 2: Tối ưu hóa trong học máy với BIT*

- **Lập kế hoạch trong robot:** Trong các hệ thống robot tự động, BIT* giúp lên kế hoạch di chuyển trong không gian phức tạp, tránh va chạm và tối ưu hóa lộ trình.

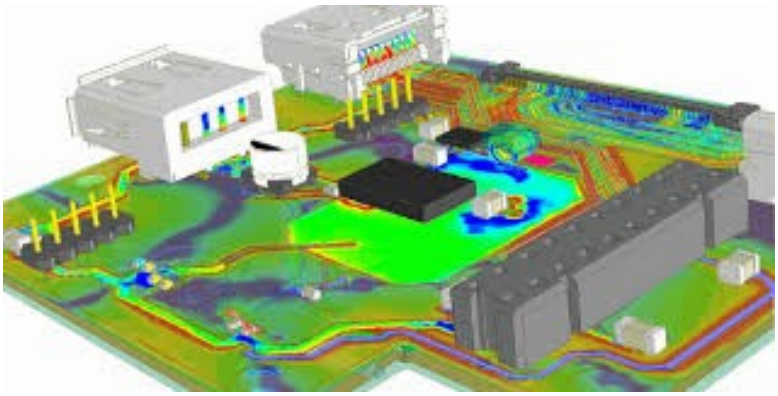
Hình 3: Lập kế hoạch di chuyển cho robot sử dụng BIT*

- **Giải quyết các bài toán tìm kiếm trong không gian trạng thái lớn:** BIT* được áp dụng trong các bài toán có không gian trạng thái quá lớn và có thể thay đổi trong quá trình tìm kiếm. Thuật toán này giúp giảm thiểu chi phí tính toán và cải thiện hiệu suất tìm kiếm.
- **Ứng dụng trong chơi game AI:** Trong các trò chơi điện tử, BIT* có thể giúp

các đối thủ AI tìm kiếm chiến lược chơi tối ưu, dựa trên các chiến thuật đã học được trong quá trình chơi.

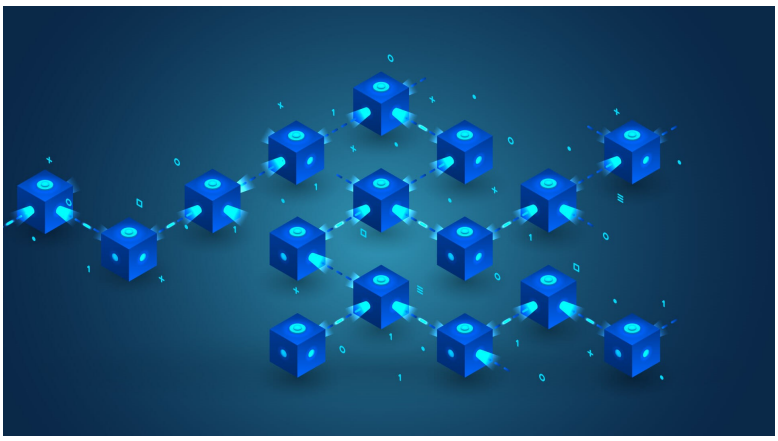
Hình 4: Ứng dụng BIT* trong AI chơi game

- **Mô phỏng và tối ưu hóa trong khoa học máy tính:** BIT* cũng được sử dụng trong các mô phỏng phức tạp và tối ưu hóa các hệ thống trong khoa học máy tính, giúp giảm thiểu thời gian tính toán và đạt được kết quả tối ưu.



Hình 5: Mô phỏng và tối ưu hóa trong khoa học máy tính với BIT*

- **Hệ thống phân tán và mạng lưới:** BIT* có thể tối ưu hóa quá trình tìm kiếm và lập kế hoạch trong các hệ thống phân tán hoặc mạng lưới phức tạp, giúp cải thiện hiệu suất trong các tình huống có độ trễ cao.



Hình 6: Ứng dụng BIT* trong các hệ thống phân tán

Chương 2

So sánh các thuật toán quy hoạch đường đi

2.1 Tiêu chí so sánh

Trong bài toán lập kế hoạch đường đi, có nhiều thuật toán được thiết kế để hoạt động hiệu quả trong những bối cảnh khác nhau. Để lựa chọn thuật toán phù hợp, cần đánh giá chúng dựa trên các tiêu chí sau:

- **Môi trường phù hợp**: Khả năng hoạt động tốt trong không gian nhỏ, không gian lớn hoặc có nhiều chướng ngại vật.
- **Tối ưu dần (Asymptotic Optimality)**: Thuật toán có hội tụ dần đến lời giải tối ưu theo thời gian hay không.
- **Phụ thuộc heuristic**: Mức độ thuật toán cần sử dụng các hàm ước lượng (heuristic).
- **Độ phức tạp cài đặt**: Độ khó khi triển khai thuật toán trong thực tế.
- **Tốc độ tìm đường**: Thời gian cần để tìm ra lời giải khả thi đầu tiên.
- **Chất lượng đường đi**: Mức độ tối ưu của đường đi được tìm thấy.
- **Phù hợp với động học**: Có thể áp dụng cho hệ động học của robot hay không.
- **Học từ dữ liệu**: Có khả năng kết hợp với học máy để cải thiện hiệu năng.

2.2 Bảng so sánh tổng quan các thuật toán

Thuật toán	**Môi trường phù hợp**	**Tối ưu dần**	**Phụ thuộc heuristic**	**Cài đặt phức tạp**	**Tốc độ tìm đường**	**Chất lượng đường đi**	**Động học**	**Học từ dữ liệu**
A*	Lưới 2D, không gian nhỏ	✗	✓	Trung bình	Nhanh	Tốt nếu h tốt	✗	✗
RRT	Không gian lớn, nhiều chướng ngại	✗	✗	Trung bình	Rất nhanh	Kém	✓	✗
RRT*	Như RRT nhưng cần tối ưu hóa	✓	✗	Cao	Trung bình	Khá	✓	✗
BIT*	Không gian lớn, cần heuristic tốt	✓	✓	Rất cao	Chậm ban đầu	Rất tốt	✓	✗
RRG	Tái sử dụng đường, nhiều goal	✓	✗	Cao	Trung bình	Khá	✓	✗
PRM	Môi trường tĩnh, multi-query	✗	✗	Trung bình	Nhanh (sau khi build)	Trung bình	✗	✗
Neural RRT*	Phức tạp, học được từ trước	✓	✓	Rất cao	Nhanh (nếu học tốt)	Rất tốt	✓	✓
Informed Neural RRT*	Học + heuristic	✓	✓	Rất cao	Rất nhanh	Tốt nhất	✓	✓

2.3 Phân tích chi tiết từng thuật toán

+) A*

Thuật toán A* sử dụng cấu trúc đồ thị và khai thác hàm heuristic để tối ưu tìm kiếm đường đi. Mặc dù đơn giản và hiệu quả trong không gian nhỏ, A* không thích hợp

cho không gian nhiều chiều hoặc có động học phức tạp.

Hình 1: Robot vacuum cleaner

+) RRT và RRT*

RRT xây dựng cây tìm kiếm dựa trên mẫu ngẫu nhiên, rất hiệu quả trong không gian lớn và có chướng ngại vật. Tuy nhiên, nó không tối ưu. RRT* cải thiện điều này bằng cách tối ưu dần đường đi, nhưng đánh đổi bằng chi phí tính toán cao hơn.

+) BIT*

BIT* kết hợp tính ngẫu nhiên và heuristic, cho phép mở rộng hiệu quả không gian tìm kiếm. Nó có tính tối ưu dần và cho kết quả rất tốt nhưng chậm ở giai đoạn đầu và cài đặt phức tạp.

+) RRG và PRM

RRG lưu trữ các đường dẫn để tái sử dụng, thích hợp với nhiều mục tiêu. PRM phù hợp với môi trường tĩnh, hoạt động hiệu quả sau giai đoạn tiền xử lý.

+) Neural RRT* và Informed Neural RRT*

Những thuật toán này sử dụng mô hình học sâu để hướng dẫn quá trình tìm kiếm, từ đó tăng tốc đáng kể và cải thiện chất lượng đường đi. Tuy nhiên, yêu cầu dữ liệu huấn luyện và mô hình học phức tạp.

2.4 Nhận xét tổng quát

Từ bảng và phân tích trên, có thể rút ra một số nhận xét:

- Các thuật toán truyền thống như A*, RRT, PRM phù hợp cho các hệ thống đơn giản, yêu cầu tính toán nhanh.
- RRT*, RRG, BIT* là lựa chọn tốt hơn khi yêu cầu đường đi tối ưu, đặc biệt với các hệ thống robot có động học.
- Các thuật toán kết hợp học sâu như Neural RRT* hay Informed Neural RRT* thể hiện tiềm năng rất lớn nhờ khả năng học từ dữ liệu và thích nghi nhanh với môi trường.
- BIT* nổi bật nhờ kết hợp tốt giữa A* và RRT*, thích hợp cho bài toán cần tối ưu dần và định hướng tốt.

Chương 3

Cơ sở lý thuyết

3.1 Hàm heuristic trong BIT*

Heuristic (tiếng Việt: hàm ước lượng) là một phương pháp nhằm đưa ra ước lượng chi phí từ một trạng thái hiện tại đến mục tiêu. Trong thuật toán **BIT***, hàm heuristic được sử dụng để đánh giá tổng chi phí ước lượng của một đỉnh (node) x thông qua công thức:

$$f(x) = g(x) + h(x) \tag{3.1}$$

Trong đó:

- x: một đỉnh trong không gian cấu hình.
- $g(x)$: chi phí thực tế từ đỉnh bắt đầu đến đỉnh x.
- $h(x)$: chi phí ước lượng (heuristic) từ x đến đích.
- $f(x)$: tổng chi phí ước lượng nếu đi qua đỉnh x.

Ý nghĩa của hàm heuristic:

- Tập trung vào các vùng không gian có khả năng cao dẫn tới lời giải tốt.
- Giảm số lượng mẫu và cạnh cần xem xét.
- Định hướng tìm kiếm theo hướng gần với đường đi tốt nhất.

3.2 Chiến lược Back Sampling

Back sampling là chiến lược lấy mẫu **ngược từ đích về gốc**, thay vì lấy mẫu hoàn toàn ngẫu nhiên. Mục tiêu của chiến lược này là:

- Lấy mẫu ở những vùng có khả năng góp phần vào đường đi tối ưu.

- Hướng dẫn quá trình lấy mẫu dựa trên thông tin từ đường đi hiện tại, chi phí còn lại (cost-to-go) hoặc heuristic.

Ứng dụng:

- Tìm ra các đường đi ngắn hơn.
- Tối ưu hóa lộ trình (đường ít va chạm, ngắn nhất, mượt nhất).

Một điểm x được chấp nhận để lấy mẫu nếu thoả mãn điều kiện:

$$g(x) + h(x) \leq c_{\text{best}} \tag{3.2}$$

Trong đó:

- $g(x)$: chi phí từ điểm bắt đầu đến x.
- $h(x)$: chi phí ước lượng từ x đến đích.
- c_{best}: chi phí của đường đi tốt nhất hiện tại (nếu có).

Ví dụ: Nếu điểm đến ở xa và bạn đã có một đường đi tạm thời với chi phí $c_{\text{best}} = 100$ đơn vị, thì chỉ nên lấy mẫu ở những điểm x sao cho $g(x) + h(x) \leq 100$, vì những điểm này mới có khả năng tạo ra đường đi ngắn hơn đường hiện có.

3.3 Thu hẹp không gian tìm kiếm bằng Ellipsoid

Ellipsoid là kỹ thuật cắt giảm không gian tìm kiếm (search space) bằng cách thu hẹp dần vùng chứa lời giải dưới dạng một **hình elip trong không gian nhiều chiều**.

Ứng dụng:

- Trong các bài toán tối ưu hóa lồi (convex optimization).
- Tìm nghiệm tối ưu bằng cách lặp lại việc "cắt" nhỏ không gian tìm kiếm.

Khi đã tìm được một đường đi có chi phí c_{best}, thay vì tiếp tục lấy mẫu toàn bộ không gian, thuật toán chỉ lấy mẫu trong một hình ellipsoid thoả mãn:

$$\|x - x_{\text{start}}\| + \|x - x_{\text{goal}}\| \leq c_{\text{best}} \tag{3.3}$$

Ví dụ:

- $x_{\text{start}} = (0, 0)$, $x_{\text{goal}} = (10, 0)$.

- Đường đi hiện tại dài 12 đơn vị $\Rightarrow c_{\text{best}} = 12$.
- Khi đó, không gian mẫu sẽ chỉ bao quanh các điểm x sao cho:

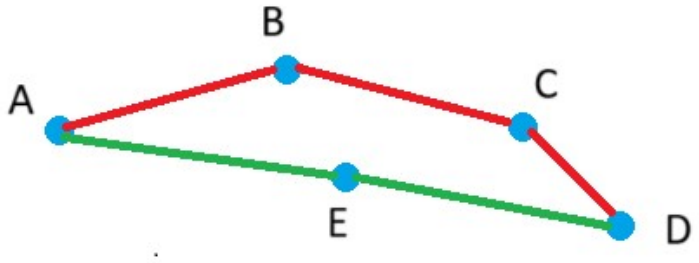
$$\|x - (0,0)\| + \|x - (10,0)\| \leq 12$$

Vùng này chứa tất cả các điểm có khả năng nằm trên đường đi ngắn hơn hoặc bằng 12 đơn vị từ start đến goal, giúp loại bỏ các vùng không cần thiết và tăng hiệu quả tìm kiếm.

3.4 Rewiring

Tưởng tượng như chơi game tìm đường trên bản đồ:

- Ban đầu bạn đi từ: A → B → C → D
- Nhưng sau đó bạn phát hiện ra đường A → E → D ngắn hơn!
- Bạn quyết định nối lại dây qua E, bỏ qua B và C cho nhanh hơn.



Hình 1: Ví dụ minh họa quá trình Rewiring: Đường đi mới (xanh) ngắn hơn so với đường cũ (đỏ)

Công thức Rewiring:

$$\text{cost}(x_{\text{new}}) = \min\{\text{cost}(x_{\text{near}}) + c(x_{\text{near}}, x_{\text{new}})\}$$

Giải thích:

- x_{new}: điểm mới muốn thêm vào cây
- x_{near}: các điểm "gần" trong cây hiện tại
- $c(x_{\text{near}}, x_{\text{new}})$: chi phí từ x_{near} đến x_{new}
- $\text{cost}(x_{\text{new}})$: tổng chi phí tốt nhất từ gốc đến x_{new}

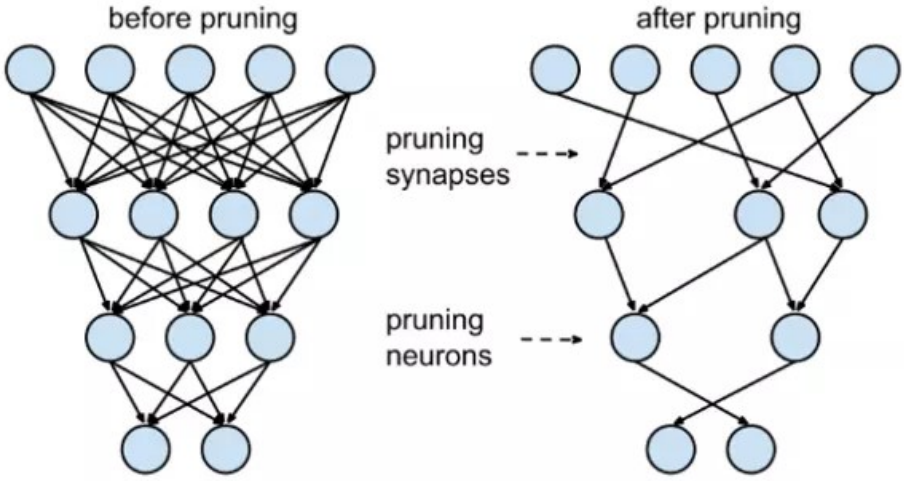
Mục tiêu: tìm đường ngắn nhất đến x_{new} qua các điểm gần nó.
Ứng dụng:

- Tìm đường đi ngắn nhất trong bản đồ (như GPS, robot tự hành)
- Tối ưu hóa đường đi trong RRT*, BIT* hoặc các thuật toán tìm đường ngẫu nhiên
- Tối ưu chi phí di chuyển trong mạng lưới (ví dụ mạng điện, giao thông)

3.5 Pruning – Cắt bỏ nhánh không cần thiết

Tưởng tượng bạn đang đi tìm kho báu, nhưng trên bản đồ có rất nhiều ngã rẽ:

- Có những ngã rẽ đi vòng vo, xa xôi
- Bạn quyết định bỏ qua những đường không khả thi.



Hình 2: Minh họa ý tưởng Pruning – giống như cắt bỏ các nhánh dư thừa

Điều kiện cắt tỉa:

$$g(x) + h(x) \geq c_{\text{best}}$$

Giải thích:

- $g(x)$: chi phí từ điểm gốc đến x
- $h(x)$: chi phí ước lượng từ x đến đích
- c_{best}: chi phí đường đi tốt nhất hiện tại

Nếu: tổng chi phí này lớn hơn $c_{\text{best}} \rightarrow$ loại bỏ x, vì nó không giúp cải thiện đường đi.

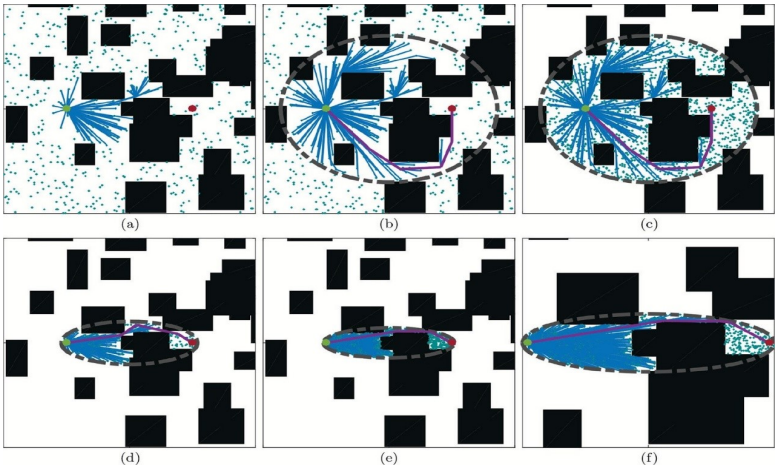
Ứng dụng:

- Tăng tốc độ tìm kiếm trong thuật toán A*, RRT*, BIT*
- Giảm thời gian xử lý trong hệ thống AI thời gian thực
- Giảm chi phí tính toán trong mạng neural, cây tìm kiếm hoặc heuristic

3.6 Bán kính RGG

Hình dung bạn là học sinh trong lớp:

- Bạn chỉ trò chuyện với những người ngồi gần bạn, ví dụ trong bán kính 2 mét
- Nếu là một điểm trong bản đồ, bạn kết nối với các điểm gần mình
- Càng nhiều điểm $\rightarrow$ vòng tròn kết nối phải nhỏ lại để tránh rối



Hình 3: Ví dụ minh họa mô hình RGG – Chỉ kết nối với các điểm trong bán kính cho phép

Công thức bán kính RGG trong BIT*:

$$r_{BIT^*} = \gamma \left(\frac{\log(n)}{n} \right)^{1/d}$$

Giải thích:

- n: số lượng điểm (nút)
- d: số chiều không gian (ví dụ 2D: $d = 2$)
- γ: hằng số điều chỉnh

Ý nghĩa: đảm bảo chỉ kết nối với điểm gần, giúp đồ thị gọn gàng mà vẫn hiệu quả.
Ứng dụng:

- Giảm số lượng kết nối cần kiểm tra trong đồ thị (đỡ lag, nhanh hơn)
- Xây dựng cây tìm đường hiệu quả hơn trong RRT*, BIT*
- Dùng trong sensor networks, đồ thị mạng không dây

Chương 4

Phân tích chi tiết thuật toán Batch Informed Trees (BIT*)

4.1 Mô hình toán học và kí hiệu

- Không gian trạng thái: $\mathcal{X} \subseteq \mathbb{R}^d$.
- Trạng thái khởi đầu: $x_{\text{start}} \in \mathcal{X}$; trạng thái đích: $x_{\text{goal}} \in \mathcal{X}$.
- Cây tìm kiếm (V, E) với V là tập đỉnh, E là tập cạnh.
- Tập mẫu: X_{sample}.
- Hàm chi phí di chuyển giữa hai điểm: $c(v, x)$.
- Hàm heuristic ước lượng chi phí đến đích: $h(x)$.
- $g(v)$: chi phí đường đi tốt nhất từ x_{start} đến v.
- Chi phí đường đi tốt nhất hiện tại: c_{best}.
- Bán kính kết nối RGG:

$$r = \gamma \left(\frac{\log q}{q} \right)^{1/d}, \quad q = |V| + |X_{\text{sample}}|.$$

4.2 Thuật toán Batch Informed Trees (BIT*)

Algorithm 1 Batch Informed Trees (BIT*)

Khởi tạo:
  $V \leftarrow \{x_{\text{start}}\}$, $E \leftarrow \emptyset$
  $X_{\text{sample}} \leftarrow \emptyset$
  $Q_V \leftarrow \emptyset$, $Q_E \leftarrow \emptyset$
  $c_{\text{best}} \leftarrow \infty$
while không hội tụ **do**
  if $Q_E = \emptyset$ **và** $Q_V = \emptyset$ **then**
    Tỉa cây theo c_{best}
    Tỉa mẫu theo c_{best}
    Sinh thêm mẫu trong ellipsoid thông tin
    Thêm mẫu vào X_{sample}
    Cập nhật r dựa trên $|V| + |X_{\text{sample}}|$
    Nạp lại Q_V theo $g(v) + h(v)$ tăng dần
  end if
  while $Q_E = \emptyset$ **và** $Q_V \neq \emptyset$ **do**
    Lấy v đầu từ Q_V
    if $g(v) + h(v) < c_{\text{best}}$ **then**
      Tìm $x \in X_{\text{sample}}$ sao cho $\|v - x\| \leq r$ và $g(v) + c(v, x) + h(x) < c_{\text{best}}$
      Đưa các cạnh (v, x) đó vào Q_E theo $g(v) + c(v, x) + h(x)$
    end if
  end while
  if $Q_E \neq \emptyset$ **then**
    Lấy cạnh (v, x) có chi phí thấp nhất từ Q_E
    if $x \notin V$ **hoặc** $g(v) + c(v, x) < g(x)$ **then**
      if va chạm giữa v và x **then**
        Bỏ qua cạnh
      else
        $V \leftarrow V \cup \{x\}$, $E \leftarrow E \cup \{(v, x)\}$
        $g(x) \leftarrow g(v) + c(v, x)$
        Thêm x vào Q_V
        Cập nhật c_{best} nếu $x = x_{\text{goal}}$
      end if
    end if
  end if
end while
return cây (V, E) và chi phí tốt nhất c_{best}

4.3 Giải thích chi tiết các bước

- **Tỉa cây/mẫu:** Loại bỏ các đỉnh và mẫu không có khả năng cải thiện chi phí tốt nhất hiện tại.
- **Sinh mẫu trong ellipsoid:** Tập trung vào vùng khả thi chứa đường đi ngắn hơn c_{best}.

- **Bán kính r:** Giới hạn số lượng cạnh trong RGG, giữ cho thuật toán hiệu quả.
- **Hàng đợi đỉnh Q_V:** Sắp xếp theo $g + h$, ưu tiên mở rộng đỉnh có tiềm năng tốt.
- **Hàng đợi cạnh Q_E:** Cạnh có chi phí $g + c + h$ nhỏ được xử lý trước để tiết kiệm kiểm tra va chạm.
- **Kiểm tra và cập nhật:** Nếu cạnh hợp lệ và cải thiện, cập nhật cây và chi phí.

4.4 Tính chất thuật toán

4.4.1 Đảm bảo xác suất đầy đủ (Probabilistic Completeness)

BIT* đảm bảo tìm ra lời giải nếu tồn tại khi số lượng mẫu tiến đến vô hạn.

4.4.2 Tối ưu tiệm cận (Asymptotic Optimality)

Khi số mẫu tăng, thuật toán sẽ dần hội tụ đến lời giải tối ưu, nhờ việc tỉa và heuristic hợp lý.

4.5 Phân tích độ phức tạp

- Chi phí tính RGG: $O(q \log q + q \cdot k)$ với k là số lân cận trung bình.
- Mỗi batch gồm: tạo mẫu, tính r, tỉa mẫu/cây, tạo Q_V, Q_E.
- Tổng thể tiết kiệm kiểm tra va chạm so với RRT* nhờ kiểm soát cạnh cần xét.

4.6 Kết luận chương

Thuật toán BIT* là sự kết hợp giữa heuristic dẫn đường (giống A*), tạo mẫu theo lô và xây đồ thị ngẫu nhiên. Các bước như tỉa cây/mẫu và sắp xếp hàng đợi theo chi phí giúp tăng hiệu quả rõ rệt. Chương tiếp theo sẽ trình bày triển khai chi tiết và kết quả đánh giá thực nghiệm.

Chương 5

Mã nguồn thuật toán

```python
# Import cac thu vien can thiet
import argparse  # Xu ly tham so dong lenh
import json      # Lam viec voi file JSON
import os        # Thao tac voi he thong file
import random    # Tao so ngau nhien
import shutil    # Thao tac voi file va thu muc
import sys       # Thao tac voi he thong
import time      # Thao tac voi thoi gian
from datetime import datetime  # Xu ly ngay gio
from queue import PriorityQueue  # Hang doi uu tien
from typing import List, Set, Tuple, Generic, TypeVar  # Annotation type

import cv2       # Xu ly anh
import matplotlib.pyplot as plt  # Ve do thi
import numpy as np  # Tinh toan so
from matplotlib.patches import Ellipse  # Ve hinh ellipse
from PIL import Image, ImageOps  # Xu ly anh
import tqdm      # Thanh tien trinh

# Mau sac cho terminal
class MauTerminal:
    HEADER = '\033[95m'  # Mau header
    BLUE = '\033[94m'    # Mau xanh nuoc bien
    CYAN = '\033[96m'    # Mau xanh lam
    GREEN = '\033[92m'   # Mau xanh la
    YELLOW = '\033[93m'  # Mau vang
    RED = '\033[91m'     # Mau do
    BOLD = '\033[1m'     # Dam
    UNDERLINE = '\033[4m' # Gach chan
    END = '\033[0m'      # Ket thuc mau

    # Mau trang thai
    THANH_CONG = '\033[92m'  # Thanh cong
    CANH_BAO = '\033[93m'    # Canh bao
    LOI = '\033[91m'         # Loi
    THONG_TIN = '\033[94m'   # Thong tin

    # Mau nen
    NEN_XANH = '\033[44m'    # Nen xanh
    NEN_LUC = '\033[42m'     # Nen luc
    NEN_VANG = '\033[43m'    # Nen vang
    NEN_DO = '\033[41m'      # Nen do

# Bien toan cuc cho lop Node
start_arr = None  # Diem bat dau
goal_arr = None   # Diem dich
```

```python

# Kieu du lieu chung cho Node cha
T = TypeVar("T")

class Node(Generic[T]):
    """Lop dai dien mot nut trong thuat toan BIT*"""

    def __init__(
        self,
        toa_do: tuple,     # Toa do (x,y)
        cha: T = None,     # Nut cha
        gt: float = np.inf,  # Chi phi tu nut goc
        chi_phi_cha: float = None,  # Chi phi tu nut cha
    ) -> None:
        self.x = toa_do[0]  # Toa do x
        self.y = toa_do[1]  # Toa do y
        self.tup = (self.x, self.y)  # Luu duoi dang tuple
        self.np_arr = np.array([self.x, self.y])  # Luu duoi dang numpy array
        self.cha = cha      # Nut cha
        self.chi_phi_cha = chi_phi_cha  # Chi phi tu nut cha
        self.gt = gt        # Chi phi tu nut goc
        self.con = set()    # Tap cac nut con
        self.bat_dau = start_arr  # Diem bat dau
        self.dich = goal_arr      # Diem dich
        self.g_hat = self.tinh_g_hat()  # Uoc luong chi phi tu nut goc
        self.h_hat = self.tinh_h_hat()  # Uoc luong chi phi den dich
        self.f_hat = self.g_hat + self.h_hat  # Tong uoc luong

    def tinh_g_hat(self) -> float:
        # Tinh khoang cach tu nut hien tai den nut goc
        return np.linalg.norm(self.np_arr - self.bat_dau)

    def tinh_h_hat(self) -> float:
        # Tinh khoang cach tu nut hien tai den dich
        return np.linalg.norm(self.np_arr - self.dich)

    def __str__(self) -> str:
        return str(self.tup)

    def __repr__(self) -> str:
        return str(self.tup)

class BanDo:
    """Lop dai dien ban do cho thuat toan BIT*"""

    def __init__(self, bat_dau: Node, dich: Node, luoi_chan: np.array) -> None:
        self.bat_dau = bat_dau  # Nut bat dau
        self.dich = dich        # Nut dich
        self.bat_dau_arr = self.bat_dau.np_arr  # Toa do bat dau
        self.dich_arr = self.dich.np_arr        # Toa do dich
        self.chan = set()       # Tap cac vat can
        self.chieu = 2          # So chieu (2D)
        self.ban_do = luoi_chan  # Luoi chan
        # Tim cac o trong
        vi_tri = np.argwhere(self.ban_do > 0)
        self.trong = set([tuple(x) for x in vi_tri])
        # Tim cac o bi chiem
        vi_tri = np.argwhere(self.ban_do == 0)
        self.bi_chiem = set([tuple(x) for x in vi_tri])
        self.tinh_ban_do_f_hat()

    def lay_mau_moi(self) -> tuple:
```

```
        # Lay ngau nhien mot diem trong khong gian trong
        while True:
            nut_trong = random.sample(list(self.trong), 1)[0]
            nhieu = np.random.uniform(0, 1, self.chieu)
            nut_moi = nut_trong + nhieu
            if (int(nut_moi[0]), int(nut_moi[1])) in self.trong:
                return nut_moi

    def tinh_ban_do_f_hat(self) -> None:
        # Tinh ban do uoc luong chi phi f_hat cho toan bo ban do
        kich_thuoc_x, kich_thuoc_y = self.ban_do.shape
        self.ban_do_f_hat = np.zeros((kich_thuoc_x, kich_thuoc_y))
        for x in range(kich_thuoc_x):
            for y in range(kich_thuoc_y):
                f_hat = np.linalg.norm(
                    np.array([x, y]) - self.dich_arr
                ) + np.linalg.norm(np.array([x, y]) - self.bat_dau_arr)
                self.ban_do_f_hat[x, y] = f_hat

class BITStar:
    """Lop trien khai thuat toan BIT* de lap ke hoach duong di"""

    def __init__(
        self,
        bat_dau: Node,
        dich: Node,
        ban_do_chan: BanDo,
        so_mau: int = 20,
        rbit: float = 100,
        chieu: int = 2,
        thu_muc_log: str = None,
        thoi_gian_dung: int = 60,
    ) -> None:
        self.bat_dau = bat_dau  # Diem bat dau
        self.dich = dich        # Diem dich
        self.ban_do = ban_do_chan  # Ban do
        self.chieu = chieu      # So chieu
        self.rbit = rbit        # Ban kinh toi da
        self.m = so_mau         # So mau moi lan
        self.ci = np.inf        # Chi phi hien tai
        self.ci_cu = np.inf     # Chi phi truoc do
        self.cmin = np.linalg.norm(self.dich.np_arr - self.bat_dau.np_arr)  #
            Chi phi toi thieu
        self.ban_do_phang = self.ban_do.ban_do.flatten()  # Ban do 1 chieu
        self.thoi_gian_dung = thoi_gian_dung  # Thoi gian dung
        self.V = set()          # Tap cac nut
        self.E = set()          # Tap cac canh
        self.E_hien_thi = set() # Tap canh de hien thi
        self.x_moi = set()      # Tap nut moi
        self.x_tai_su_dung = set()  # Tap nut tai su dung
        self.chua_mo_rong = set()   # Tap nut chua mo rong
        self.chua_ket_noi = set()   # Tap nut chua ket noi
        self.v_giai_phap = set()    # Tap nut trong duong di
        self.hang_doi_v = PriorityQueue()  # Hang doi nut
        self.hang_doi_e = PriorityQueue()  # Hang doi canh
        self.thu_tu_e = 0       # Thu tu canh
        self.thu_tu_v = 0       # Thu tu nut

        # Khoi tao
        self.V.add(bat_dau)
        self.chua_ket_noi.add(dich)
        self.chua_mo_rong = self.V.copy()
```

```
        self.x_moi = self.chua_ket_noi.copy()
        self.hang_doi_v.put((bat_dau.gt + bat_dau.h_hat, self.thu_tu_v, bat_dau)
            )
        self.thu_tu_v += 1
        self.lay_PHS()
        self.luu = False
        if thu_muc_log is not None:
            self.luu = True
            self.thu_muc_log = thu_muc_log
            self.noi_dung_json = {
                "canh_moi": [],
                "canh_xoa": [],
                "duong_di_cuoi": [],
                "ci": [],
                "danh_sach_canh_cuoi": [],
            }

    def gt(self, nut: Node) -> float:
        # Tinh chi phi tu nut goc den nut hien tai
        if nut == self.bat_dau:
            return 0
        elif nut not in self.V:
            return np.inf
        return nut.chi_phi_cha + nut.cha.gt

    def c_hat(self, nut1: Node, nut2: Node) -> float:
        # Uoc luong chi phi giua 2 nut
        return np.linalg.norm(nut1.np_arr - nut2.np_arr)

    def a_hat(self, nut1: Node, nut2: Node) -> float:
        # Uoc luong tong chi phi qua 2 nut
        return nut1.g_hat + self.c_hat(nut1, nut2) + nut2.h_hat

    def c(self, nut1: Node, nut2: Node, ty_le: int = 10) -> float:
        # Tinh chi phi thuc te giua 2 nut (kiem tra vat can)
        x1, y1 = nut1.tup
        x2, y2 = nut2.tup
        so_doan = int(ty_le * np.linalg.norm(nut1.np_arr - nut2.np_arr))
        for lam in np.linspace(0, 1, so_doan):
            x = int(x1 + lam * (x2 - x1))
            y = int(y1 + lam * (y2 - y1))
            if (x, y) in self.ban_do.bi_chiem:
                return np.inf
        return self.c_hat(nut1, nut2)

    def gan(self, tap_tim: set, nut: Node) -> set:
        # Tim cac nut gan nut hien tai trong ban kinh rbit
        gan = set()
        for n in tap_tim:
            if (self.c_hat(n, nut) <= self.rbit) and (n != nut):
                gan.add(n)
        return gan

    def mo_rong_nut_tiep_theo(self) -> None:
        # Mo rong nut tiep theo trong hang doi
        v_min = self.hang_doi_v.get(False)[2]
        x_gan = None
        if v_min in self.chua_mo_rong:
            x_gan = self.gan(self.chua_ket_noi, v_min)
        else:
            giao = self.chua_ket_noi & self.x_moi
            x_gan = self.gan(giao, v_min)
```

```
231         for x in x_gan:
232             if self.a_hat(v_min, x) < self.ci:
233                 chi_phi = v_min.gt + self.c(v_min, x) + x.h_hat
234                 self.hang_doi_e.put((chi_phi, self.thu_tu_e, (v_min, x)))
235                 self.thu_tu_e += 1
236         if v_min in self.chua_mo_rong:
237             v_gan = self.gan(self.V, v_min)
238             for v in v_gan:
239                 if (
240                     (not (v_min, v) in self.E)
241                     and (self.a_hat(v_min, v) < self.ci)
242                     and (v_min.g_hat + self.c_hat(v_min, v) < v.gt)
243                 ):
244                     chi_phi = v_min.gt + self.c(v_min, v) + v.h_hat
245                     self.hang_doi_e.put((chi_phi, self.thu_tu_e, (v_min, v)))
246                     self.thu_tu_e += 1
247             self.chua_mo_rong.remove(v_min)
248 
249     def lay_mau_hinh_cau_don_vi(self) -> np.array:
250         # Lay mau ngau nhien trong hinh cau don vi
251         u = np.random.uniform(-1, 1, self.chieu)
252         chuan = np.linalg.norm(u)
253         r = np.random.random() ** (1.0 / self.chieu)
254         return r * u / chuan
255 
256     def lay_mau_PHS(self) -> np.array:
257         # Lay mau trong vung PHS (Prolate Hyperspheroid)
258         tam = (self.bat_dau.np_arr + self.dich.np_arr) / 2
259         a1 = (self.dich.np_arr - self.bat_dau.np_arr) / self.cmin
260         one_1 = np.eye(a1.shape[0])[:, 0]
261         U, S, Vt = np.linalg.svd(np.outer(a1, one_1.T))
262         Sigma = np.diag(S)
263         lam = np.eye(Sigma.shape[0])
264         lam[-1, -1] = np.linalg.det(U) * np.linalg.det(Vt.T)
265         cwe = np.matmul(U, np.matmul(lam, Vt))
266         r1 = self.ci / 2
267         rn = [np.sqrt(self.ci**2 - self.cmin**2) / 2] * (self.chieu - 1)
268         r = np.array([r1] + rn)
269         while True:
270             try:
271                 x_ball = self.lay_mau_hinh_cau_don_vi()
272                 op = np.matmul(np.matmul(cwe, r), x_ball) + tam
273                 op = np.around(op, 7)
274                 if (int(op[0]), int(op[1])) in self.giao:
275                     break
276             except:
277                 print(MauTerminal.BOLD, MauTerminal.LOI, op, x_ball, r, self.
                    cmin, self.ci, cwe, MauTerminal.END)
278                 exit()
279         return op
280 
281     def lay_PHS(self) -> None:
282         # Xac dinh vung PHS
283         self.xphs = set([tuple(x) for x in np.argwhere(self.ban_do.ban_do_f_hat
                < self.ci)])
284         self.giao = self.xphs & self.ban_do.trong
285 
286     def lay_mau(self) -> Node:
287         # Lay mau ngau nhien trong khong gian tim kiem
288         xrand = None
289         if self.ci_cu != self.ci:
290             self.lay_PHS()
```

```
        if len(self.xphs) < len(self.ban_do_phang):
            xrand = self.lay_mau_PHS()
        else:
            xrand = self.ban_do.lay_mau_moi()
        return Node(xrand)

    def cat_tia(self) -> None:
        # Loai bo cac nut khong can thiet
        self.x_tai_su_dung = set()
        chua_ket_noi_moi = set()
        for n in self.chua_ket_noi:
            if n.f_hat < self.ci:
                chua_ket_noi_moi.add(n)
        self.chua_ket_noi = chua_ket_noi_moi
        canh_xoa = []
        nut_sap_xep = sorted(self.V, key=lambda x: x.gt, reverse=True)
        for v in nut_sap_xep:
            if v != self.bat_dau and v != self.dich:
                if (v.f_hat > self.ci) or (v.gt + v.h_hat > self.ci):
                    self.V.discard(v)
                    self.v_giai_phap.discard(v)
                    self.chua_mo_rong.discard(v)
                    self.E.discard((v.cha, v))
                    self.E_hien_thi.discard((v.cha.tup, v.tup))
                    if self.luu:
                        canh_xoa.append((v.cha.tup, v.tup))
                    v.cha.con.remove(v)
                    if v.f_hat < self.ci:
                        self.x_tai_su_dung.add(v)
                    else:
                        del v
        if self.luu:
            self.luu_du_lieu(None, canh_xoa)
        self.chua_ket_noi.add(self.dich)

    def giai_phap_cuoi_cung(self) -> Tuple[List[Tuple[float, float]], float]:
        # Tra ve duong di tot nhat tim duoc
        if self.dich.gt == np.inf:
            return None, None
        duong_di = []
        do_dai_duong_di = 0
        nut = self.dich
        while nut != self.bat_dau:
            duong_di.append(nut.tup)
            do_dai_duong_di += nut.chi_phi_cha
            nut = nut.cha
        duong_di.append(self.bat_dau.tup)
        return duong_di[::-1], do_dai_duong_di

    def cap_nhat_gt_con(self, nut: Node) -> None:
        # Cap nhat chi phi cho tat ca cac nut con
        for c in nut.con:
            c.gt = c.chi_phi_cha + nut.gt
            self.cap_nhat_gt_con(c)

    def luu_du_lieu(
        self, canh_moi: tuple, canh_xoa: list, giai_phap_moi: bool = False
    ) -> None:
        # Luu du lieu qua trinh tim kiem
        self.noi_dung_json["ci"].append(self.ci)
        self.noi_dung_json["canh_moi"].append(canh_moi)
        self.noi_dung_json["canh_xoa"].append(canh_xoa)
```

```
353             if giai_phap_moi:
354                 giai_phap_hien_tai, _ = self.giai_phap_cuoi_cung()
355                 self.noi_dung_json["duong_di_cuoi"].append(giai_phap_hien_tai)
356             else:
357                 self.noi_dung_json["duong_di_cuoi"].append(None)
358
359         def xuat_du_lieu(self, so_lan_dat_dich: int) -> None:
360             # Xuat du lieu ra file JSON
361             print(f"{MauTerminal.NEN_LUC}Du lieu da luu.{MauTerminal.END}")
362             self.noi_dung_json["danh_sach_canh_cuoi"] = list(self.E_hien_thi)
363             json_object = json.dumps(self.noi_dung_json, indent=4)
364             with open(
365                 f"{self.thu_muc_log}/duong_di{so_lan_dat_dich:02d}.json",
366                 "w",
367             ) as outfile:
368                 outfile.write(json_object)
369             self.noi_dung_json = {
370                 "canh_moi": [],
371                 "canh_xoa": [],
372                 "duong_di_cuoi": [],
373                 "ci": [],
374                 "danh_sach_canh_cuoi": [],
375             }
376
377         def lap_ke_hoach(self) -> Tuple[List[Tuple[float, float]], float, List[float
                ]]:
378             # Ham chinh thuc hien thuat toan
379             if self.bat_dau.tup not in self.ban_do.trong or self.dich.tup not in
                    self.ban_do.trong:
380                 print(f"{MauTerminal.CANH_BAO}Diem bat dau hoac dich khong trong
                        khong gian tu do.{MauTerminal.END}")
381                 return None, None, None
382             if self.bat_dau.tup == self.dich.tup:
383                 print(f"{MauTerminal.THANH_CONG}Diem bat dau va dich trung nhau.{
                        MauTerminal.END}")
384                 self.v_giai_phap.add(self.bat_dau)
385                 self.ci = 0
386                 return [self.bat_dau.tup], 0, None
387             vong_lap = 0
388             so_lan_dat_dich = 0
389             thoi_gian_bat_dau = time.time()
390             bat_dau = time.time()
391             thoi_gian_thuc_hien = []
392             try:
393                 while True:
394                     if time.time() - thoi_gian_bat_dau >= self.thoi_gian_dung:
395                         print(
396                             f"\n\n{MauTerminal.CANH_BAO
                                }========================================== Dung
                                do het thoi gian
                                ==========================================={
                                MauTerminal.END}"
397                         )
398                         duong_di, do_dai_duong_di = self.giai_phap_cuoi_cung()
399                         return duong_di, do_dai_duong_di, thoi_gian_thuc_hien
400                     vong_lap += 1
401                     if self.hang_doi_e.empty() and self.hang_doi_v.empty():
402                         self.cat_tia()
403                         mau_x = set()
404                         while len(mau_x) < self.m:
405                             mau_x.add(self.lay_mau())
406                         self.x_moi = self.x_tai_su_dung | mau_x
```

```
                    self.chua_ket_noi = self.chua_ket_noi | self.x_moi
                    for n in self.V:
                        self.hang_doi_v.put((n.gt + n.h_hat, self.thu_tu_v, n))
                        self.thu_tu_v += 1
                while True:
                    if self.hang_doi_v.empty():
                        break
                    self.mo_rong_nut_tiep_theo()
                    if self.hang_doi_e.empty():
                        continue
                    if self.hang_doi_v.empty() or self.hang_doi_v.queue[0][0] <=
                        self.hang_doi_e.queue[0][0]:
                        break
                if not (self.hang_doi_e.empty()):
                    (v_min, x_min) = self.hang_doi_e.get(False)[2]
                    if v_min.gt + self.c_hat(v_min, x_min) + x_min.h_hat < self.
                        ci:
                        if v_min.gt + self.c_hat(v_min, x_min) < x_min.gt:
                            chi_phi_canh = self.c(v_min, x_min)
                            if v_min.gt + chi_phi_canh + x_min.h_hat < self.ci:
                                if v_min.gt + chi_phi_canh < x_min.gt:
                                    canh_xoa = []
                                    if x_min in self.V:
                                        self.E.remove((x_min.cha, x_min))
                                        self.E_hien_thi.remove((x_min.cha.tup,
                                            x_min.tup))
                                        x_min.cha.con.remove(x_min)
                                        canh_xoa.append((x_min.cha.tup, x_min.
                                            tup))
                                        x_min.cha = v_min
                                        x_min.chi_phi_cha = chi_phi_canh
                                        x_min.gt = self.gt(x_min)
                                        self.E.add((x_min.cha, x_min))
                                        self.E_hien_thi.add((x_min.cha.tup,
                                            x_min.tup))
                                        x_min.cha.con.add(x_min)
                                        self.cap_nhat_gt_con(x_min)
                                    else:
                                        self.V.add(x_min)
                                        x_min.cha = v_min
                                        x_min.chi_phi_cha = chi_phi_canh
                                        x_min.gt = self.gt(x_min)
                                        self.E.add((x_min.cha, x_min))
                                        self.E_hien_thi.add((x_min.cha.tup,
                                            x_min.tup))
                                        self.thu_tu_v += 1
                                        self.chua_mo_rong.add(x_min)
                                        if x_min == self.dich:
                                            self.v_giai_phap.add(x_min)
                                        x_min.cha.con.add(x_min)
                                        self.chua_ket_noi.remove(x_min)
                                        canh_moi = (x_min.cha.tup, x_min.tup)
                                        self.ci = max(self.dich.gt, self.cmin)
                                        if self.luu:
                                            self.luu_du_lieu(
                                                canh_moi, canh_xoa, self.ci !=
                                                    self.ci_cu
                                            )
                                    if self.ci != self.ci_cu:
                                        if time.time() - thoi_gian_bat_dau >=
                                            self.thoi_gian_dung:
                                            print(
```

```
                                        f"\n\n{MauTerminal.CANH_BAO}═══════════════ Dung do het thoi gian ═══════════════{MauTerminal.END}"
                                    )
                                    duong_di, do_dai_duong_di = self.giai_phap_cuoi_cung()
                                    return duong_di, do_dai_duong_di, thoi_gian_thuc_hien
                                print(
                                    f"\n\n{MauTerminal.THANH_CONG}═══════════════ TIM THAY DICH LAN THU {so_lan_dat_dich:02d} ═══════════════{MauTerminal.END}"
                                )
                                print(
                                    f"{MauTerminal.BLUE}Thoi gian thuc hien:{MauTerminal.END} {time.time() - bat_dau}",
                                    end="\t\t",
                                )
                                thoi_gian_thuc_hien.append(time.time() - bat_dau)
                                bat_dau = time.time()
                                giai_phap, do_dai = self.giai_phap_cuoi_cung()
                                print(
                                    f"{MauTerminal.BLUE}Do dai duong di:{MauTerminal.END} {do_dai}{MauTerminal.END}"
                                )
                                print(
                                    f"{MauTerminal.BLUE}CI cu:{MauTerminal.END} {self.ci_cu}\t{MauTerminal.BLUE}CI moi:{MauTerminal.END} {self.ci}\t{MauTerminal.BLUE}ci - cmin:{MauTerminal.END} {round(self.ci - self.cmin, 5)}\t {MauTerminal.BLUE}Chenh lech CI:{MauTerminal.END} {round(self.ci_cu - self.ci, 5)}"
                                )
                                print(f"{MauTerminal.BLUE}Duong di:{MauTerminal.END} {giai_phap}")
                                self.ci_cu = self.ci
                                if self.luu:
                                    self.xuat_du_lieu(so_lan_dat_dich)
                                so_lan_dat_dich += 1
                        else:
                            self.hang_doi_e = PriorityQueue()
                            self.hang_doi_v = PriorityQueue()
                    else:
                        self.hang_doi_e = PriorityQueue()
                        self.hang_doi_v = PriorityQueue()
            except KeyboardInterrupt:
                print(time.time() - bat_dau)
                print(self.giai_phap_cuoi_cung())
                duong_di, do_dai_duong_di = self.giai_phap_cuoi_cung()
```

```
            return duong_di, do_dai_duong_di, thoi_gian_thuc_hien

class TrinhBay:
    """Lop hien thi qua trinh thuat toan"""

    def __init__(
        self, bat_dau: np.array, dich: np.array, ban_do_chan: np.array,
            thu_muc_ket_qua: str
    ) -> None:
        self.cac_canh = set()
        self.duong_di_cuoi = None
        self.ci = np.inf
        self.lan_sim = 0
        self.ban_do_chan = cv2.cvtColor(ban_do_chan, cv2.COLOR_BGR2RGB)
        self.bat_dau = bat_dau
        self.dich = dich
        (
            self.tat_ca_canh_moi,
            self.tat_ca_canh_xoa,
            self.tat_ca_duong_di,
            self.tat_ca_ci,
            self.tat_ca_danh_sach_canh,
        ) = (
            [],
            [],
            [],
            [],
            [],
        )
        self.cac_duong = {}
        self.fig, self.ax = plt.subplots(figsize=(20, 20))
        self.thu_muc_ket_qua = thu_muc_ket_qua

    def doc_json(self, thu_muc: str, so_lan_toi_da: int = np.inf) -> None:
        # Doc du lieu tu file JSON
        cac_file = sorted(os.listdir(thu_muc))
        so_lan_toi_da = min(so_lan_toi_da, len(cac_file))
        for i in tqdm.tqdm(range(so_lan_toi_da), desc="Dang doc file JSON",
            total=so_lan_toi_da):
            with open(os.path.join(thu_muc, cac_file[i]), "r") as f:
                du_lieu = json.load(f)
                self.tat_ca_canh_moi.append(du_lieu["canh_moi"])
                self.tat_ca_canh_xoa.append(du_lieu["canh_xoa"])
                self.tat_ca_duong_di.append(du_lieu["duong_di_cuoi"])
                self.tat_ca_ci.append(np.array(du_lieu["ci"]))
                self.tat_ca_danh_sach_canh.append(du_lieu["danh_sach_canh_cuoi"
                    ])

    def ve_duong_di_cuoi(self, duong_di: List[Tuple[float, float]]) -> None:
        # Ve duong di cuoi cung
        if len(duong_di) == 0:
            return
        duong_di = np.array(duong_di)
        x, y = duong_di[:, 0], duong_di[:, 1]
        self.ax.plot(
            y,
            x,
            color="darkorchid",
            lw=4,
            label="Duong di cuoi",
        )

```

```
555     def ve_ellipse(self, ci: float, mau: str = "dimgrey") -> None:
556         # Ve hinh ellipse dai dien cho vung tim kiem
557         if ci == np.inf:
558             return
559         cmin = np.linalg.norm(self.dich - self.bat_dau)
560         tam = (self.bat_dau + self.dich) / 2.0
561         r1 = ci
562         r2 = np.sqrt(ci**2 - cmin**2)
563         goc = np.arctan2(self.dich[0] - self.bat_dau[0], self.dich[1] - self.
                bat_dau[1])
564         goc = np.degrees(goc)
565         patch = Ellipse(
566             xy=(tam[1], tam[0]),
567             width=r1,
568             height=r2,
569             angle=goc,
570             color=mau,
571             fill=False,
572             lw=5,
573             ls="--",
574             label="Elip (PHS)",
575         )
576         self.ax.add_patch(patch)
577 
578     def ve_canh(self, canh: List[List[List[float]]]) -> None:
579         # Ve mot canh noi giua 2 nut
580         canh_tup = tuple(map(tuple, canh))
581         l = self.ax.plot(
582             [canh[0][1], canh[1][1]],
583             [canh[0][0], canh[1][0]],
584             color="sandybrown",
585             lw=2,
586             marker="x",
587             markersize=4,
588             markerfacecolor="darkcyan",
589             markeredgecolor="darkcyan",
590             label="Nhanh",
591         )
592         self.cac_duong[canh_tup] = l
593 
594     def ve_cay(self, sim: int) -> None:
595         # Ve toan bo cay tim kiem
596         bat_dau = self.bat_dau
597         dich = self.dich
598         self.ve_lai_ban_do(sim)
599         fig = self.fig
600         ax = self.ax
601 
602         for i in range(len(self.tat_ca_canh_moi[sim])):
603             canh_moi = self.tat_ca_canh_moi[sim][i]
604             canh_xoa = self.tat_ca_canh_xoa[sim][i]
605             duong_di = self.tat_ca_duong_di[sim][i]
606             ci = self.tat_ca_ci[sim][i]
607 
608             if canh_xoa:
609                 for canh_x in canh_xoa:
610                     canh_x_tup = tuple(map(tuple, canh_x))
611                     try:
612                         ax.lines.remove(self.cac_duong[canh_x_tup][0])
613                         self.cac_canh.remove(canh_x_tup)
614                     except:
615                         continue
```

```
            if canh_moi is not None:
                canh_moi_tup = tuple(map(tuple, canh_moi))
                self.cac_canh.add(canh_moi_tup)
                self.ve_canh(canh_moi)

            if duong_di is None:
                if self.duong_di_cuoi is not None:
                    self.ve_duong_di_cuoi(self.duong_di_cuoi)
            else:
                self.duong_di_cuoi = duong_di
                self.ve_duong_di_cuoi(duong_di)

            self.ve_ellipse(ci, mau="dimgrey")
            ax.plot(
                bat_dau[1],
                bat_dau[0],
                color="red",
                marker="*",
                markersize=20,
            )
            ax.plot(
                dich[1],
                dich[0],
                color="blue",
                marker="*",
                markersize=20,
            )
            handles, labels = self.ax.get_legend_handles_labels()
            by_label = dict(zip(labels, handles))
            plt.legend(
                by_label.values(),
                by_label.keys(),
                bbox_to_anchor=(1.05, 1.0),
                loc="upper left",
            )
            plt.show(block=False)
            plt.pause(0.0001)

    def ve_nhanh(self, sim: int) -> None:
        # Ve nhanh (chi ve ket qua cuoi cung)
        self.cac_canh = self.tat_ca_danh_sach_canh[sim]
        duong_di = self.tat_ca_duong_di[sim][-1]
        ci = self.tat_ca_ci[sim][0]
        self.ve_lai_ban_do(sim)
        if duong_di is not None:
            self.duong_di_cuoi = duong_di
            self.ve_duong_di_cuoi(duong_di)
        self.ve_ellipse(ci, mau="dimgrey")
        self.ax.plot(
            self.bat_dau[1],
            self.bat_dau[0],
            color="red",
            marker="*",
            markersize=20,
        )
        self.ax.plot(
            self.dich[1],
            self.dich[0],
            color="blue",
            marker="*",
            markersize=20,
```

```
        )
        handles, labels = self.ax.get_legend_handles_labels()
        by_label = dict(zip(labels, handles))
        plt.legend(
            by_label.values(),
            by_label.keys(),
            bbox_to_anchor=(1.05, 1.0),
            loc="upper left",
        )
        plt.savefig(f"{self.thu_muc_ket_qua}/BITStar_Sim_{sim:02d}.png")
        plt.show(block=False)
        plt.pause(1)

    def hien_thi(self, sim: int, nhanh: bool = False) -> None:
        # Hien thi ket qua
        if nhanh:
            self.ve_nhanh(sim)
        else:
            self.ve_cay(sim)

    def ve_lai_ban_do(self, sim: int) -> None:
        # Ve lai ban do va cac thanh phan
        self.ax.clear()
        im = self.ax.imshow(self.ban_do_chan, cmap=plt.cm.gray, extent=[0, 100,
            100, 0])
        for e in self.cac_canh:
            self.ve_canh(e)
        self.ax.plot(
            self.bat_dau[1],
            self.bat_dau[0],
            color="red",
            marker="*",
            markersize=20,
            label="Bat dau",
        )
        self.ax.plot(
            self.dich[1],
            self.dich[0],
            color="blue",
            marker="*",
            markersize=20,
            label="Dich",
        )
        self.ax.set_title(f"BIT* - Lan mo phong {sim}", fontsize=30)
        self.ax.set_xlabel(r"X$\rightarrow$", fontsize=10)
        self.ax.set_ylabel(r"Y$\rightarrow$", fontsize=10)
        handles, labels = self.ax.get_legend_handles_labels()
        by_label = dict(zip(labels, handles))
        plt.legend(
            by_label.values(),
            by_label.keys(),
            bbox_to_anchor=(1.05, 1.0),
            loc="upper left",
        )
        self.ax.set_xlim(-10, 110)
        self.ax.set_ylim(-10, 110)

def main(
    ten_ban_do: str,
    hien_thi: bool,
    bat_dau: list,
    dich: list,
```

```
    rbit: float,
    mau: int,
    chieu: int,
    seed: int,
    thoi_gian_dung: int,
    nhanh: bool,
) -> None:
    # Ham chinh thuc hien chuong trinh
    thu_muc_hien_tai = os.path.abspath(os.path.dirname(__file__))
    tat_ca_thoi_gian = []
    tat_ca_do_dai_duong_di = []
    duong_dan_file = f"{thu_muc_hien_tai}/../KetQua/do_dai_va_thoi_gian_{ten_ban_do}.txt"
    os.makedirs(os.path.dirname(duong_dan_file), exist_ok=True)

    for seed in range(seed):
        random.seed(seed)
        np.random.seed(seed)
        bat_dau = np.array(bat_dau, dtype=float)
        dich = np.array(dich, dtype=float)
        bat_dau = np.array(bat_dau)
        dich = np.array(dich)
        thu_muc_log = f"{thu_muc_hien_tai}/../Log/{ten_ban_do}"
        os.makedirs(thu_muc_log, exist_ok=True)
        duong_dan_ban_do = f"{thu_muc_hien_tai}/../BanDo/{ten_ban_do}.png"
        ban_do_chan = np.array(Image.open(duong_dan_ban_do))
        global start_arr, goal_arr
        start_arr = bat_dau
        goal_arr = dich
        nut_bat_dau = Node(tuple(bat_dau), gt=0)
        nut_dich = Node(tuple(dich))
        ban_do = BanDo(bat_dau=nut_bat_dau, dich=nut_dich, luoi_chan=ban_do_chan)
        bo_ke_hoach = None

        if hien_thi or nhanh:
            bo_ke_hoach = BITStar(
                bat_dau=nut_bat_dau,
                dich=nut_dich,
                ban_do_chan=ban_do,
                so_mau=mau,
                rbit=rbit,
                chieu=chieu,
                thu_muc_log=thu_muc_log,
                thoi_gian_dung=thoi_gian_dung,
            )
        else:
            bo_ke_hoach = BITStar(
                bat_dau=nut_bat_dau,
                dich=nut_dich,
                ban_do_chan=ban_do,
                so_mau=mau,
                rbit=rbit,
                chieu=chieu,
                thoi_gian_dung=thoi_gian_dung,
            )
        duong_di, do_dai_duong_di, thoi_gian = bo_ke_hoach.lap_ke_hoach()

        print(
            f"{MauTerminal.THANH_CONG}seed: {seed}\t\tCI cuoi cung: {bo_ke_hoach.ci}\tCI cu: {bo_ke_hoach.ci_cu}\tDo dai duong di cuoi cung: {do_dai_duong_di}\nDuong di:{MauTerminal.END} {duong_di}\n{
```

```
                MauTerminal.THANH_CONG}Thoi gian thuc hien moi lan:{MauTerminal.
                END} {thoi_gian}\n{MauTerminal.END}"
797         )
798         tat_ca_thoi_gian.append(thoi_gian)
799         tat_ca_do_dai_duong_di.append(do_dai_duong_di)
800         chuoi_thoi_gian = ", ".join([str(t) for t in thoi_gian])
801         with open(duong_dan_file, "a") as f:
802             f.write(
803                 f"seed: {seed}\nDo dai duong di: {do_dai_duong_di}\nThoi gian: {
                    chuoi_thoi_gian}\n"
804             )
805 
806         if hien_thi or nhanh:
807             thu_muc_ket_qua = f"../KetQua/{ten_ban_do} - {datetime.now().
                strftime('%Y-%m-%d_%H-%M-%S')}/"
808             os.makedirs(thu_muc_ket_qua, exist_ok=True)
809             ban_do_dao = np.where((ban_do_chan == 0) | (ban_do_chan == 1),
                ban_do_chan ^ 1, ban_do_chan)
810             bo_hien_thi = TrinhBay(bat_dau, dich, ban_do_dao, thu_muc_ket_qua)
811             print(
812                 f"{MauTerminal.THANH_CONG}{MauTerminal.BOLD}{len(os.listdir(
                    thu_muc_log))} file trong thu muc Log:{MauTerminal.END} {
                    thu_muc_log}/{sorted(os.listdir(thu_muc_log))}\n"
813             )
814             bo_hien_thi.doc_json(thu_muc_log, so_lan_toi_da=np.inf)
815 
816             for i in range(len(os.listdir(thu_muc_log))):
817                 bo_hien_thi.hien_thi(i, nhanh)
818             bo_hien_thi.ax.set_title("BIT* - Duong di cuoi cung", fontsize=30)
819             plt.show()
820 
821         print(
822             f"{MauTerminal.LOI}========== Xoa thu muc log: {thu_muc_log}
                ==========={MauTerminal.END}"
823         )
824         shutil.rmtree(thu_muc_log)
825 
826 def phan_tich_tham_so() -> argparse.Namespace:
827     # Phan tich tham so dong lenh
828     phan_tich = argparse.ArgumentParser()
829     phan_tich.add_argument(
830         "--ten_ban_do",
831         type=str,
832         default="MacDinh",
833         help="Ten file ban do. Ban do phai o thu muc BanDo."
834     )
835     phan_tich.add_argument(
836         "--hien_thi",
837         action="store_true",
838         help="Co luu va hien thi ket qua hay khong",
839     )
840     phan_tich.add_argument("--bat_dau", nargs="+", help="Toa do bat dau. Vi du:
        --bat_dau 0 0")
841     phan_tich.add_argument("--dich", nargs="+", help="Toa do dich. Vi du: --dich
         99 99")
842     phan_tich.add_argument(
843         "--rbit", type=float, default=10, help="Do dai canh toi da. Vi du: --
            rbit 10"
844     )
845     phan_tich.add_argument(
846         "--mau",
847         type=int,
```

```
        default=50,
        help="So mau moi moi lan lap. Vi du: --mau 50",
    )
    phan_tich.add_argument(
        "--chieu", type=int, default=2, help="So chieu cua khong gian lam viec.
            Vi du: --chieu 2"
    )
    phan_tich.add_argument(
        "--seed",
        type=int,
        default=1,
        help="seed ngau nhien. Vi du: --seed 1",
    )
    phan_tich.add_argument(
        "--thoi_gian_dung",
        type=int,
        default=60,
        help="Thoi gian dung thuat toan. Vi du: --thoi_gian_dung 60",
    )
    phan_tich.add_argument(
        "--nhanh",
        action="store_true",
        help="Co chi ve danh sach canh cuoi cung cua moi lan lap hay khong"
    )
    tham_so = phan_tich.parse_args()
    if tham_so.nhanh:
        tham_so.hien_thi = True
    return tham_so

if __name__ == "__main__":
    if len(sys.argv) == 1:
        sys.argv.append("--help")
    tham_so = phan_tich_tham_so()
    print(f"{MauTerminal.YELLOW}{tham_so}{MauTerminal.END}")
    assert len(tham_so.bat_dau) == tham_so.chieu
    assert len(tham_so.dich) == tham_so.chieu
    main(**vars(tham_so))
```

Chương 6

Demo thuật toán và kiểm nghiệm

6.1 Mô phỏng thuật toán Batch Informed Trees (BIT*)

Thuật toán **Batch Informed Trees (BIT*)** là một phương pháp lập kế hoạch đường đi có tính tối ưu tiệm cận, hoạt động dựa trên nguyên lý lấy mẫu theo lô trong không gian trạng thái. Không gian trạng thái được giả định là rời rạc và có thể lấy mẫu ngẫu nhiên để xây dựng một đồ thị gọi là *Đồ thị Hình học Ngẫu nhiên* (Random Geometric Graph - RGG). Trong RGG, các nút đại diện cho trạng thái và các cạnh biểu diễn chuyển tiếp giữa các trạng thái lân cận trong không gian.

A. Quá trình hoạt động của BIT* được mô tả như sau :

1. **Khởi tạo:** Một đồ thị RGG ban đầu được tạo bằng cách lấy mẫu ngẫu nhiên trong không gian tự do, sau đó kết nối các mẫu này với trạng thái bắt đầu và trạng thái đích.

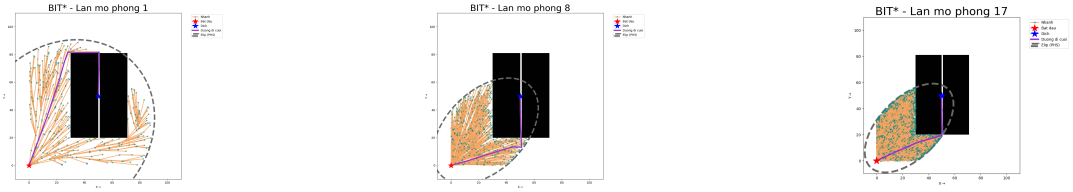
2. **Tìm kiếm lần đầu:** Sử dụng một hàm heuristic, thuật toán xây dựng cây tìm kiếm từ trạng thái bắt đầu. Chỉ các cạnh không va chạm mới được xem xét. Quá trình dừng lại khi tìm được một lời giải đầu tiên khả thi.

3. **Cải thiện lời giải:** Nếu đã có lời giải, BIT* tăng mật độ mẫu bằng cách thêm một lô mẫu mới vào đồ thị, nhưng chỉ trong phạm vi một hình ellipse chứa lời giải tốt hơn. Điều này giúp giới hạn không gian tìm kiếm.

4. **Cắt tỉa cây:** Các nhánh không có khả năng đóng góp vào việc cải thiện lời giải hiện tại sẽ bị loại bỏ. Chỉ giữ lại phần cây liên quan đến lời giải tiềm năng.

5. **Lặp lại:** Với cây đã cắt tỉa, thuật toán tiếp tục thêm mẫu trong hình ellipse và khởi động lại quá trình tìm kiếm. Quá trình này lặp lại mỗi khi tìm được một lời giải tốt hơn.

6. **Kết thúc:** Thuật toán dừng khi không còn lời giải nào tốt hơn, không còn cạnh khả thi hoặc đã hết thời gian cho phép.

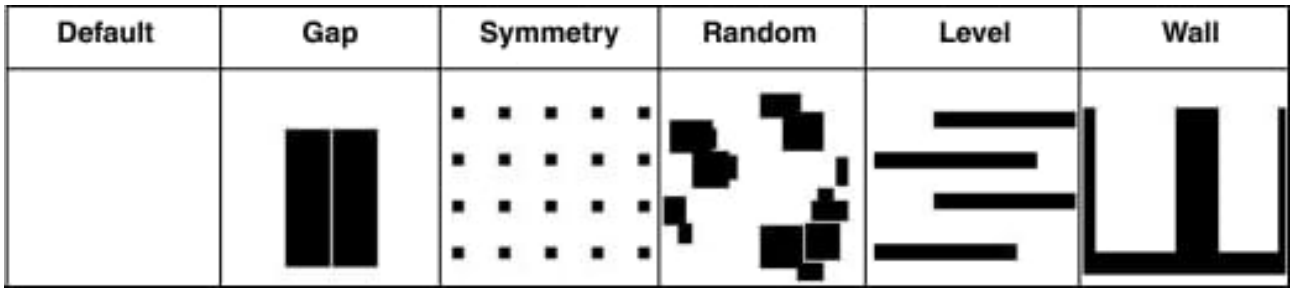


a **b** **c**

Hình 1: **Ví dụ minh họa hoạt động của thuật toán BIT*:** Trạng thái bắt đầu và kết thúc được tô đỏ và xanh tương ứng. Đường đi tốt nhất hiện tại được tô màu tím. (a) Giai đoạn đầu tiên mở rộng heuristic cho đến khi tìm được lời giải. (b) Tìm kiếm tiếp tục sau khi cắt tỉa và tăng độ chính xác với mẫu mới trong một ellipse. (c) Quá trình tiếp tục và tinh chỉnh không gian tìm kiếm cho đến khi lời giải tối ưu toàn cục được tìm thấy hoặc hết thời gian.

B. Dự án

- Mục tiêu của dự án này là hiểu sâu hơn về thuật toán Batch Informed Trees (BIT*) và đánh giá hiệu năng của nó.
- Trong khuôn khổ dự án, chúng em đã lập trình thuật toán BIT* bằng ngôn ngữ Python. Cụ thể, chúng em đã hiện thực Thuật toán (BIT*), Thuật toán (Mở rộng đỉnh kế tiếp – Expand Next Vertex) và Thuật toán (Cắt tỉa – Prune). Ngoài ra, chúng em cũng cài đặt phương pháp lấy mẫu thông minh (informed sampling) sử dụng Thuật toán (Sample) và Thuật toán (SamplePHS) từ tài liệu.
- Chúng em đã tiến hành phân tích hiệu năng của thuật toán BIT* trong không gian hai chiều $\mathbb{R}^2$, với các độ dài đường đi khác nhau, sử dụng 5 seed ngẫu nhiên và 5 kịch bản lập kế hoạch chuyển động phổ biến như mô tả trong.



Hình 2: Minh họa các trường hợp lập kế hoạch chuyển động trong môi trường có chướng ngại vật (màu đen). Các trường hợp bao gồm:(Default), (Gap), (Symmetry), (Random), (Level), (Wall).

6.2 Cài đặt thực nghiệm

Vấn đề lập kế hoạch chuyển động được biết là gặp nhiều khó khăn theo nhiều cách khác nhau. Để cô lập các thách thức cụ thể trong lập kế hoạch chuyển động, chúng tôi đã xây dựng sáu trường hợp lập kế hoạch.

Các trường hợp này được thiết kế nhằm giúp chúng em hiểu rõ hiệu suất của thuật toán trong nhiều môi trường khác nhau:

1. **Default:** Đây là một trường hợp đơn giản không có chướng ngại vật. Mục tiêu là đánh giá hiệu suất cơ bản của thuật toán trong môi trường đơn giản nhất.

2. **Gap:** Trường hợp này chứa một khe hẹp, giúp đánh giá khả năng lấy mẫu của thuật toán trong các không gian hẹp và cách heuristic ảnh hưởng đến quá trình tìm lời giải tối ưu đi qua khe này.

3. **Level:** Đây là môi trường có đường đi vòng vèo. Mục tiêu là kiểm tra cách thuật toán xử lý các thay đổi luân phiên về chi phí, từ đó đánh giá khả năng phân nhánh và thời gian tìm đến lời giải tối ưu.

4. **Wall:** Trường hợp này có điểm bắt đầu và đích bị bao quanh bởi chướng ngại vật. Nó kiểm tra khả năng của thuật toán trong việc vượt qua các heuristic Euclidean sai lệch và tìm đường đi không va chạm tối ưu.

5. **Symmetry:** Trường hợp này có các chướng ngại vật lặp lại đều đặn, giúp đánh giá hành vi của thuật toán trong môi trường có tính kiểm soát cao. Nhờ các chướng ngại vật lặp lại, thuật toán phải đối mặt với cùng một thử thách nhiều lần, từ đó dễ dàng phát hiện ra các vấn đề tiềm ẩn.

6. **Random:** Trường hợp này có các chướng ngại vật được đặt ngẫu nhiên nhằm kiểm tra độ ổn định và khả năng thích nghi của thuật toán trong các môi trường không xác định.

Các môi trường này có kích thước 100 pixel x 100 pixel.

6.3 Phân tích kết quả

Thuật toán Batch Informed Trees (BIT*) đã được đánh giá trên các trường hợp lập kế hoạch khác nhau để phân tích hiệu suất cả định lượng và định tính.

Tất cả các thí nghiệm được thực hiện trên toàn bộ các trường hợp sau đó thời gian thực hiện và độ dài đường đi trung vị được phân tích. Kết quả đánh giá được trình bày trong bảng.

A. (Default)

Trong trường hợp này, không có chướng ngại vật trong môi trường. Vị trí bắt đầu được đặt tại $[0, 0]$ và đích tại $[99, 99]$, tức là hai điểm nằm ở góc đối diện trên lưới 100x100. Do đó, đường đi ngắn nhất là một đường chéo nối từ điểm bắt đầu đến đích. Khi giới hạn thời gian lập kế hoạch là 60 giây, BIT* đã tạo ra một đường đi với độ dài trung vị là 140.0072 đơn vị.

B. (Gap)

Trường hợp này được thiết kế để kiểm tra khả năng lấy mẫu trong không gian hẹp của thuật toán. Vị trí bắt đầu đặt tại $[0, 0]$, và đích tại $[50, 50]$. Môi trường chứa một khe hẹp chỉ 1 pixel để đường đi có thể xuyên qua. Mục tiêu là kiểm tra cách thuật toán xử lý các không gian bị giới hạn.

BIT* đã tìm được đường đi với độ dài trung vị là 84.5480 đơn vị sau khi giới hạn thời gian lập kế hoạch là 60 giây.

C.(Level)

Trường hợp này cho phép đánh giá hiệu quả của thuật toán trong môi trường có sự thay đổi chi phí luân phiên. Điểm bắt đầu là $[0, 0]$ và đích là $[99, 99]$.

Với thời gian lập kế hoạch giới hạn 60 giây, BIT* đạt được đường đi có độ dài trung vị là 239.2301

D. (Wall)

Trường hợp này mô tả tình huống trong đó cả điểm bắt đầu và điểm đích đều bị bao quanh bởi các chướng ngại vật. Mục đích là phân tích hành vi của thuật toán khi đường đi bị ảnh hưởng bởi hàm heuristic, buộc thuật toán phải di chuyển lệch hướng so với vị trí đích để tránh va chạm với chướng ngại vật. Vị trí bắt đầu là $[50, 25]$ và vị trí đích là $[50, 70]$.

Sau khi giới hạn thời gian lập kế hoạch là 60 giây, BIT* đã tạo ra đường đi với độ dài trung vị là 107.2184 đơn vị.

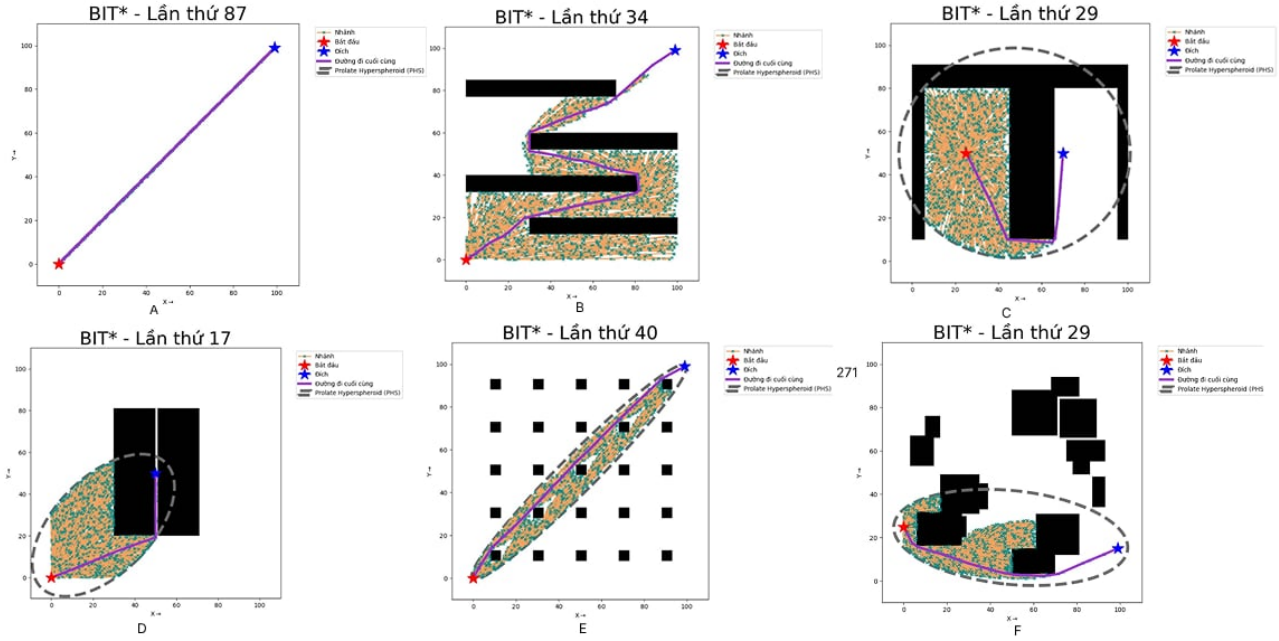
E.(Symmetry)

Trường hợp này bao gồm các chướng ngại vật được phân bố đều khắp môi trường, là một bài kiểm tra hữu ích cho khả năng cơ bản của thuật toán lập kế hoạch chuyển động. Vị trí bắt đầu được đặt tại $[0, 0]$ và điểm đích tại $[99, 99]$. Sau khi giới hạn thời gian lập kế hoạch là 60 giây, BIT* đạt được đường đi với độ dài trung vị là 141.0208 đơn vị.

F. (Random)

Trong trường hợp này, các chướng ngại vật được đặt ngẫu nhiên để kiểm tra độ bền vững của thuật toán. Điểm bắt đầu là $[25, 0]$ và điểm đích là $[15, 99]$. Do các chướng ngại vật được tạo ngẫu nhiên, không có chiều dài đường đi tối ưu chuẩn để so sánh.

Thuật toán BIT* đã đưa ra một đường đi có độ dài trung vị là 108.5151 đơn vị.



Hình 3: Minh họa về đường dẫn cuối cùng được tính toán bằng BIT* trong tất cả các trường hợp lập kế hoạch của chúng tôi. (a) Default, (b) Level, (c) Wall, (d) Gap, (e) Symmetric, and (f) Random.

Bảng 6.1: Thời gian và độ dài đường đi khi chạy thuật toán BIT* qua các trường hợp khác nhau

Trường hợp	**Thời gian (s)**	**Độ dài đường đi**
(Default)	52.62	140.0072
(Level)	56.0738	239.2301
(Wall)	62.2360	107.2184
(Gap)	55.58	84.5480
(Symmetry)	56.02	141.0208
(Random)	52.99	108.5151

Chương 7

Kết luận

Trong báo cáo này, thuật toán Batch Informed Trees (BIT*) đã được nghiên cứu, triển khai và đánh giá hiệu suất thông qua nhiều trường hợp lập kế hoạch đường đi với mức độ phức tạp khác nhau. Các thử nghiệm bao gồm các trường hợp không có chướng ngại vật, có khe hẹp, mê cung, bao vây, đối xứng và môi trường ngẫu nhiên. Kết quả cho thấy BIT* có khả năng tìm được đường đi gần tối ưu trong thời gian hợp lý và hội tụ đến lời giải tối ưu khi được phép chạy đủ lâu.

BIT* tỏ ra vượt trội hơn so với các thuật toán truyền thống như RRT* về thời gian hội tụ và hiệu quả tìm kiếm trong không gian phức tạp. Thuật toán tận dụng tốt thông tin heuristic để dẫn hướng việc lấy mẫu, từ đó tối ưu hóa hiệu suất và độ chính xác của lời giải. Dù không phải là thuật toán nhanh nhất trong mọi trường hợp, BIT* thể hiện rõ sự cân bằng giữa tốc độ và độ tối ưu của đường đi.

Ngoài ra, việc áp dụng BIT* vào các trường hợp thực tế mở ra tiềm năng ứng dụng rộng rãi trong lĩnh vực điều hướng robot, phương tiện tự hành và các hệ thống yêu cầu lập kế hoạch đường đi hiệu quả trong không gian nhiều chiều.

Tài liệu tham khảo

1. J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.

2. E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

3. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

4. S. M. LaValle et al., “Rapidly-exploring random trees: A new tool for path planning,” 1998.

5. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

6. J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, “Informed sampling for asymptotically optimal path planning,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

7. J. D. Gammell, M. P. Strub, and V. N. Hartmann, “Planner developer tools (pdt): Reproducible experiments and statistical analysis for developing and testing motion planners,” in *Proceedings of the Workshop on Evaluating Motion Planning Performance (EMPP), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

8. S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

9. J. J. Kuffner and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.