**Lập kế hoạch đường đi sử dụng tìm kiếm Neural A\***

Tác giả: Ryo Yonetani \* 1 Tatsunori Taniai \* 1 Mohammadamin Barekatain 1 2 Mai Nishimura 1 Asako Kanezaki 3

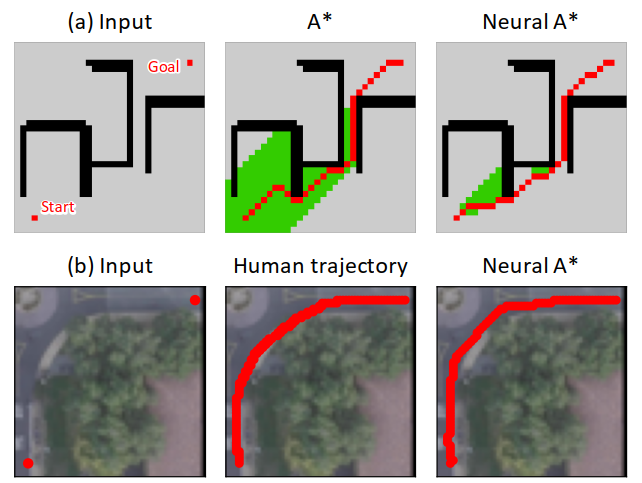
**Tóm tắt**

Chúng tôi giới thiệu Neural A\*, một phương pháp tìm kiếm mới dựa trên dữ liệu cho bài toán lập kế hoạch đường đi. Mặc dù lĩnh vực lập kế hoạch đường đi dựa trên dữ liệu gần đây nhận được nhiều sự quan tâm, nhưng việc áp dụng học máy vào lập kế hoạch dựa trên tìm kiếm vẫn còn nhiều thách thức do bản chất rời rạc của các thuật toán tìm kiếm. Trong nghiên cứu này, chúng tôi tái cấu trúc thuật toán A\* cổ điển thành một phiên bản khả vi và kết hợp nó với một bộ mã hóa tích chập để tạo thành một mạng nơ-ron lập kế hoạch có thể huấn luyện toàn bộ đầu-cuối.

Neural A\* giải quyết bài toán lập kế hoạch đường đi bằng cách mã hóa một trường hợp cụ thể thành một bản đồ hướng dẫn, sau đó thực hiện tìm kiếm A\* khả vi với bản đồ này. Bằng cách học cách khớp kết quả tìm kiếm với các đường đi chuẩn do chuyên gia cung cấp, Neural A\* có thể tạo ra đường đi chính xác và hiệu quả, phù hợp với dữ liệu gốc. Các thí nghiệm chuyên sâu của chúng tôi xác nhận rằng Neural A\* vượt trội hơn các bộ lập kế hoạch dựa trên dữ liệu hiện đại về mặt cân bằng giữa tối ưu hóa tìm kiếm và hiệu suất. Hơn nữa, Neural A\* còn dự đoán thành công các quỹ đạo di chuyển thực tế của con người bằng cách thực hiện trực tiếp việc lập kế hoạch dựa trên tìm kiếm từ đầu vào là hình ảnh tự nhiên.

**1. Giới thiệu**

Lập kế hoạch đường đi là bài toán tìm một lộ trình hợp lệ và chi phí thấp từ điểm bắt đầu đến điểm đích trong bản đồ môi trường. Lập kế hoạch dựa trên tìm kiếm, bao gồm thuật toán tìm kiếm A\* nổi tiếng (Hart et al., 1968), là một phương pháp phổ biến trong việc giải quyết các bài toán lập kế hoạch đường đi và đã được ứng dụng rộng rãi trong nhiều lĩnh vực như điều hướng phương tiện tự hành (Paden et al., 2016), điều khiển cánh tay robot (Smith et al., 2012) và trí tuệ nhân tạo trong trò chơi (Abd Algfoor et al., 2015). So với các phương pháp khác như lập kế hoạch dựa trên lấy mẫu (Gonzalez et al., 2015) hay lập kế hoạch phản xạ (Tamar et al., 2016; Lee et al., 2018), lập kế hoạch dựa trên tìm kiếm đảm bảo sẽ tìm được đường đi nếu tồn tại, bằng cách khám phá bản đồ một cách tuần tự và toàn diện.



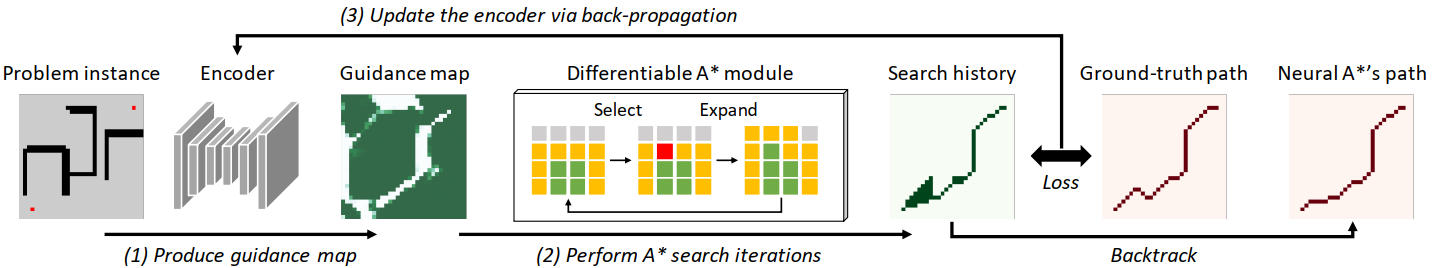
**Hình 1. Hai kịch bản lập kế hoạch đường đi với Neural A\***

(a) Tìm đường ngắn nhất từ điểm này đến điểm kia: tìm một đường đi gần tối ưu (màu đỏ) với số lượng nút được khám phá ít hơn (màu xanh lá) trên một bản đồ đầu vào.

(b) Lập kế hoạch đường đi từ hình ảnh thô: dự đoán chính xác quỹ đạo di chuyển của con người (màu đỏ) trên một hình ảnh tự nhiên.

Học cách lập kế hoạch từ các ví dụ do chuyên gia cung cấp đang nhận được sự quan tâm như là một hướng mở rộng đầy hứa hẹn cho các bộ lập kế hoạch cổ điển. Các nghiên cứu gần đây đã chỉ ra những ưu điểm lớn của việc lập kế hoạch đường đi dựa trên dữ liệu trong hai trường hợp sau: (1) tìm được đường đi gần tối ưu hiệu quả hơn so với các bộ lập kế hoạch cổ điển trong bài toán tìm đường ngắn nhất giữa hai điểm (Choudhury et al., 2018; Qureshi et al., 2019; Takahashi et al., 2019; Chen et al., 2020; Ichter et al., 2020), và  
(2) cho phép lập kế hoạch đường đi từ dữ liệu hình ảnh thô (Tamar et al., 2016; Lee et al., 2018; Ichter & Pavone, 2019; Vlastelica et al., 2020), điều vốn khó khăn đối với các bộ lập kế hoạch truyền thống nếu không có nhãn ngữ nghĩa chi tiết theo từng điểm ảnh của môi trường.

Trong nghiên cứu này, chúng tôi giải quyết cả hai trường hợp nêu trên một cách có hệ thống, như được minh họa trong Hình 1. Khác với phần lớn các phương pháp dựa trên dữ liệu hiện nay mở rộng từ lập kế hoạch lấy mẫu hoặc phản xạ, chúng tôi theo đuổi hướng lập kế hoạch dựa trên tìm kiếm với ưu điểm nội tại là đảm bảo thành công trong việc tìm đường đi.



**Hình 2. Sơ đồ khái quát của Neural A\***

(1) Một trường hợp bài toán lập kế hoạch đường đi được đưa vào bộ mã hóa (encoder) để tạo ra bản đồ hướng dẫn. (2) Mô-đun A\* khả vi thực hiện tìm kiếm đường đi ngắn nhất từ điểm này đến điểm kia dựa trên bản đồ hướng dẫn và tạo ra lịch sử tìm kiếm cùng với đường đi kết quả. (3) Hàm mất mát giữa lịch sử tìm kiếm và đường đi chuẩn (ground-truth) được lan truyền ngược để huấn luyện bộ mã hóa.

Các nghiên cứu theo hướng này cho đến nay còn bị hạn chế do gặp khó khăn phát sinh từ tính rời rạc của các bước tìm kiếm trong lập kế hoạch dựa trên tìm kiếm, khiến việc huấn luyện bằng phương pháp lan truyền ngược (backpropagation) trở nên phức tạp. Một số phương pháp hiện tại huấn luyện các hàm chi phí heuristic tại từng điểm lưới một cách độc lập, điều này đòi hỏi các chú thích chuyên gia rất chi tiết, chẳng hạn như các bộ lập kế hoạch chuẩn (oracle) chạy trực tiếp trong quá trình huấn luyện (Choudhury et al., 2018) hoặc việc tính toán trước một cách toàn diện hàm heuristic tối ưu cho một bộ lập kế hoạch (Takahashi et al., 2019). Tuy nhiên, những loại chú thích chi tiết như vậy không phải lúc nào cũng sẵn có hay có khả năng mở rộng, đặc biệt là khi liên quan đến các quy trình gán nhãn thủ công tốn nhiều công sức như trong (Kim & Pineau, 2016; Kretzschmar et al., 2016; Perez-Higueras et al., 2018).

Gần đây hơn, Vlastelica et al. (2020) đã áp dụng phương pháp tối ưu hóa "hộp đen" vào các bộ giải tổ hợp tích hợp trong mạng nơ-ron, từ đó cho phép huấn luyện đầu-cuối thông qua các thuật toán tổ hợp, bao gồm cả lập kế hoạch dựa trên tìm kiếm. Phương pháp tối ưu hóa đa mục đích này có thể được điều chỉnh để áp dụng cho bài toán của chúng tôi. Tuy nhiên, việc coi toàn bộ quá trình tìm kiếm là một hàm "hộp đen" khiến việc theo dõi chi tiết các bước tìm kiếm bên trong trở nên khó khăn, từ đó làm cho quá trình huấn luyện trở nên phức tạp.

Để giải quyết vấn đề không khả vi của lập kế hoạch đường đi dựa trên tìm kiếm, chúng tôi đề xuất một bộ lập kế hoạch mới dựa trên dữ liệu có tên là **Neural A\***. Cốt lõi của phương pháp *này là cải biến thuật toán A\* cổ điển thành một phiên bản* ***khả vi***, gọi là **A\* khả vi (differentiable A\*)**, bằng cách kết hợp kỹ thuật kích hoạt rời rạc lấy cảm hứng từ Hubara et al. (2016) với các phép toán ma trận cơ bản. Mô-đun này cho phép thực hiện tìm kiếm A\* trong bước lan truyền thuận của mạng nơ-ron và lan truyền ngược hàm mất mát qua từng bước tìm kiếm đến các mô-đun nền có thể huấn luyện được khác. Như minh họa trong Hình 2, **Neural A\*** bao gồm sự kết hợp giữa một **bộ mã hóa tích chập hoàn toàn (fully-convolutional encoder)** và mô-đun **A\* khả vi**, và được huấn luyện như sau: (1) Với một trường hợp bài toán (ví dụ: bản đồ môi trường được đánh dấu điểm bắt đầu và điểm đích), bộ mã hóa sẽ biến đổi đầu vào thành một bản đồ số vô hướng, gọi là **bản đồ hướng dẫn (guidance map)**; (2) Mô-đun A\* khả vi sau đó sẽ thực hiện quá trình tìm kiếm với bản đồ hướng dẫn này để tạo ra **lịch sử tìm kiếm** và **đường đi kết quả**; (3) Lịch sử tìm kiếm được so sánh với đường đi chuẩn (ground-truth) của trường hợp đầu vào để tạo ra **hàm mất mát**, và hàm này được lan truyền ngược để huấn luyện bộ mã hóa.

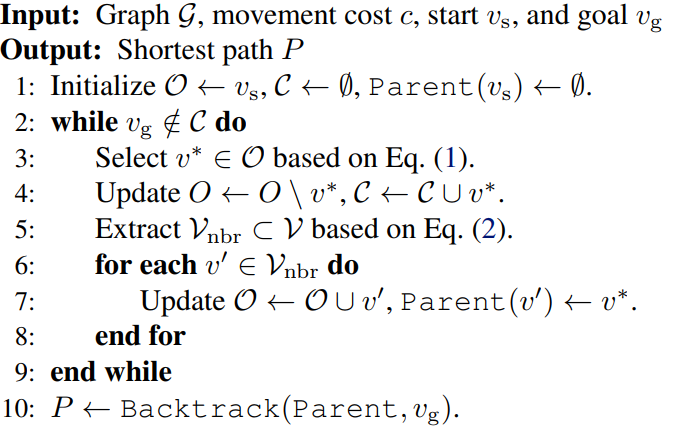
**Vai trò của A\* khả vi trong quá trình huấn luyện rất rõ ràng**: dạy cho bộ mã hóa tạo ra các bản đồ hướng dẫn sao cho làm giảm sự sai lệch giữa lịch sử tìm kiếm kết quả và đường đi chuẩn. Qua đó, bộ mã hóa học cách nắm bắt các tín hiệu hình ảnh trong đầu vào một cách hiệu quả để tái tạo lại các đường đi đúng với dữ liệu gốc. Nguyên lý học tập này cung cấp một giải pháp thống nhất cho cả hai kịch bản được nêu trước đó. Cụ thể, trong kịch bản **tìm đường ngắn nhất** (Hình 1a), nơi đường đi chuẩn được cung cấp bởi các bộ lập kế hoạch tối ưu, bộ mã hóa được huấn luyện để tìm ra đường đi gần tối ưu một cách hiệu quả bằng cách khai thác các tín hiệu hình ảnh như hình dạng của các ngõ cụt. Tại đây, bản đồ hướng dẫn được dùng để bổ trợ bản đồ đầu vào, giúp ưu tiên những nút cần khám phá hoặc cần tránh nhằm cải thiện sự cân bằng giữa tính tối ưu và hiệu quả của tìm kiếm. Ngược lại, khi **đường đi chuẩn được cung cấp bởi người gán nhãn thủ công** từ hình ảnh thô (Hình 1b), bộ mã hóa học được cách lập kế hoạch trực tiếp trên hình ảnh bằng cách phân biệt các vùng có thể đi qua và không thể đi qua, thông qua màu sắc và kết cấu — tương ứng với các nút có chi phí thấp và cao trên bản đồ hướng dẫn.

Chúng tôi đã tiến hành đánh giá rộng rãi phương pháp của mình trên cả các bộ dữ liệu tổng hợp (Bhardwaj et al., 2017) và bộ dữ liệu thực tế (Sturtevant, 2012; Robicquet et al., 2016). Kết quả cho thấy **Neural A\*** vượt trội hơn các bộ lập kế hoạch dựa trên tìm kiếm hiện đại khác (Choudhury et al., 2018; Vlastelica et al., 2020) về mặt cân bằng giữa độ tối ưu của tìm kiếm và hiệu suất trong các bài toán tìm đường ngắn nhất giữa hai điểm. Thêm vào đó, chúng tôi cũng chứng minh rằng **Neural A\*** có thể học cách dự đoán chính xác hơn các quỹ đạo người đi bộ từ hình ảnh giám sát thực tế so với phương pháp của Vlastelica et al. (2020) và các mô hình học theo chuyên gia khác (Ratliff et al., 2006; Tamar et al., 2016; Lee et al., 2018).

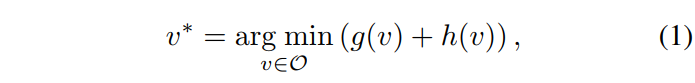
**2. Kiến thức cơ bản**

**Bài toán lập kế hoạch đường đi.** Xét một bài toán lập kế hoạch đường đi trên đồ thị G=(V,E) trong đó V là tập hợp các nút đại diện cho các vị trí trong môi trường và E là tập hợp các chuyển động hợp lệ tiềm năng giữa các nút. Với mỗi v ∈ V, ta định nghĩa N(v)={v′ ∣ (v,v′) ∈ E, v≠v′} là tập hợp các nút lân cận của v. Mỗi cạnh (v,v′) được gán một chi phí không âm c(v,v′) ∈ R+​, biểu thị chi phí di chuyển từ nút v đến nút v′. Mục tiêu của bài toán lập kế hoạch đường đi là tìm một chuỗi các nút được kết nối, P=(v1,v2,…,vT) ∈ VT, với v1=vs​, vT=vg​, sao cho tổng chi phí ∑t=1T−1c(vt,vt+1) là nhỏ nhất. Theo Choudhury và các cộng sự (2018), nghiên cứu này tập trung vào một thiết lập phổ biến, trong đó đồ thị G là thế giới lưới 8-lân cận và chi phí c(v,v′) là một hằng số, ví dụ, c(v,v′)=1 khi v′ có thể đi qua được và c(v,v′)=∞ khi v′ bị chặn bởi chướng ngại vật, chẳng hạn.

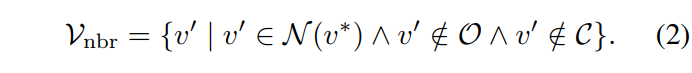
Thuật toán 1: Tìm kiếm A\*



*Tìm kiếm A.*\* Thuật toán 1 trình bày tổng quan về việc triển khai tìm kiếm A\* được sử dụng trong nghiên cứu này. Thuật toán này khám phá các nút để tìm đường đi ngắn nhất P bằng cách lặp lại giữa (1) việc chọn nút có chi phí thấp nhất từ danh sách ứng viên và (2) mở rộng danh sách ứng viên cho nút được chọn, cho đến khi nút đích vg được tìm thấy. Cụ thể hơn, việc chọn nút được thực hiện dựa trên tiêu chí sau:



trong đó O ⊂ V là danh sách mở quản lý các nút ứng viên để khám phá. g(v) biểu thị chi phí thực tế tích lũy c(v′) của các nút v′ trên đường đi hiện tại từ vs​ đến v, được cập nhật liên tục trong quá trình tìm kiếm. Mặt khác, h(v) là một hàm heuristic ước lượng chi phí tổng từ v đến vg​, thường sử dụng khoảng cách đường thẳng giữa v và vg​ trong thế giới lưới. Tất cả các nút đã được chọn đều được lưu trữ trong danh sách đóng C ⊂ V, như được thực hiện ở Dòng 4. Ở Dòng 5 của Thuật toán 1, chúng tôi mở rộng các nút lân cận của v∗ là Vnbr ​⊂ V dựa trên tiêu chí sau:



Các nút lân cận *Vnbr* sau đó được thêm vào danh sách mở *O* ở Dòng 7 để đề xuất các ứng viên mới cho việc lựa chọn trong vòng lặp tiếp theo. Quá trình tìm kiếm sẽ kết thúc khi nút đích *vg* được chọn ở Dòng 3 và được lưu vào danh sách đóng *C*, sau đó thực hiện hàm *Backtrack* để truy ngược các nút cha *Parent(v)* từ *vg* đến *vs*, từ đó thu được đường đi kết quả *P*.

**Thiết lập lập kế hoạch dựa trên dữ liệu.** Như đã giải thích ở Mục 1, chúng tôi tìm kiếm một phương pháp lập kế hoạch dựa trên tìm kiếm có hệ thống cho hai kịch bản khác biệt trong các nghiên cứu hiện có về lập kế hoạch đường đi dựa trên dữ liệu: (1) tìm các đường đi gần tối ưu một cách hiệu quả cho bài toán tìm đường ngắn nhất giữa hai điểm (Choudhury và các cộng sự, 2018), và (2) hỗ trợ lập kế hoạch đường đi từ hình ảnh thô khi không có thông tin về chi phí di chuyển (Vlastelica và các cộng sự, 2020).

Để đạt được mục tiêu này, chúng tôi trừu tượng hóa hai thiết lập trên bằng cách giới thiệu một biến bản đồ môi trường 2D *X*, đại diện cho đồ thị đầu vào *G*. Cụ thể, tương ứng với hai thiết lập có chi phí di chuyển *c(v’)* đã biết và chưa biết, bản đồ *X* biểu thị *c(v’)* theo một trong hai cách: (1) một cách rõ ràng dưới dạng bản đồ nhị phân *X* thuộc tập {0, 1}^(*V*), nhận giá trị 1 tại các vị trí có thể đi qua (tức là *c(v) = 1*) và 0 nếu không thể đi qua, hoặc (2) một cách ngầm định dưới dạng hình ảnh màu thô *X* thuộc tập [0, 1]3×*V*. Kết quả là, mỗi trường hợp bài toán được biểu diễn đồng nhất dưới dạng một bộ giá trị có chỉ số *Q(i) = (X(i), vs(i), vg(i))*.

Với mỗi trường hợp bài toán *Q(i)*, chúng tôi giả định thêm rằng một đường đi chuẩn *P(i)* thuộc tập {0, 1}*V* được cung cấp dưới dạng bản đồ nhị phân 2D, trong đó các phần tử nhận giá trị 1 dọc theo đường đi mong muốn. Khi *X(i)* là một bản đồ nhị phân biểu thị chi phí di chuyển một cách rõ ràng, *P (i)* được tạo ra bằng cách giải bài toán tìm đường ngắn nhất với một bộ lập kế hoạch tối ưu. Nếu *X(i)* là một hình ảnh thô, chúng tôi giả định rằng *P(i)* được cung cấp bởi người gán nhãn thủ công.

**3. Tìm kiếm Neural A**\*

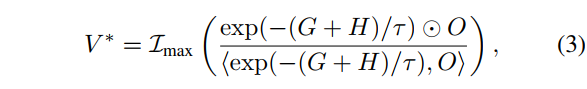
Bây giờ, chúng tôi giới thiệu phương pháp tìm kiếm Neural A\* được đề xuất. Ở cấp độ tổng quan, Neural A\* sử dụng một bộ mã hóa để biến đổi một trường hợp bài toán *Q(i)* thành một bản đồ hướng dẫn, như được minh họa trong Hình 2. Bản đồ hướng dẫn này áp đặt một chi phí hướng dẫn *phi(i)(v)* thuộc tập *R+* cho mỗi nút *v*. Sau đó, mô-đun A\* khả vi, được trình bày chi tiết ở Mục 3.1, thực hiện quá trình tìm kiếm theo chính sách nhằm giảm thiểu tổng chi phí hướng dẫn, tức là tổng từ *t* = 1 đến *T* - 1 của *phi(i)(vt+1)*. Bằng cách lặp lại bước lan truyền thuận này và lan truyền ngược qua quy trình huấn luyện được mô tả ở Mục 3.2, bộ mã hóa học cách nắm bắt các tín hiệu hình ảnh trong các trường hợp huấn luyện để cải thiện độ chính xác của đường đi và hiệu quả tìm kiếm.

**3.1. Mô-đun A\* khả vi**

**Biểu diễn biến số.** Để tái cấu trúc thuật toán tìm kiếm A\* theo dạng khả vi, chúng tôi biểu diễn các biến trong Thuật toán 1 dưới dạng ma trận có kích thước của bản đồ, sao cho mỗi dòng có thể được thực thi thông qua các phép toán ma trận. Cụ thể, gọi *O*, *C*, *Vnbr* thuộc tập {0, 1}*V* là các ma trận nhị phân biểu thị các nút có trong *O*, *C*, *Vnbr*, tương ứng (ở đây, chúng tôi bỏ qua chỉ số *i* để đơn giản hóa).

Chúng tôi biểu diễn nút bắt đầu *vs*, nút đích *vg* và nút được chọn *v\** dưới dạng các ma trận chỉ thị một-nóng (*one-hot*) *Vs*, *Vg*, *V\** thuộc tập {0, 1}*V*, tương ứng, trong đó tích vô hướng của *Vs* với *1*, *Vg* với *1*, và *V\** với *1* đều bằng 1 (tức là ma trận một-nóng), với *1* là ma trận toàn giá trị 1 có kích thước của bản đồ. Ngoài ra, *G*, *H*, *Phi* thuộc tập R+*V* là phiên bản ma trận của *g(v)*, *h(v)* và *phi(v)*, tương ứng.

**Lựa chọn nút.** Việc thực hiện phương trình (1) dưới dạng khả vi là không đơn giản vì nó liên quan đến một thao tác rời rạc. Ở đây, chúng tôi tận dụng một phép kích hoạt rời rạc được cải tiến dựa trên nghiên cứu của Hubara và các cộng sự (2016), và diễn đạt lại phương trình như sau:

**

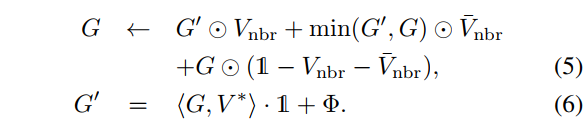
trong đó *A* o *B* biểu thị phép nhân từng phần tử (*element-wise product*), và *t* là một tham số điều chỉnh nhiệt độ (*temperature parameter*) sẽ được định nghĩa cụ thể. Imax(A) là một ma trận một-nóng trong bước lan truyền thuận, trong khi hoạt động như một hàm lựa chọn mềm (*soft-selection function*) cho việc lan truyền ngược. Danh sách mở *O* được sử dụng ở đây để che (*mask*) các nút trong danh sách mở hiện tại, nhưng việc này được thực hiện từ các nút trong danh sách mở gần đây.

**Mở rộng nút lân cận.** Việc mở rộng các nút lân cận của *v\** trong phương trình (2) bao gồm nhiều điều kiện, tức là, *v'* thuộc *N(v\**) và *v'* không thuộc *O* cũng không thuộc *C*. Theo nghiên cứu của Tamar và các cộng sự (2016), chúng tôi triển khai *N* dưới dạng một phép tích chập 2D giữa *V\** và một nhân tích chập cố định *K* = [[1, 1, 1]T; [1, 0, 1]T; [1, 1, 1]T]. Khi *X* được cung cấp dưới dạng bản đồ chi phí di chuyển không thể vượt qua, *Vnbr* được tính bằng các phép toán ma trận sau:

**

trong đó *A* o *B* là phép tích chập 2D của *A* và *B*. Phép tích chập với *X* o (1 - *O*) o (1 - *C*) đóng vai trò như một mặt nạ để giữ lại các nút không nằm trong danh sách mở và danh sách đóng. Việc che bằng *X* giúp bảo tồn các cấu trúc chướng ngại vật và do đó đảm bảo tính chất của thuật toán A\* hoàn chỉnh (*complete*), tức là luôn tìm được một giải pháp nếu một giải pháp tồn tại trong đồ thị *G*, chẳng hạn như trong trường hợp chuẩn *A*. Chúng tôi cũng giới thiệu *Vnbr* = (*V\** o *K*) o *X* o (1 - *O*) o (1 - *C*) và *Vnbr* = (*V\** o *K*) o (1 - *O*) o (1 - *C*) để sử dụng trong việc cập nhật *G* dưới đây. Khi *X* là một hình ảnh thô không biểu thị rõ ràng các nút có thể đi qua, chúng tôi sử dụng *Vnbr* = (*V\** \* *K*) o (1 - *O*) o (1 - *C*) và *Vnbr* = (*V\** \* *K*) o (1 - *O*) o (1 - *C*) để thay thế.

**Cập nhật *G*.** Như đã giải thích trước đó, *g(v)* biểu thị chi phí đường đi thực tế từ *vs* đến *v*, độc lập với chi phí di chuyển thực tế *c(v’)* là không luôn khả dụng. Để cập nhật chi phí này tại mỗi vòng lặp, chúng tôi cập nhật một phần *G* với chi phí mới *G'* như sau:

**

Trong phương trình (5), các nút lân cận *Vnbr* đã mở trước đó và các nút lân cận *Vnbr* hiện tại đã được gán giá trị chi phí thấp hơn. Các chi phí mới *G'* và *G* trước đó được tính toán bằng phương trình (6) dưới dạng tổng của chi phí hiện tại của nút được chọn, *g(v*\**)*, áp dụng *G'* và *G* và chi phí hướng dẫn *phi* ở bước tiếp theo để biểu diễn các nút lân cận theo *Phi*.

**3.2. Huấn luyện Neural A**\*

**Thiết kế tổn thất.** Mô-đun A\* khả vi kết nối đầu ra với bản đồ hướng dẫn *Phi* và do đó với bộ mã hóa. Ở đây, đầu ra là danh sách đóng *C*, là một ma trận nhị phân biểu thị các nút *V\** đã được tìm kiếm trong lịch sử tìm kiếm (xem Eq 8 để biết chi tiết). Chúng tôi đánh giá tổn thất *L1* giữa *C* và bản đồ đường đi chuẩn *P* như sau:

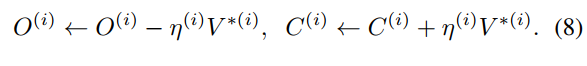
**

Tổn thất này giám sát việc lựa chọn nút bằng cách phạt cả (1) các lựa chọn âm tính giả (*false-negative selections*), tức là các nút nên được đưa vào *P* nhưng không có trong *C*, và (2) các lựa chọn dương tính giả (*false-positive selections*) của các nút có trong *C* nhưng không thuộc *P*. Ngoài ra, tổn thất này khuyến khích Neural A\* (1) tìm kiếm một đường đi gần giống với đường đi chuẩn *P* với ít lần mở rộng nút hơn.

Trong thực tế, chúng tôi làm mất ổn định các mức độ của *O* trong phương trình (3), *Vnbr* trong phương trình (5), và *G* và *G'* trong phương trình (6) bằng cách ngắt các chuỗi lan truyền ngược từ *Phi*, điều này giúp cải thiện đáng kể hiệu quả hướng dẫn *Phi* trong phương trình (6) cho việc lựa chọn nút, trong khi vẫn đơn giản hóa việc lan truyền ngược lớn để ổn định huấn luyện và giảm tiêu thụ bộ nhớ.

**Thiết kế bộ mã hóa.** Tổn thất được hiển thị ở trên được lan truyền ngược qua mỗi bước tìm kiếm trong mô-đun A\* khả vi đến bộ mã hóa. Ở đây, chúng tôi mong đợi bộ mã hóa học được các tín hiệu hình ảnh từ các trường hợp bài toán cho việc lập kế hoạch dựa trên tìm kiếm chính xác và hiệu quả. Các tín hiệu bao gồm, ví dụ, hình dạng của các ngõ cụt và các điểm cuối trong các bản đồ chi phí nhị phân hoặc các mẫu màu sắc và kết cấu trong các hình ảnh thô có thể đi qua. Với mục đích này, chúng tôi sử dụng kiến ​​trúc mạng tích chập hoàn toàn như U-Net (Ronneberger và cộng sự, 2015) được sử dụng để phân đoạn ngữ nghĩa, mạng này có thể học các biểu diễn hình ảnh cục bộ ở độ phân giải gốc, rất hữu ích cho nhiệm vụ phân tích sau đó. Đầu vào của bộ mã hóa được cung cấp dưới dạng sự kết hợp của *X* và *V\_s* + *V\_g*. Theo cách này, việc trích xuất các tín hiệu hình ảnh được điều kiện hóa một cách phù hợp bởi vị trí điểm bắt đầu và điểm đích.

**Huấn luyện mini-batch.** Quá trình huấn luyện đầy đủ của Neural A\* là rất khó khăn vì cần xử lý nhiều bài toán khác nhau trong một mini-batch. Tuy nhiên, các mẫu trong một mini-batch có thể được giải quyết ở các bước tìm kiếm khác nhau. Chúng tôi giải quyết vấn đề này bằng cách giới thiệu một thao tác kiểm tra mục tiêu nhị phân *eta^(i)(v)* = 1 nếu *|V\_g^(i)|^\_1* và cập nhật *O(i)*, *C(i)* như sau:

**

Trực giác ở đây là các thao tác giữ nguyên *O(i)* và *C(i)* không thay đổi cho đến khi mục tiêu được tìm thấy. Chúng tôi lặp lại các Dòng 9-15 cho đến khi tất cả các mẫu trong mini-batch đạt được mục tiêu.

**4. Thử nghiệm**

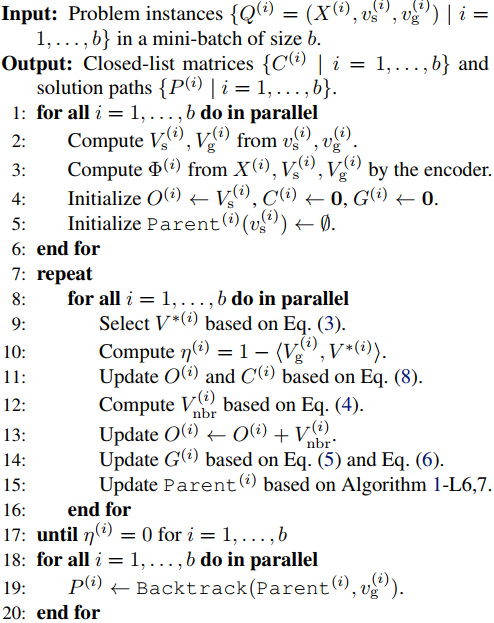
Trong phần này, chúng tôi đánh giá tác động của Neural A\* đối với khả năng thực nghiệm trong việc tối ưu hóa tìm kiếm và hiệu quả, tức là Neural A\* có thể tìm được các đường đi gần tối ưu cho bài toán tìm đường ngắn nhất giữa hai điểm như thế nào.

**4.1. Bộ dữ liệu**

Chúng tôi áp dụng các bộ dữ liệu lập kế hoạch đường đi công khai sau đây với các chú thích chướng ngại vật.

* **Bộ dữ liệu Motion Planning (MP).** Bộ sưu tập gồm 8 loại bản đồ thế giới lưới với các hình dạng chướng ngại vật riêng biệt, được tạo bởi Bhardwaj và các cộng sự (2017). Mỗi nhóm môi trường bao gồm 800 mẫu huấn luyện, 100 mẫu xác nhận và 100 mẫu kiểm tra, với cùng loại chướng ngại vật được đặt ở các vị trí khác nhau. Chúng tôi thay đổi kích thước mỗi môi trường thành 32 × 32 để hoàn thành toàn bộ thử nghiệm trong một khoảng thời gian hợp lý. Theo thiết lập ban đầu, quá trình đánh giá được tiến hành cho từng nhóm môi trường một cách độc lập.
* **Bộ dữ liệu Tiled MP.** Là một phần mở rộng của bộ dữ liệu MP, chúng tôi thiết lập bốn bản đồ phức tạp và đa dạng hơn một cách ngẫu nhiên. Bộ dữ liệu MP được sử dụng để tạo bản đồ môi trường với kích thước 64 × 64. Chúng tôi lặp lại quá trình này để tạo ra 3.200 mẫu huấn luyện, 400 mẫu xác nhận và 400 mẫu kiểm tra.
* **Bộ dữ liệu City/Street Map (CSM).** Bộ sưu tập gồm 30 bản đồ thành phố với các chú thích chướng ngại vật, được tổ chức bởi Sturtevant (2012). Đối với việc tăng cường dữ liệu, chúng tôi cắt ngẫu nhiên nhiều mẫu từ mỗi bản đồ và thay đổi kích thước chúng thành 128 × 128, từ mỗi bản đồ tạo ra 20 mẫu huấn luyện, 10 mẫu xác nhận và 10 mẫu kiểm tra, với tổng cộng 64 × 64. Chúng tôi sử dụng 20 trong số 30 bản đồ để tạo ra 3.200 mẫu huấn luyện và 400 mẫu kiểm tra cho 400 mẫu xác nhận. Theo cách này, chúng tôi đảm bảo rằng không có mẫu nào được chia sẻ giữa các tập huấn luyện/validation và tập kiểm tra.

Thuật toán 2: Tìm kiếm Neural A\*



Đối với mỗi bản đồ, chúng tôi tạo các trường hợp bài toán bằng cách chọn ngẫu nhiên một tập hợp gồm một trong bốn vùng góc của bản đồ làm điểm bắt đầu và một trong bốn vị trí góc khác làm điểm đích, đảm bảo tính hợp lệ của mục tiêu và tập kiểm tra, tương ứng. Từ đó, chúng tôi lấy được các đường đi ngắn nhất chuẩn bằng thuật toán Dijkstra.

**4.2. Phương pháp và cải tiến**

*Neural A.*\* Bộ mã hóa mà chúng tôi áp dụng là U-Net (Ronneberger và các cộng sự, 2015) với VGG-16 backbone (Simonyan & Zisserman, 2015), được cải tiến bởi Yakubovskiy (2020), trong đó các tầng cuối được kích hoạt với hàm sigmoid để hạn chế khoảng giá trị của *Phi* trong khoảng 0 đến 1. Đối với mô-đun A\* khả vi, chúng tôi sử dụng hàm heuristic Chebyshev cho thế giới lưới 8-lân cận (Patel, 1997). Khoảng cách Euclidean được sử dụng cho việc chia nhỏ hằng số (0.001) *tau*. Trong phương trình (3), giá trị *t* được đặt theo căn bậc hai của chiều rộng bản đồ.

**Các phương pháp cơ sở.** Để đánh giá tầm quan trọng của Neural A\* trong lập kế hoạch dựa trên tìm kiếm, chúng tôi áp dụng các phương pháp cơ sở cạnh tranh có đảm bảo thành công lập kế hoạch, tương tự như chúng tôi, theo thiết kế. Chúng tôi mở rộng các cải tiến có sẵn trực tuyến của các tác giả.

* **SAIL (Choudhury và các cộng sự, 2018).** Một phương pháp tìm kiếm tốt nhất trước dựa trên dữ liệu, đạt được hiệu quả cao bằng cách học hàm heuristic từ các thử nghiệm. Không giống như Neural A\*, SAIL sử dụng các khoảng cách đường thẳng từ mỗi nút đến mục tiêu và các mục tiêu phụ gần nhất. Chúng tôi đánh giá hai biến thể của SAIL, trong đó các mẫu huấn luyện được cung cấp bởi hàm heuristic đã học (SAIL) hoặc bởi bộ lập kế hoạch chuẩn (SAIL-SL).
* **Blackbox Differentiation (Vlastelica và các cộng sự, 2020).** Một cách tiếp cận tổng quát để giải quyết bài toán lập kế hoạch đường đi dựa trên dữ liệu, bao gồm một bộ mã hóa biến đổi các bản đồ môi trường thành các bản đồ chi phí không gian và một mô-đun thuật toán rời rạc cố định giống như một hộp đen để thực hiện tìm kiếm A\* (BB-A\*). Chúng tôi triển khai một phiên bản tìm kiếm A\* khả vi (BB-A\*) bằng cách sử dụng mô-đun A\* khả vi của chúng tôi để tối ưu hóa hộp đen, trong khi giữ nguyên các thiết lập khác, chẳng hạn như bộ mã hóa và hàm tổn thất, giống như Neural A\*.

Chúng tôi cũng đánh giá tìm kiếm tốt nhất trước (BF) và tìm kiếm A\* có trọng số (Pohl, 1970) làm các phương pháp cơ sở cổ điển, trong đó cả hai đều sử dụng hàm heuristic giống như Neural A\*. Ngoài ra, chúng tôi áp dụng một phiên bản nâng cao của Neural BF, trong đó *G* = *Phi* trong phương trình (3), thay vì được cập nhật dần dần như trong phương trình (5). Bằng cách này, Neural BF bỏ qua việc tích lũy chi phí hướng dẫn từ nút bắt đầu, tương tự như tìm kiếm tốt nhất trước.

**4.3. Thiết lập thử nghiệm**

Tất cả các mô hình đều được huấn luyện bằng cách sử dụng tối ưu hóa RMSProp, với số lần lặp epoch là (100, 100, 0.001) cho bộ dữ liệu MP và (100, 400, 0.001) cho bộ dữ liệu Tiled MP và CSM.

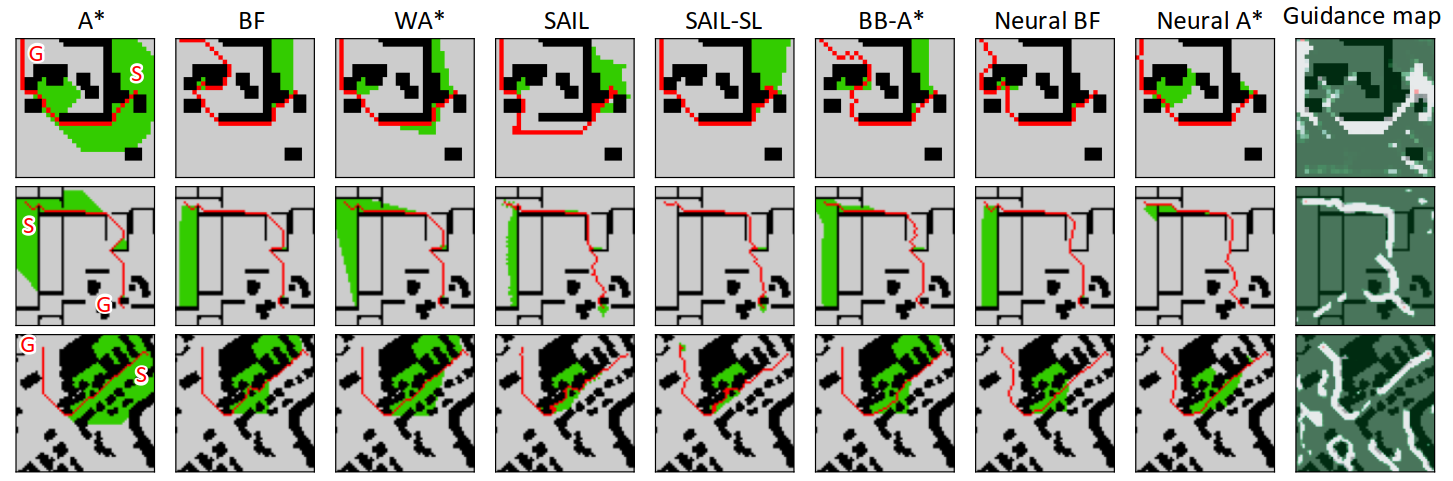
Để huấn luyện mô hình, chúng tôi đo lường sự đánh đổi giữa tính tối ưu của tìm kiếm và hiệu quả bằng cách cải thiện từ tìm kiếm A\* tiêu chuẩn, dựa trên các số liệu sau:

* **Tỷ lệ đường đi tối ưu (Opt).** Đo lường tỷ lệ phần trăm các dự đoán đường đi ngắn nhất cho mỗi bản đồ môi trường, theo cách được sử dụng bởi Vlastelica và các cộng sự (2020).
* **Tỷ lệ giảm số bước mở rộng nút (Exp).** Đo lường số bước mở rộng nút được giảm so với tìm kiếm A\* tiêu chuẩn trong khoảng từ 0 đến 100 (%). Cụ thể, với *E* và *E'* lần lượt là số lần mở rộng nút của A\* và mô hình tương ứng, nó được định nghĩa bằng 100 nhân (*E* - *E'*) chia *E* và được lấy trung bình trên tất cả các bài toán trong mỗi môi trường.
* **Độ đo Harmonic mean (Hmean) của Opt và Exp.** Hiển thị mức độ cải thiện của tỷ lệ đường đi tối ưu và số bước mở rộng nút trên một mô hình.

Trong quá trình huấn luyện, chúng tôi lưu trữ các trọng số mô hình đánh dấu điểm kiểm tra tốt nhất dựa trên độ đo xác nhận của tập chia huấn luyện, để tính toán các độ đo thống kê trên tập chia kiểm tra, với khoảng tin cậy trung bình bootstrap là 95% cho mỗi độ đo.

**4.4. Kết quả**

**So sánh với các phương pháp cơ sở.** Bảng 1 tóm tắt kết quả định lượng tổng thể. Neural A\* vượt trội hơn các phương pháp cơ sở về cả độ đo Opt và Hmean, đồng thời cải thiện hiệu quả tìm kiếm và tỷ lệ đường đi tối ưu. SAIL và SAIL-SL đôi khi hoạt động hiệu quả hơn, trong đó SAIL có tỷ lệ tối ưu thấp hơn một chút so với Neural A\*, trong khi SAIL-SL ghi nhận điểm Hmean cao hơn so với SAIL nhưng vẫn thấp hơn Neural A\*. Sự khác biệt duy nhất giữa Neural A\* và BB-A\* là cách tạo tìm kiếm A\* khả vi hoặc xem nó như một hộp đen trong việc cập nhật một cách khác biệt của mô-đun A\* khả vi. Neural A\* có quyền truy cập vào thông tin phong phú hơn, điều này cần thiết để phân tích mối quan hệ nhân quả giữa các lựa chọn nút riêng lẻ trong các kết quả tìm kiếm. Thông tin này, tuy nhiên, được đóng hộp đen trong BB-A\*. Chúng tôi cũng nhận thấy rằng các bộ lập kế hoạch heuristic cổ điển (BF và WA\*) hoạt động tương đương hoặc đôi khi tốt hơn so với các phương pháp cơ sở dựa trên dữ liệu khác. Kết quả này có thể được giải thích bởi thiết lập thử nghiệm đầy thách thức của chúng tôi, trong đó sử dụng các vị trí bắt đầu và đích được chọn ngẫu nhiên thay vì các vị trí được định nghĩa sẵn từ các nghiên cứu ban đầu (Choudhury và các cộng sự, 2018; Vlastelica và các cộng sự, 2020). Cuối cùng, Neural BF đạt được điểm số Exp cao nhất trên các bộ dữ liệu Tiled MP và CSM, nhưng thường kết thúc với các đường đi không tối ưu vì nó bỏ qua việc tích lũy chi phí thực tế. Để biết thêm các đánh giá bao gồm phân tích về độ phức tạp và thời gian chạy của Neural A\*, hãy xem Phụ lục B.



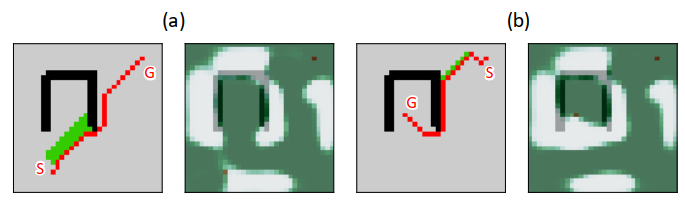
**Hình 3. Kết quả lập kế hoạch đường đi được chọn.**

Các điểm ảnh màu đen biểu thị chướng ngại vật, các nút bắt đầu (được biểu thị bằng "S"), các nút đích (được biểu thị bằng "G") và các đường đi được tìm thấy được chú thích bằng màu đỏ. Các nút được mở rộng khác được tô màu xanh lá cây. Trong cột bên phải nhất, các bản đồ hướng dẫn được phủ lên trên các bản đồ đầu vào, nơi các vùng có chi phí thấp được trực quan hóa bằng màu trắng. Các kết quả chi tiết hơn được trình bày trong Phụ lục B.

**Kết quả định tính.** Hình 3 trực quan hóa các ví dụ kết quả tìm kiếm với các bản đồ hướng dẫn được tạo ra bởi bộ mã hóa của Neural A\*. Chúng tôi xác nhận rằng bộ mã hóa đã thành công trong việc ghi nhận các tín hiệu hình ảnh trong các bài toán. Cụ thể, các chi phí hướng dẫn cao hơn (được hiển thị bằng màu xanh lá cây) được gán cho toàn bộ các vùng chướng ngại vật tạo ra ngõ cụt, trong khi các chi phí thấp hơn (được hiển thị bằng màu trắng) được gán cho các lối đi vòng và đường tắt dẫn hướng tìm kiếm đến mục tiêu. Hình 4 mô tả cách Neural A\* thực hiện tìm kiếm thích nghi cho các vị trí bắt đầu và đích khác nhau trong cùng một bản đồ. Lưu ý rằng lối vào của ngõ cụt hình chữ U được (a) chặn với chi phí hướng dẫn cao khi ngõ cụt nằm giữa điểm bắt đầu và điểm đích, và (b) mở khi điểm đích nằm bên trong ngõ cụt.

**Nghiên cứu ablation.** Chúng tôi tiếp tục đánh giá ảnh hưởng của bộ mã hóa bằng cách so sánh một số cấu hình khác nhau cho kiến trúc của nó. Trong cấu hình không có điểm bắt đầu và đích, chúng tôi chỉ cung cấp bản đồ môi trường *Xi* làm đầu vào cho bộ mã hóa. Trong cấu hình với ResNet-18 backbone, một kiến trúc mạng nơ-ron dư (He và các cộng sự, 2016) được sử dụng thay vì VGG-16. Bảng 2 cho thấy hiệu suất suy giảm với các cấu hình này, đặc biệt là về tỷ lệ tối ưu. Ngoài ra, chúng tôi đã thử các biến thể sau của phương pháp được đề xuất, nhưng nhận thấy chúng không hiệu quả. Việc áp dụng straight-through Gumbel-Softmax (Jang và các cộng sự, 2017) cho phương trình (3) gây ra thất bại hoàn toàn trong lập kế hoạch (với Opt, Exp, Hmean đều bằng 0) vì nó làm lệch kế hoạch tìm kiếm, tránh các nút hứa hẹn nhất. Việc sử dụng tổn thất bình phương trung bình cho phương trình (7) tạo ra gradient tương tự như tổn thất trung bình L1, chỉ khác một hằng số đối với các biến nhị phân (tức là, *C* và ), và thực tế dẫn đến hiệu suất tương đương.

**Hạn chế.** Mặc dù Neural A\* hoạt động tốt trong nhiều môi trường khác nhau, triển khai hiện tại của nó giả định rằng các môi trường là lưới thế giới với chi phí nút đơn vị. Một hướng nghiên cứu thú vị là mở rộng Neural A\* để hoạt động trên không gian trạng thái đa chiều hoặc liên tục. Điều này sẽ đòi hỏi bộ mã hóa được điều chỉnh để phù hợp với không gian như vậy, như đã được thực hiện trong Qureshi và các cộng sự (2019); Chen và các cộng sự (2020); Ichter & Pavone (2019). Chúng tôi để lại phần mở rộng này cho công việc tương lai.



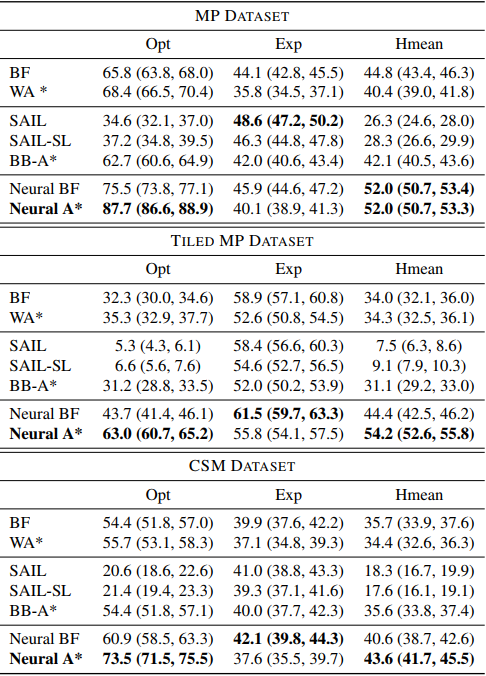
**Hình 4. Kết quả mã hóa thích nghi** (a) Chướng ngại vật hình chữ U tạo ngõ cụt được đặt giữa nút bắt đầu và nút đích; (b) Nút đích nằm bên trong ngõ cụt hình chữ U.

**5. Lập kế hoạch đường đi trên hình ảnh thô**

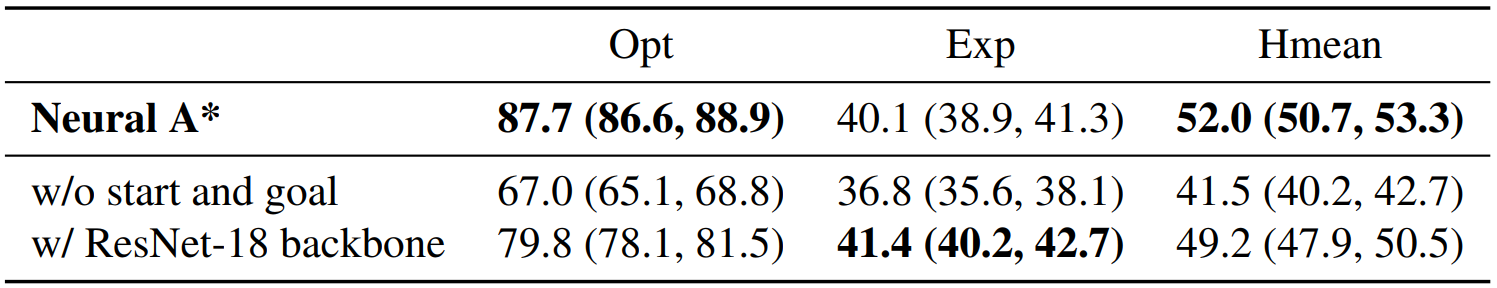
Như một kịch bản khác cho Neural A\*, chúng tôi giải quyết nhiệm vụ lập kế hoạch đường đi trực tiếp trên hình ảnh thô. Cụ thể, hãy giả sử một video ghi lại cảnh ngoài trời được quay bởi một camera giám sát tĩnh. Các minh họa lập kế hoạch bao gồm các hình ảnh màu của cảnh và các quỹ đạo thực tế của người đi bộ (tức là, các đường đi thực tế được cung cấp bởi người chú thích). Dựa trên dữ liệu này, chúng tôi sử dụng Neural A\* để dự đoán các quỹ đạo thực tế phù hợp với quỹ đạo của người đi bộ khi được cung cấp vị trí bắt đầu và đích. Ở đây, chúng tôi so sánh Neural A\* với BB-A\* (Vlastelica và các cộng sự, 2020) như một đối thủ cạnh tranh cũng có khả năng lập kế hoạch trên hình ảnh thô. Để biết thêm các so sánh với những người học bắt chước khác (Ratliff và cộng sự, 2006; Tamar và cộng sự, 2016; Lee và cộng sự, 2018), xem Phụ lục B.1.

**Bộ dữ liệu**. Chúng tôi sử dụng Bộ dữ liệu Stanford Drone (SDD) (Robicquet và các cộng sự, 2016), bao gồm các video giám sát được quay bởi camera drone tĩnh tại tám địa điểm riêng biệt. Chúng tôi chia bộ dữ liệu theo hai cách: (1) *Intra-scenes*: đối với mỗi địa điểm, chọn một video để tạo tập kiểm tra đơn, trong khi sử dụng các video còn lại để huấn luyện mô hình, nhằm xem liệu mô hình có thể dự đoán quỹ đạo của các cá nhân chưa từng thấy được quan sát vào các thời điểm khác nhau hay không; (2) *Inter-scenes*: thực hiện kiểm định chéo loại bỏ một địa điểm (leave-one-location-out cross-validation) để đánh giá khả năng tổng quát hóa của mô hình đã học đối với các địa điểm chưa từng thấy. Để làm minh họa lập kế hoạch, chúng tôi trích xuất các quỹ đạo của người đi bộ và vùng cục bộ của khung hình video bao quanh các quỹ đạo đó.

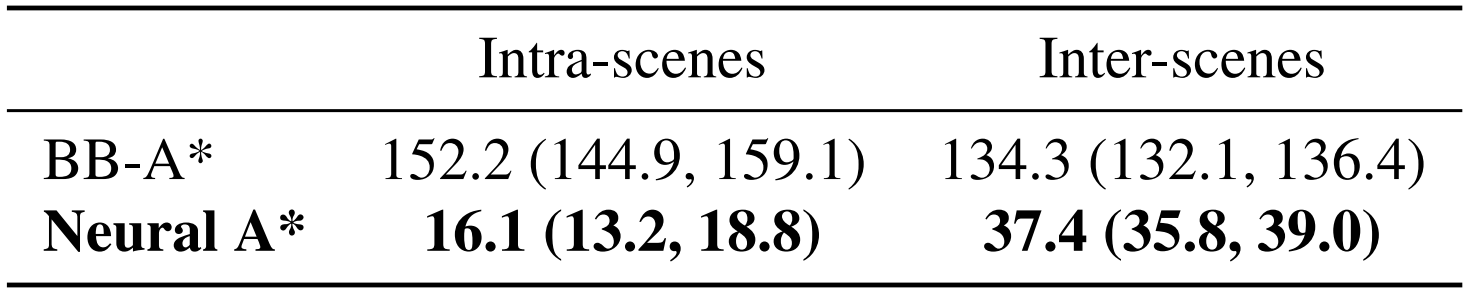
**Bảng 1. Kết quả định lượng.** Giá trị trung bình bootstrap và khoảng tin cậy 95% của tỷ lệ tối ưu đường đi (Opt), tỷ lệ giảm số bước mở rộng nút (Exp), và trung bình điều hòa (Hmean) của chúng.



**Bảng 2. Nghiên cứu ablation.** So sánh hiệu suất với các lựa chọn thiết kế kiến trúc khác nhau cho bộ mã hóa trên bộ dữ liệu MP.



**Bảng 3. Kết quả định lượng trên SDD.** Giá trị trung bình bootstrap và khoảng tin cậy 95% của khoảng cách Chamfer giữa các quỹ đạo người đi bộ được dự đoán và quỹ đạo thực tế.

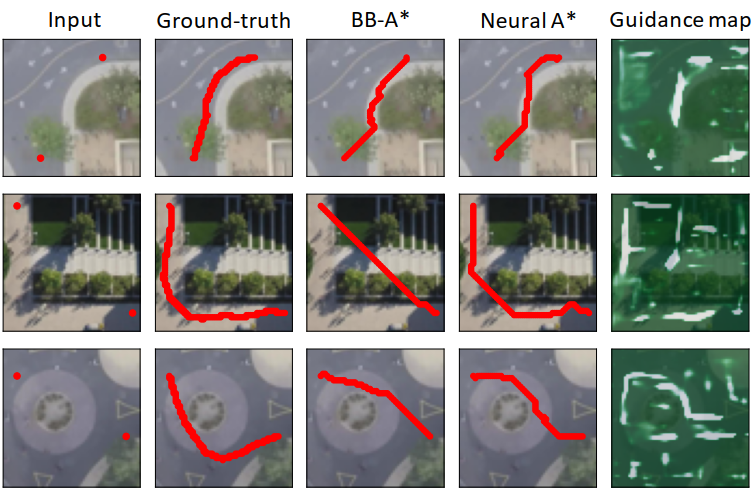


**Triển khai và thiết lập thử nghiệm.** Không giống như các thử nghiệm trước đây với vị trí chướng ngại vật được xác định rõ ràng, mỗi mô hình giờ đây phải học các biểu diễn hình ảnh của chướng ngại vật để tránh chúng trong bộ mã hóa U-Net bằng cách nhân với hàm sigmoid được khởi tạo bằng giá trị dương (khởi tạo là 10.0), để các vùng chướng ngại vật có thể được gán chi phí đủ cao. Chúng tôi đã huấn luyện cả Neural A\* và BB-A\* bằng RMSProp với kích thước batch, số lần lặp và tỷ lệ học lần lượt là (64, 20, 0.001). Vì mục tiêu chính của bài toán là dự đoán các đường đi gần với đường đi thực tế, thay vì cải thiện tính tối ưu và hiệu quả tìm kiếm, chúng tôi đã tính khoảng cách Chamfer làm độ đo để đánh giá sự khác biệt giữa các đường đi này.

**Kết quả.** Bảng 3 cho thấy Neural A\* vượt trội hơn BB-A\*. Như được trực quan hóa trong hai ví dụ đầu tiên ở Hình 5, Neural A\* thường dự đoán các đường đi dọc theo các con đường, dẫn đến các đường đi thực tế gần hơn với đường đi thực tế so với BB-A\*. Tuy nhiên, cả hai phương pháp đôi khi không dự đoán chính xác các quỹ đạo người đi bộ khi có nhiều tuyến đường khả thi đến đích, như được minh họa trong ví dụ ở dưới cùng. Một hướng mở rộng khả thi cho vấn đề này là áp dụng một khung tính toán tổng quát (Gupta và các cộng sự, 2018; Salzmann và các cộng sự, 2020) cho phép lập kế hoạch đường đi có tính ngẫu nhiên.

**6. Công việc liên quan**

Các phương pháp giải quyết bài toán lập kế hoạch đường đi có thể được phân loại rộng rãi thành nhiều loại, mỗi loại có ưu điểm và hạn chế riêng. Ví dụ, lập kế hoạch dựa trên lấy mẫu, như RRT (LaValle, 1998) và PRM (Kavraki và các cộng sự, 1996), có thể được sử dụng trong không gian trạng thái chuyển động cao cấp và được phát triển rộng rãi để lập kế hoạch cho robot (Kingston và các cộng sự, 2018). Một thách thức quan trọng là xác định các vùng quan trọng trong không gian trạng thái để lấy mẫu các điểm trạng thái thưa một cách hiệu quả. Để giải quyết vấn đề này, nhiều phương pháp dựa trên dữ liệu đã được phát triển nhằm học các vùng quan trọng (Ichter và các cộng sự, 2018; Ichter & Pavone, 2019) hoặc học các chiến lược khám phá (Qureshi và các cộng sự, 2019; Pérez-Higueras và các cộng sự, 2018; Chen và các cộng sự, 2020) từ các minh họa chuyên gia hoặc các thử nghiệm lập kế hoạch thành công.



**Hình 5. Ví dụ lập kế hoạch đường đi trên SDD.** Neural A\* đã dự đoán các đường đi tương tự như quỹ đạo thực tế của người đi bộ.

Trong khi đó, các bộ lập kế hoạch phản ứng học được một chính sách di chuyển của tác nhân để xác định hành động tốt nhất tiếp theo, chẳng hạn như di chuyển sang trái hoặc sang phải, từ các trạng thái hiện tại thông qua học có giám sát (Tamar và các cộng sự, 2016; Kanezaki và các cộng sự, 2017; Gupta và các cộng sự, 2017; Karkus và các cộng sự, 2017; Lee và các cộng sự, 2018; Bency và các cộng sự, 2019) hoặc học tăng cường (IRL) (Ratliff và các cộng sự, 2006; Ziebart và các cộng sự, 2008; Wulfmeier và các cộng sự, 2015; Kim & Pineau, 2016; Kretzschmar và các cộng sự, 2016; Fu và các cộng sự, 2018; Yu và các cộng sự, 2019). Các phương pháp này có thể hữu ích cho các môi trường động, nơi các tác nhân cần phản ứng thích nghi. Tuy nhiên, chúng thường gặp khó khăn trong việc phục hồi hàm chi phí từ các minh họa, đặc biệt đối với các bài toán lập kế hoạch dài hạn và phức tạp (Faust và các cộng sự, 2018). Trong Phụ lục B.1, chúng tôi cung cấp các đánh giá định lượng với một số phương pháp liên quan (Ratliff và các cộng sự, 2006; Tamar và các cộng sự, 2016; Lee và các cộng sự, 2018) trong thiết lập thử nghiệm của chúng tôi.

So với các phương pháp này, lập kế hoạch dựa trên tìm kiếm có lợi thế về việc đảm bảo tìm được các đường đi hợp lệ trong một bản đồ lưới thế giới. Các bộ lập kế hoạch heuristic cổ điển tìm kiếm tốt hơn các hàm heuristic và triển khai hiệu quả để cải thiện hiệu suất tìm kiếm (ví dụ, Burns và các cộng sự, 2012; Zhou & Zeng, 2015; Abd Algfoor và các cộng sự, 2015). Các phương pháp dựa trên dữ liệu gần đây đã mở rộng các bộ lập kế hoạch heuristic để học (a) các đường đi gần tối ưu hiệu quả từ các minh họa chuyên gia (Choudhury và các cộng sự, 2018; Takahashi và các cộng sự, 2019) hoặc (b) lập kế hoạch trực tiếp trên hình ảnh thô (Vlastelica và các cộng sự, 2020). Đứng trên cả hai khía cạnh này, công việc của chúng tôi đề xuất bộ lập kế hoạch dựa trên tìm kiếm đầu tiên có thể giải quyết cả hai vấn đề này một cách nguyên tắc. Từ góc độ khác, các bộ lập kế hoạch heuristic dựa trên dữ liệu có thể học các hàm chi phí của chúng, như trong nghiên cứu của chúng tôi và Vlastelica và các cộng sự (2020), cũng như các hàm heuristic của chúng, như trong Choudhury và các cộng sự (2018); Takahashi và các cộng sự (2019). Những biến thể tương tự bao gồm học phần thưởng (negative reward) trong IRL và một hàm Q trong RL, tương ứng. Gần đây, Archetti và các cộng sự (2021) đã mở rộng Neural A\* để học cả hai hàm này.

**7. Kết luận**

Chúng tôi đã giới thiệu một bộ lập kế hoạch dựa trên tìm kiếm theo dữ liệu mới có tên Neural A\*, kết hợp thuật toán A\* khả vi. Neural A\* học từ các minh họa để cải thiện tính tối ưu và hiệu quả của tìm kiếm đường đi, đồng thời cho phép lập kế hoạch trực tiếp trên hình ảnh thô. Các đánh giá thử nghiệm rộng rãi trên nhiều bộ dữ liệu công khai đã chứng minh hiệu quả của phương pháp của chúng tôi so với các bộ lập kế hoạch tiên tiến.

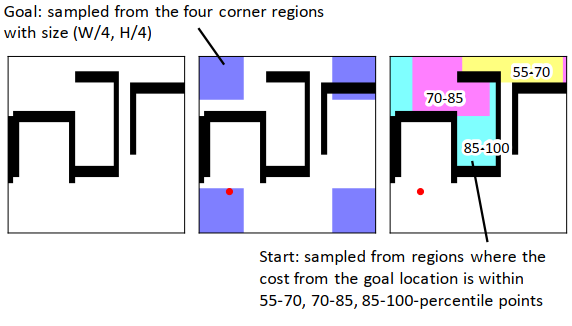
**Tài liệu tham khảo**

1. Abd Algfoor, Z., Sunar, M. S., and Kollvand, H. (2015). A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, 2015.
2. Anderson, P., Chang, A., Chaplot, D. S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al. (2018). On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*.
3. Archetti, A., Cannici, M., and Matteucci, M. (2021). Neural weighted A*: Learning graph costs and heuristics with differentiable anytime A*. *arXiv preprint arXiv:2105.01480*.
4. Bency, M. J., Qureshi, A. H., and Yip, M. C. (2019). Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
5. Bhardwaj, M., Choudhury, S., and Scherer, S. (2017). Learning heuristic search via imitation. In *Proceedings of the Conference on Robot Learning (CoRL)*.
6. Burns, E. A., Hatem, M., Leighton, M. J., and Ruml, W. (2012). Implementing fast heuristic search code. In *Proceedings of the Annual Symposium on Combinatorial Search (SOCS)*.
7. Chen, B., Dai, B., Lin, Q., Ye, G., Liu, H., and Song, L. (2020). Learning to plan in high dimensions via neural exploration-exploitation trees. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
8. Choudhury, S., Bhardwaj, M., Arora, S., Kapoor, A., Ranade, G., Scherer, S., and Dey, D. (2018). Data-driven planning via imitation learning. *International Journal of Robotics Research (IJRR)*, 37(13-14):1652-1672.
9. Faust, A., Oslund, K., Ramirez, O., Francis, A., Tapia, L., Fiser, M., and Davidson, J. (2018). PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113-5120.
10. Fu, J., Luo, K., and Levine, S. (2018). Learning robust rewards with adversarial inverse reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
11. Gonzalez, D., Perez, J., Milanes, V., and Nashashibi, F. (2015). A review of emotion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135-1145.
12. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2255-2264.
13. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. (2017). Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2616-2625.
14. Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100-107.
15. He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778.
16. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4107-4115.
17. Ichter, B. and Pavone, M. (2019). Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):2407-2414.
18. Ichter, B., Harrison, J., and Pavone, M. (2018). Learning sampling distributions for robot motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7087-7094.
19. Ichter, B., Schmerling, E., Lee, T. W. E., and Faust, A. (2020). Learned critical probabilistic roadmaps for robotic motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9535-9541.
20. Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
21. Kanezaki, A., Nitta, J., and Sasaki, Y. (2017). Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks. *IEEE Robotics and Automation Letters (RA-L)*, 3(2):696-703.
22. Karkus, P., Hsu, D., and Lee, W. S. (2017). QMDP-Net: Deep learning for planning under partial observability. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4694-4704.
23. Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566-580.
24. Kim, B. and Pineau, J. (2016). Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8(1):51-66.
25. Kingston, Z., Moll, M., and Kavraki, L. E. (2018). Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:159-185.
26. Kretzschmar, H., Spies, M., Sprunk, C., and Burgard, W. (2016). Socially compliant mobile robot navigation via inverse reinforcement learning. *International Journal of Robotics Research (IJRR)*, 35(11):1289-1307.
27. LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. *Technical report*, Computer Science Dept., Iowa State University.
28. Lee, L., Parisotto, E., Chaplot, D. S., Xing, E., and Salakhutdinov, R. (2018). Gated path planning networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.
29. Paden, B., Cap, M., Yong, S. Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33-55.
30. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024-8035.
31. Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2006). Maximum margin planning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 729-736.
32. Robicquet, A., Sadeghian, A., Alahi, A., and Savarese, S. (2016). Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 549-565.
33. Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pp. 234-241.
34. Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. (2020). Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
35. Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
36. Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D. V., and Kragic, D. (2012). Dual arm manipulation - A survey. *Robotics and Autonomous Systems*, 60(10):1340-1353.
37. Sturtevant, N. (2012). Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):144-148.
38. Takahashi, T., Sun, H., Tian, D., and Wang, Y. (2019). Learning heuristic functions for mobile robot path planning using deep neural networks. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 764-772.
39. Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. (2016). Value iteration networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2154-2162.
40. Tange, O. (2011). GNU parallel - The command-line power tool. *The USENIX Magazine*, 36(1):42-47.
41. Vlastelica, M., Paulus, A., Musil, V., Martius, G., and Rolinek, M. (2020). Differentiation of blackbox combinatorial solvers. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
42. Wulfmeier, M., Ondruska, P., and Posner, I. (2015). Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*.
43. Yakubovskiy, P. (2020). Segmentation models pytorch. *GitHub repository*.
44. Yu, L., Yu, T., Finn, C., and Ermon, S. (2019). Meta-inverse reinforcement learning with probabilistic context variables. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11772-11783.
45. Zhou, Y. and Zeng, J. (2015). Massively parallel A\* search on a GPU. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.
46. Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pp. 1433-1438.

**A. Chi tiết về thiết lập thử nghiệm**

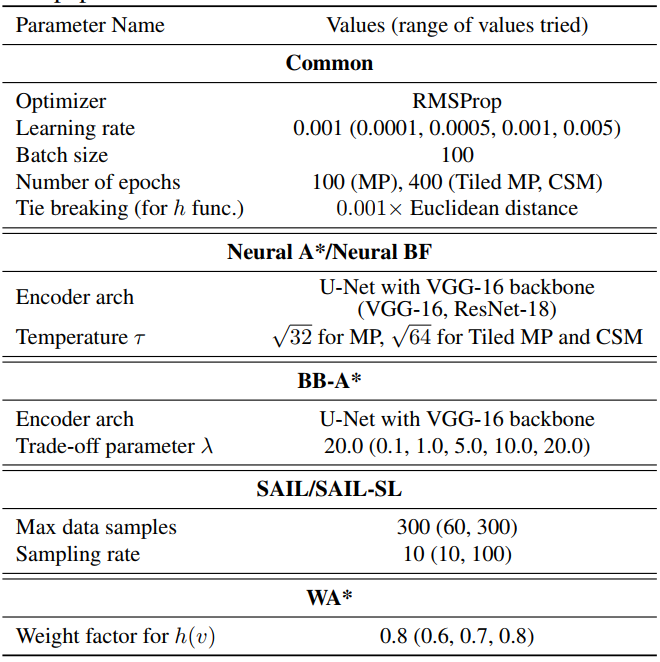
**A.1. Tạo bộ dữ liệu**

Ở đây, chúng tôi trình bày thông tin bổ sung về cách mỗi bộ dữ liệu được tạo ra trong các thí nghiệm của chúng tôi. Vì quá trình tạo dữ liệu của chúng tôi có tính ngẫu nhiên, chúng tôi đã cố định một hạt giống ngẫu nhiên (random seed) trong mỗi chương trình tạo dữ liệu để đảm bảo khả năng tái tạo. Vui lòng tham khảo mã nguồn tại trang dự án của chúng tôi: <https://omron-sinicx.github.io/neural-astar/>.



Hình 6. **Lấy mẫu vị trí bắt đầu và đích.**

Bảng 4. **Lựa chọn siêu tham số**. Danh sách các siêu tham số và phạm vi của các giá trị này đã được thử nghiệm trong quá trình phát triển bài báo này



**Bộ dữ liệu MP/Tiled-MP/CSM.** Trong các thí nghiệm với bộ dữ liệu MP/Tiled-MP/CSM, chúng tôi đã áp dụng các thiết lập thử thách hơn, liên quan đến các vị trí bắt đầu và đích được chọn ngẫu nhiên thay vì các vị trí cố định được xác định trước như trong các nghiên cứu trước đó (Choudhury và các cộng sự, 2018; Vlastelica và các cộng sự, 2020). Chúng tôi đã xác định các vị trí bắt đầu và đích này một cách chiến lược dựa trên khoảng cách thực tế của chúng để tránh tạo ra các bài toán quá dễ. Cụ thể, đối với mỗi bản đồ môi trường, một vị trí đích duy nhất được xác định ngẫu nhiên một lần và cố định xuyên suốt các thí nghiệm. Ở đây, đối với một bản đồ có chiều rộng và chiều cao được ký hiệu là (*W*, *H*), tức là (32, 32) cho MP và (64, 64) cho bộ dữ liệu Tiled MP và CSM, vị trí đích được lấy mẫu từ một trong bốn vùng góc có kích thước (*W*/4, *H*/4), như được minh họa trong Hình 6 (giữa). Sau đó, chúng tôi thực hiện thuật toán Dijkstra để tính toán chi phí di chuyển thực tế từ mọi nút có thể đi qua đến đích, và tính toán các điểm phân vị 55, 70, 85 của chi phí. Trong mỗi lần lặp của giai đoạn huấn luyện, chúng tôi lấy mẫu một vị trí bắt đầu ngẫu nhiên mới từ các vùng có chi phí thực tế cao hơn điểm phân vị 55. Đối với dữ liệu xác nhận và kiểm tra, chúng tôi lấy mẫu lần lượt hai và năm vị trí bắt đầu ngẫu nhiên nhưng cố định, từ mỗi loại vùng có chi phí nằm trong các khoảng phân vị [55, 70], [70, 85], và [85, 100] (xem Hình 6 (phải) để minh họa). Kết quả là, chúng tôi đã tạo ra 2 × 3 = 6 và 5 × 3 = 15 vị trí bắt đầu đa dạng cho mỗi bản đồ xác nhận và kiểm tra, tương ứng. Các đường đi thực tế (ground-truth) cho tất cả các bài toán được tạo ra đều được thu thập bằng thuật toán Dijkstra. Khi tính toán tổn thất cho các bộ dữ liệu Tiled MP và CSM, các đường đi này được làm giãn với một nhân 3 × 3, điều này giúp ổn định quá trình huấn luyện.

**SDD.** Trong SDD, chúng tôi đã trích xuất các quỹ đạo tương đối đơn giản của người đi bộ di chuyển trực tiếp đến đích của họ. Cụ thể, đối với mỗi quỹ đạo được cung cấp bởi Robicquet và các cộng sự (2016), chúng tôi đầu tiên cắt ngẫu nhiên quỹ đạo đó để có độ dài chuỗi trong khoảng [300, 600] bước thời gian (tại 2.5fps). Sau đó, chúng tôi tính tỷ lệ giữa khoảng cách đường thẳng từ điểm bắt đầu đến điểm đích và độ dài quỹ đạo, như một thước đo độ đơn giản của đường đi, mang lại giá trị thấp hơn cho quỹ đạo phức tạp hơn. Chúng tôi loại bỏ các quỹ đạo có độ đơn giản nhỏ hơn 0.5. Cuối cùng, chúng tôi cắt các vùng ảnh bao quanh mỗi quỹ đạo với lề 50 điểm ảnh và thay đổi kích thước chúng thành 64×64. Kết quả là, 8.325 mẫu đã được trích xuất từ bộ dữ liệu.

**A.2. Lựa chọn siêu tham số**

Bảng 4 hiển thị danh sách các siêu tham số cũng như phạm vi giá trị của các tham số này mà chúng tôi đã thử để tạo ra kết quả cuối cùng. Chúng tôi đã chọn các tham số cuối cùng dựa trên hiệu suất xác nhận trên bộ dữ liệu MP, dựa trên điểm Hmean. Vì việc hoàn thành tất cả các thí nghiệm mất một khoảng thời gian đáng kể (xem phần tiếp theo), chúng tôi chỉ thực hiện mỗi thí nghiệm một lần với một tập hợp hạt giống ngẫu nhiên cố định.

Dưới đây, chúng tôi đưa ra một số nhận xét liên quan đến danh sách siêu tham số. Chúng tôi nhận thấy rằng việc xử lý trường hợp hòa (tie-breaking) trong tìm kiếm A\*, tức là điều chỉnh *h(v)* bằng cách cộng thêm khoảng cách Euclidean từ *v* đến đích, được nhân với một hằng số dương nhỏ (0.001), là rất quan trọng để cải thiện hiệu quả cơ bản của tìm kiếm A\*. Do đó, chúng tôi đã sử dụng thiết lập này cho tất cả các phương pháp dựa trên A\* trong suốt các thí nghiệm. Việc lựa chọn tỷ lệ học không ảnh hưởng nhiều đến hiệu suất cuối cùng nếu có đủ số lượng lần lặp (epochs). BB-A\* có một siêu tham số bổ sung *λ* kiểm soát sự đánh đổi giữa "tính thông tin của gradient" và "sự trung thực với hàm ban đầu" (Vlastelica và các cộng sự, 2020). Chúng tôi đã thử một số giá trị và nhận thấy rằng bất kỳ lựa chọn nào cũng hoạt động khá tốt, ngoại trừ các giá trị cực kỳ nhỏ (ví dụ, 0.1). SAIL và SAIL-SL (Choudhury và các cộng sự, 2018) có các siêu tham số về cách thu thập mẫu từ mỗi trường hợp môi trường huấn luyện, nhưng điều này ít ảnh hưởng đến hiệu suất cuối cùng. Cuối cùng, phương pháp A\* có trọng số cơ bản đã sử dụng một tiêu chí chọn nút được điều chỉnh với yếu tố trọng số *w* cho hàm heuristic; tức là, (1 − *w*) · *g(v)* + *w* · *h(v)*, trong đó chúng tôi đặt *w* = 0.8 trong suốt các thí nghiệm. Lưu ý rằng việc sử dụng *w* = 1.0 cho tiêu chí này tương ứng với tìm kiếm ưu tiên tốt nhất (best-first search) trong các phương pháp cơ bản của chúng tôi.

**A.3. Hiệu suất tính toán trên máy chủ vận hành**

Chúng tôi đã thực hiện tất cả các thí nghiệm trên một máy chủ vận hành Ubuntu 18.04.3 LTS với GPU NVIDIA V100, Intel Xeon Gold 6252 CPU @ 2.10GHz (48 lõi), và 786GB bộ nhớ chính. Việc triển khai của chúng tôi được xây dựng dựa trên PyTorch 1.5 (Paszke và các cộng sự, 2019) và Segmentation Models Pytorch (Yakubovskiy, 2020). Để thực hiện các thí nghiệm một cách hiệu quả, chúng tôi đã sử dụng GNU Parallel (Tange, 2011) để chạy nhiều thí nghiệm và DocE từ 1.1 trong danh sách tất cả các phụ thuộc.

Trong các phiên huấn luyện, mỗi mô hình được huấn luyện bằng một GPU V100 duy nhất với 16 GB bộ nhớ đồ họa. Việc huấn luyện mỗi mô hình (Neural A\* và Neural BF) mất khoảng 50 phút trên bộ dữ liệu MP (100 epoch × 800 mẫu với kích thước 32 × 32) và 35 giờ trên bộ dữ liệu Tiled MP và CSM (400 epoch × 3.200 mẫu với kích thước 64 × 64). Đối với SAIL/SAIL-SL và BB-A\*, thời gian huấn luyện trên bộ dữ liệu MP tương tự, trong khi trên Tiled MP và CSM, việc huấn luyện SAIL/SAIL-SL và BB-A\* mất khoảng 22 và 65 giờ, tương ứng.

**B. Kết quả bổ sung**

Hình 7 và 8 cho thấy các kết quả định tính bổ sung (bao gồm cả các kết quả trên MP đã được giới thiệu ở trên). Theo đó, chúng tôi đưa ra các đánh giá chi tiết về hiệu suất của Neural A\* từ các góc độ khác nhau.

**B.1. So sánh với các phương pháp học mô phỏng**

Như đã giới thiệu trong Mục 6, một số phương pháp học mô phỏng (IL) rất phổ biến trong việc xây dựng các bộ lập kế hoạch tìm kiếm, nhưng chúng tôi nhận thấy rằng các hàm chi phí âm (negative cost functions) không được biết trước thường không hiệu quả trong các minh họa của chúng tôi. Ở đây, chúng tôi so sánh phương pháp của mình với các tác vụ lập kế hoạch khác, cụ thể là Maximum Margin Planning (MMP) (Ratliff và các cộng sự, 2006), Value Iteration Network (VIN) (Tamar và các cộng sự, 2016) và Gated Path Planning Network (GPPN) (Lee và các cộng sự, 2018).

Đối với MMP, chúng tôi đã tuân theo nghiên cứu ban đầu và mô hình hóa chi phí dưới dạng một ánh xạ tuyến tính tính năng, sử dụng mạng VGG-16 để trích xuất tính năng trên ImageNet. Trong Neural A\*, chúng tôi kích hoạt chi phí với hàm sigmoid để hạn chế giá trị trong khoảng [0, 1]. Không giống như các bộ lập kế hoạch dựa trên IL khác, MMP sử dụng tìm kiếm A\* để lập kế hoạch đường đi với chi phí ước tính. Chúng tôi sử dụng phương pháp của Neural A\* làm cơ sở.

Đối với VIN và GPPN, chúng tôi đã sử dụng mã nguồn chính thức của Lee và các cộng sự (2018). Vì các bộ lập kế hoạch phản ứng này không sử dụng thuật toán tìm kiếm nên chúng không thể tận dụng Exp và Hmean, là các độ đo liên quan đến hiệu suất của tìm kiếm. Thay vào đó, chúng tôi tính toán tỷ lệ thành công (Suc), là tỷ lệ phần trăm của các bài toán mà bộ lập kế hoạch tìm được một đường đi hợp lệ.

Như được hiển thị trong Bảng 5 và 6, chúng tôi xác nhận rằng hiệu suất của các phương pháp IL này bị hạn chế so với phương pháp của Neural A\*. Mặc dù MMP đạt được tỷ lệ thành công 100% trong việc sử dụng tìm kiếm A\*, nhưng về mặt Opt, Exp và Hmean, phương pháp này không thể hiện được lý do mà các bước lịch sử và kết quả định tính của Neural A\* vượt trội hơn BB-A\* trong Mục 4.4. Trong khi GPPN đạt được điểm Opt cao hơn Neural A\* trên bộ dữ liệu Tiled MP, nó không đạt được điểm thành công cao trên một đường đi hợp lệ như đã được hiển thị trên SDD. Hơn nữa, GPPN và VIN hoàn toàn thất bại trong việc học từ các minh họa phức tạp (một quỹ đạo trên mỗi bài toán), do đó dẫn đến các cài đặt thử thách hơn của tác phẩm gốc.

**B.2. Đánh giá độ dài tối ưu của đường đi**

Theo sau Tamar và các cộng sự (2016), Anderson và các cộng sự (2018), chúng tôi giới thiệu một độ đo khác để đánh giá độ tối ưu của đường đi, gọi là *P*. Để đảm bảo tính nhất quán với các độ đo khác, độ dài đường đi tối ưu được tính trong khoảng từ 0 đến 100% và được ký hiệu là |P| / |*P*| × 100. Như được hiển thị trong Bảng 5, Neural A\* tạo ra các đường đi gần tối ưu nhất trên tất cả các bộ dữ liệu.

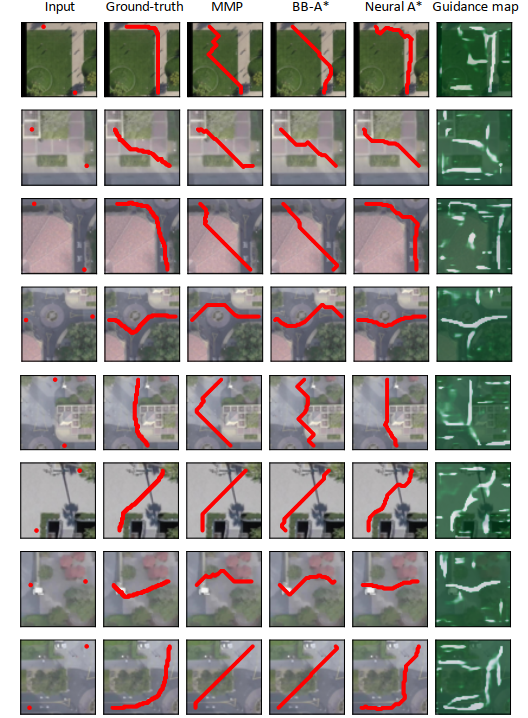
**B.3. Độ phức tạp tính toán và phân tích thời gian chạy**

Mô-đun tính toán chính của Neural A\* nằm ở phần khác biệt của thuật toán A\*. Vì mô-đun này sử dụng các phép toán ma trận bao gồm tất cả các nút có thể đi qua trong quá trình lan truyền, độ phức tạp của nó phụ thuộc vào số lượng bước tìm kiếm *k* và số lượng nút có thể đi qua trên đường đi thực tế *d*. Do đó, độ phức tạp tính toán của *O(d)* và *O(k)* được biểu diễn cho trường hợp tốt nhất và xấu nhất, tương ứng, khi số lượng nút mở rộng tìm kiếm tăng lên theo từng bước. Trong thực tế, khi huấn luyện hoàn tất, chúng tôi có thể thay thế mô-đun khác biệt của A\* bằng thuật toán tìm kiếm A\* tiêu chuẩn (không thay đổi hành vi của Neural A\*). Với việc thay thế này, Neural A\* có thể thực hiện lập kế hoạch trong *O(d)* cho trường hợp tốt nhất và *O(k)* cho trường hợp xấu nhất.



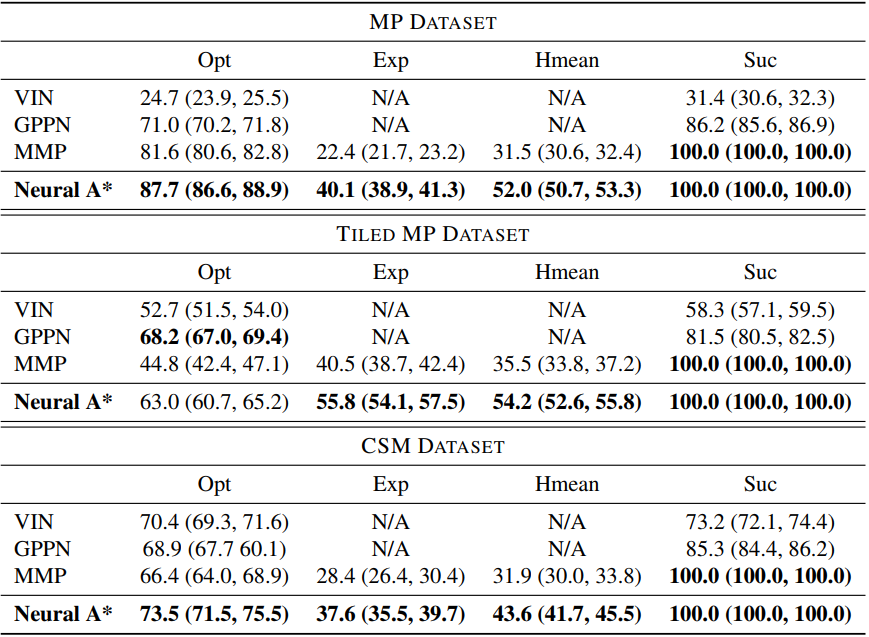
Hình 7. Kết quả định tính bổ sung (Bộ dữ liệu MP/Tiled-MP/CSM).

Để phân tích thời gian chạy của tìm kiếm một cách thực nghiệm, chúng tôi đã tạo ra ba tập hợp gồm 50 bản đồ với các kích thước 64 × 64, 128 × 128 và 256 × 256, bằng cách vẽ ngẫu nhiên các bản đồ từ bộ dữ liệu MP và ghép chúng lại. Sau đó, chúng tôi đo thời gian chạy trung bình cho mỗi bài toán bằng cách sử dụng triển khai tìm kiếm A\* tiêu chuẩn (kết hợp với 4 bản đồ hướng dẫn cho Neural A\*) và giải cùng một bài toán năm lần sau một lần thử khởi động duy nhất. Việc sử dụng các triển khai tìm kiếm A\* phức tạp hơn có thể mang lại cải thiện hiệu suất đáng kể hơn, nhưng điều này nằm ngoài phạm vi của nghiên cứu này, dù có sử dụng hay không sử dụng các bản đồ hướng dẫn của chúng tôi. Bất kể kích thước của bản đồ kiểm tra, các bản đồ hướng dẫn đều được huấn luyện bằng bộ dữ liệu Tiled MP với kích thước 64 × 64, để xem liệu mô hình của chúng tôi có tổng quát hóa tốt trên các bản đồ lớn hơn hay không. Như được hiển thị trong Bảng 8, chúng tôi xác nhận rằng Neural A\* đã giảm đáng kể thời gian chạy của tìm kiếm A\* nhờ sự hỗ trợ của các bản đồ hướng dẫn và cũng thể hiện khả năng tổng quát hóa tốt trên các bản đồ lớn hơn.

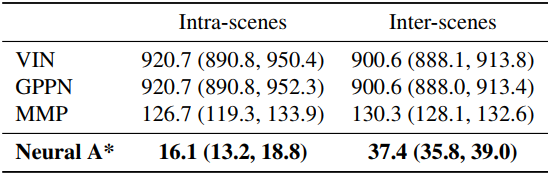


Hình 8. Kết quả định tính bổ sung (SDD).

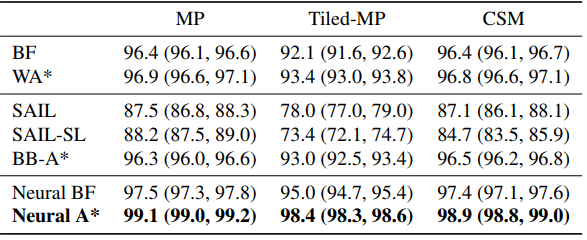
**Bảng 5. So sánh với các phương pháp học mô phỏng.** Giá trị trung bình bootstrap và khoảng tin cậy 95% của tỷ lệ tối ưu đường đi (Opt), tỷ lệ giảm số bước mở rộng nút (Exp), trung bình điều hòa (Hmean) giữa Opt và Exp, và tỷ lệ thành công (Suc).



**Bảng 6. So sánh với các phương pháp học mô phỏng trên SDD.** Giá trị trung bình bootstrap và khoảng tin cậy 95% của khoảng cách Chamfer giữa các quỹ đạo người đi bộ được dự đoán và quỹ đạo thực tế.



**Bảng 7. Đánh giá độ dài tối ưu của đường đi.** Giá trị trung bình bootstrap và khoảng tin cậy 95% của tỷ lệ giữa độ dài đường đi tối ưu và độ dài đường đi được tạo ra (càng cao càng tốt).



**Bảng 8. Đánh giá thời gian chạy của tìm kiếm.** Giá trị trung bình bootstrap và khoảng tin cậy 95% của thời gian chạy (giây) cần thiết để giải một bài toán đơn với các kích thước bản đồ khác nhau.

