DOI: https://doi.org/10.59294/HIUJS.KHTT.2026.050

PHƯƠNG PHÁP THIẾT KẾ HỆ HỖ TRỢ MINH HOẠ THUẬT GIẢI TRONG HỌC PHẦN LÝ THUYẾT ĐỒ THỊ

Trần Phan Thị Diễm Thúy, Đỗ Viết Thiện, Phan Minh Thông, Lê Gia Huy, Phan Quốc An*, Mai Trung Thành
Trường Đại học Quốc tế Hồng Bàng

TÓM TẮT
Thiết kế ứng dụng hỗ trợ học tập trong học phần lý thuyết đồ thị có vai trò quan trọng và vô cùng ý nghĩa đối với sinh viên đang theo học ngành Công nghệ thông tin. Đặc biệt là hệ thống có khả năng hỗ trợ học các thuật giải. Hiện nay có nhiều hệ thống có khả năng hỗ trợ học các thuật giải, tuy nhiên các hệ thống này cũng còn hạn chế và cần phải được cải thiện và phát triển. Bài báo này trình bày một phương pháp thiết kế hệ thống có khả năng minh họa quá trình hoạt động của thuật giải một cách từng bước. Trong đó, các vấn đề về cấu trúc hóa cho quá trình hoạt động của thuật giải, các kỹ thuật về minh họa gồm sự hiển thị lời giải dạng văn bản, trực quan hóa từng bước hoạt động của thuật giải, giọng nói cũng sẽ được trình bày. Từ cơ sở đó, bài báo cũng đã cài đặt thử nghiệm thành công, minh họa một số thuật giải trên miền tri thức Lý thuyết Đồ thị.

Từ khóa: *minh họa thuật giải, biểu diễn tri thức, hệ hỗ trợ học thuật giải*

DESIGN METHOD FOR AN ALGORITHM VISUALIZATION SUPPORT SYSTEM IN GRAPH THEORY COURSES

Tran Phan Thi Diem Thuy, Do Viet Thien, Phan Minh Thong, Le Gia Huy, Phan Quoc An, Mai Trung Thanh

ABSTRACT
Designing learning support applications for the Graph Theory course is crucial and highly meaningful for students majoring in Information Technology. This is especially true for systems capable of supporting algorithmic learning. Currently, many systems exist that support learning algorithmic solutions; however, these systems have limitations and require improvement and development. This paper presents a system designed to illustrate the step-by-step process of algorithmic problem-solving. It addresses issues related to the structuring of algorithmic processes, illustration techniques including displaying textual explanations, visualizing each step of the algorithm, and voice narration. Based on a database, the paper also successfully implemented and tested several algorithms in the Graph Theory knowledge domain.

Keywords: *algorithm visualization, knowledge representation, algorithm learning support system*

1. ĐẶT VẤN ĐỀ

Trong bối cảnh chuyển đổi số mạnh mẽ của giáo dục đại học, đặc biệt trong lĩnh vực Công nghệ thông tin và các ngành kỹ thuật, việc giảng dạy và học tập các thuật giải đóng vai trò nền tảng trong việc hình thành tư duy thuật toán, tư duy logic và năng lực giải quyết vấn đề cho sinh viên. Đối với các học phần có tính trừu tượng cao như Lý thuyết Đồ thị, sinh viên không chỉ cần nắm được kết quả cuối cùng của thuật giải mà còn cần hiểu rõ quá trình vận hành, mối quan hệ giữa các bước xử lý và sự thay đổi trạng thái của dữ liệu trong suốt quá trình thực thi. Tuy nhiên, việc tiếp cận các thuật giải

* Tác giả liên hệ: Phan Quốc An, Email: anpq2310112@student.hiu.vn
(Ngày nhận bài: 08/4/2026; Ngày nhận bản sửa: 22/4/2026; Ngày duyệt đăng: 24/4/2026)

này thông qua giáo trình truyền thống hoặc bảng phấn thường gặp nhiều khó khăn, do thiếu tính trực quan và khó hình dung các bước trung gian. Thực tiễn giảng dạy cho thấy, sinh viên thường gặp trở ngại trong việc liên kết giữa mô tả lý thuyết, mã giả và cách thuật giải thực sự hoạt động trên dữ liệu cụ thể. Do đó, nhu cầu về các hệ thống hỗ trợ minh họa thuật giải, cho phép trực quan hóa quá trình thực thi theo từng bước, đang trở nên ngày càng cấp thiết trong giáo dục kỹ thuật. Một hệ thống minh họa thuật giải hiệu quả không chỉ giúp sinh viên hiểu rõ cách thuật giải hoạt động, mà còn hỗ trợ quá trình tư duy, phân tích và tự học một cách chủ động. Từ nhu cầu thực tế đó, một hệ thống minh họa thuật giải trong giáo dục cần đáp ứng một số yêu cầu cốt lõi sau: (1) trực quan hóa rõ ràng và chính xác từng bước hoạt động của thuật giải; (2) cho phép tương tác với dữ liệu đầu vào và điều khiển tiến trình thực thi (chạy từng bước, tạm dừng, quay lại); (3) trình bày mã giả theo cấu trúc gần gũi với cách giảng dạy và chương trình đào tạo; (4) cung cấp phần diễn giải cho từng bước thực thi bằng ngôn ngữ tự nhiên, dễ hiểu; và (5) hỗ trợ đa phương thức, trong đó có sự kết hợp giữa văn bản, hoạt ảnh và giọng nói nhằm nâng cao khả năng tiếp thu kiến thức của người học.

Trong lĩnh vực hỗ trợ học thuật giải, đã có khá nhiều công trình nghiên cứu cũng như phần mềm ứng dụng được phát triển và công bố. Qua quá trình tìm hiểu tài liệu, các công trình liên quan có thể được phân theo hai dạng chính như sau.

Công trình "Exploring the Role of Visualization and Engagement in Computer Science Education" [1] đã làm rõ mối quan hệ giữa trực quan hóa (visualization) và mức độ tham gia của người học (engagement), từ đó đưa ra một luận điểm quan trọng: Việc sử dụng visualization chỉ thực sự hiệu quả khi người học tham gia một cách chủ động, thay vì chỉ quan sát thụ động.

Công trình "A Method for Designing The Intelligent System in Learning of Algorithms" [2] tác giả có đề cập đến ứng dụng hỗ trợ học thuật giải và đã có thử nghiệm trên miền tri thức về Lý thuyết đồ thị. Bài báo đã trình bày một giải pháp khoa học, theo hướng cận biểu diễn tri thức trong việc cấu trúc, mô hình hóa được thuật giải và cấu trúc hóa của từng bước giải, gọi là các task. Bài báo cũng đã triển khai thử nghiệm thiết kế ứng dụng hỗ trợ học các thuật giải trên miền tri thức lý thuyết đồ thị. Tuy nhiên trong ứng dụng chỉ quan tâm đến việc cấu trúc mô hình hóa các thuật giải, vì vậy việc minh họa này cũng bị giới hạn về tính năng cho phép nhập dữ liệu vào chương trình.

Song song với các nghiên cứu lý thuyết, một số phần mềm hỗ trợ học thuật giải đã được triển khai thực tế và sử dụng trong giảng dạy. Dưới đây là ba ứng dụng phổ biến và thường được tham chiếu nhiều nhất trong các tài liệu liên quan.

VisuAlgo [3] là ứng dụng web được xây dựng bởi Dr. Steven Halim và sinh viên Đại học Quốc gia Singapore (NUS), hiện là một trong những công cụ trực quan hóa thuật giải được dùng phổ biến nhất. Ứng dụng hỗ trợ nhiều loại thuật giải từ sắp xếp, cây nhị phân đến các bài toán trên đồ thị, với giao diện cho phép theo dõi từng bước thực thi một cách sinh động. Điểm hạn chế chính là hệ thống chưa hỗ trợ tính năng truy vấn tri thức; tại mỗi bước chạy cũng chưa hiển thị các giá trị trung gian dưới dạng văn bản mô tả, và toàn bộ quá trình minh họa chưa được tích hợp giọng đọc thuyết minh đi kèm.

Algorithm-Visualizer là nền tảng mã nguồn mở, có đặc điểm riêng là cho phép người dùng xem trực quan hóa trực tiếp từ mã C++ hoặc Java thay vì qua giao diện đồ họa trừu tượng. Điều này tạo sự liên kết tốt hơn giữa mã lập trình và hành vi thực thi của thuật giải. Tuy nhiên, giao diện còn khá phức tạp, đòi hỏi người dùng có kiến thức lập trình nhất định mới khai thác được đầy đủ. Các hạn chế về truy vấn tri thức, hiển thị giá trị văn bản và giọng đọc cũng tương tự như VisuAlgo.

Data Structure Visualizations do David Galles (Đại học San Francisco) phát triển, có giao diện bố cục rõ ràng và thân thiện hơn so với hai ứng dụng trên. Tuy vậy, toàn bộ minh họa chỉ hoạt động dựa trên bộ dữ liệu mẫu được cài sẵn, người dùng không thể tự nhập dữ liệu tùy ý. Ứng dụng cũng chưa giải quyết được các hạn chế tương tự về truy vấn tri thức, mô tả giá trị tại từng bước và hỗ trợ giọng đọc.

Công trình "Visualization Techniques for the Design and Analysis of Dynamic Programming Algorithms" [4] đề xuất kỹ thuật trực quan hóa giúp người học nhận diện các mẫu logic chung trong

bài toán quy hoạch động. Tuy nhiên, nghiên cứu chỉ tập trung vào một dạng thuật giải cụ thể, chưa áp dụng rộng rãi cho các miền tri thức khác.

Công trình "Realizing Algorithms Using GUI" [5] đã đánh giá một nền tảng trực tuyến tương tác giúp trực quan hóa thuật giải sắp xếp từ mã nguồn. Hệ thống cho phép người học điều chỉnh tốc độ và kích thước dữ liệu đầu vào. Tuy nhiên, nghiên cứu này vẫn còn hạn chế do chưa hỗ trợ giọng thuyết minh và chưa hiển thị đầy đủ các giá trị trung gian trong quá trình thực thi.

Công trình "Algorithm Visualization: The State of the Field" [6] phân tích hơn 500 hệ thống AV hiện có, chỉ ra sự phân bổ chủ đề không đồng đều và thiếu kho lưu trữ hiệu quả. Nghiên cứu cung cấp cơ sở quan trọng để định hướng xây dựng các hệ thống AV chất lượng cao hơn trong tương lai.

Công trình [7] đề xuất hệ thống truy vấn thông minh cho e-learning dựa trên mô hình ontology và đồ thị tri thức hai lớp (Rela-KG). Hệ thống đã thử nghiệm hiệu quả trong học phần Cơ sở Dữ liệu nhưng chưa tích hợp khả năng minh họa thuật giải trên miền Lý thuyết Đồ thị.

Công trình "Enhancing learning and exploratory search with concept semantics in online healthcare knowledge management systems" [8] đề xuất khung quản lý tri thức trong lĩnh vực y tế, sử dụng ngữ nghĩa khái niệm dựa trên Description Logics để hỗ trợ việc học theo định hướng nhiệm vụ. Mô hình này có thể được tham chiếu để thiết kế các hệ thống truy vấn tri thức trong giáo dục kỹ thuật.

Qua khảo sát các công trình trên, nhận thấy rằng dù là bài báo khoa học hay phần mềm ứng dụng, phần lớn các hệ thống hiện có vẫn chưa đồng thời đáp ứng ba yêu cầu cốt lõi: (1) hỗ trợ truy vấn tri thức theo ngữ cảnh, (2) hiển thị giá trị dạng văn bản tại mỗi bước thực thi thuật giải, và (3) tích hợp giọng đọc mô tả quá trình hoạt động. Đây chính là cơ sở để xác định hướng nghiên cứu và xây dựng hệ thống được trình bày trong bài báo này.

So với các hệ thống hiện có, đóng góp mới của bài báo này gồm: (1) tích hợp đồng bộ giọng nói diễn giải từng bước minh họa; (2) hỗ trợ nhập dữ liệu đầu vào tùy ý từ người dùng; (3) hiển thị lời giải dạng văn bản song song với trực quan hóa đồ thị.

2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Mô hình biểu diễn cho bài toán và thuật giải

Cơ sở tri thức trong hệ thống được dựa trên cấu trúc COKB. Với 8 thành phần đặc trưng, mô hình này cho phép biểu diễn các đối tượng tri thức một cách hệ thống, tạo cơ sở cho việc thiết kế các tiến trình suy luận và mô phỏng trực quan.

$$COKB\_G = (C, H, R, Ops, Funcs, Rules, Problems, Algorithms)$$

Trong đó:

Bộ (C, H, R, Ops, Funcs, Rules) là mô hình biểu diễn cơ sở tri thức theo COKB [2]. Với: C là tập khái niệm trong miền tri thức, mỗi khái niệm $c \in C$ là một lớp các đối tượng tính toán. Mỗi đối tượng tính toán có cấu trúc gồm bộ (Attrs, Facts, Rules), Attrs là tập thuộc tính của khái niệm c, Facts là tập các tính chất nội tại của khái niệm c, Rules là tập các luật nội tại của khái niệm c, mỗi luật r có dạng là một luật dẫn; H là tập các mối quan hệ phân cấp trên các khái niệm C; R là tập các quan hệ hai ngôi giữa các khái niệm trên C; Ops là tập tập các thành phần toán tử trên các khái niệm C; Funcs là tập các tri thức hàm; Rules là tập các định lý, tính chất, tiền đề, hệ quả;

❖ **Problems**: Là tập các lớp bài toán trong miền tri thức, với mỗi bài toán $p \in$ Problems có cấu trúc bao gồm:

Problems có cấu trúc bao gồm:

• Tên nhóm bài toán, để chỉ bài toán p thuộc nhóm bài toán nào.

• Định danh của bài toán, để phân biệt giữa các bài toán với nhau trong tập Problems.

• Thành phần diễn đạt nội dung của bài toán p. Phần diễn đạt thông tin nội dung của bài toán có cấu trúc gồm (statement, examples), trong đó: Statement là phát biểu của bài toán, examples là tập các ví dụ cụ thể cho bài toán p.

Ví dụ: Theo [2], ta xem xét bài toán duyệt đồ thị theo chiều rộng. Ta có thể đặc tả thông tin bài toán bởi cấu trúc sau:

"begin_problem
group_name ::= "duyệt đồ thị"
Id: := "duyệt đồ thị theo chiều rộng"
statement ::= "Cho đồ thị vô hướng không có trọng số G (V, E), thực hiện duyệt đồ thị G theo chiều rộng, bắt đầu từ đỉnh S"
begin_examples
begin_example_1
example_content ::= "…/exam_duyet_do_thi_1.jpg"
end_example_1
begin_examples
end_problem"

❖ **Algorithms**: Là tập các lớp thuật giải cho các lớp bài toán trong miền tri thức, mỗi thuật giải algo trong tập thuật giải có cấu trúc sau:

(Name, Content, Pseudocode, ListTasks, ListSpeech, Procedure, Visualization)

Trong đó:

(1) Thành phần tri thức Name: Tên của thuật giải.

(2) Thành phần tri thức Content: Nội dung ý tưởng của thuật giải.

(3) Thành phần tri thức Pseudocode: Là thuật giải được viết dưới dạng mã giả.

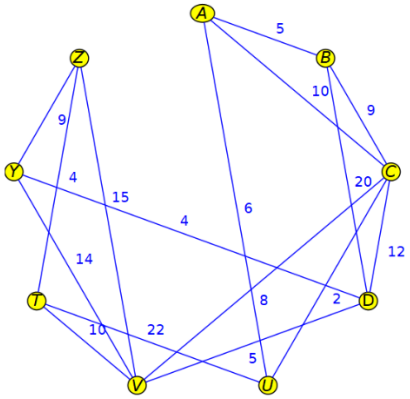
(4) Thành phần tri thức ListTasks: Là tập các bước hành động task (bước nhiệm vụ) của thuật giải, mỗi hành động task gồm phần định danh (id) và phần nội dung task.

(5) Thành phần tri thức ListSpeechs: Là tập các voice theo từng bước của ListTasks, mỗi voice gồm phần định danh (s) và phần nội dung voice.

(6)Thành phần tri thức Procedure: Là thủ tục của thuật giải được viết bằng một ngôn ngữ lập trình cụ thể, cho phép nhận các dữ liệu đầu vào (input) và cho ra (output) các kết quả đầu ra là visualization.

(7) Thành phần tri thức Visualization: Là tập các bước minh họa của thuật giải, có cấu trúc gồm: 1) tập lời giải được diễn giải bằng ngôn ngữ tự nhiên list-text-sol; 2) tập các list-task-sol đánh dấu cho quá trình vận hành để đưa ra được lời giải trong list-text-sol. 3) tập các bước ghi nhận lại các thông tin để thực hiện các hành động qua đồ họa trực quan list-graph-sol; 4) thành phần speech-sol sẽ diễn giải từng bước minh họa bằng giọng nói theo list-text-sol.

Ví dụ: Trong tri thức lý thuyết đồ thị [9, 10], cho đồ thị G có cấu trúc đỉnh cạnh như Hình 1.



Hình 1. Đồ thị vô hướng có trọng số G

Bắt đầu từ đỉnh A, trên đồ thị G hãy thực hiện tìm cây khung tối tiểu bằng thuật giải Prim. Ta có thể cấu trúc hóa thuật giải Prim với các thành phần như sau:

1) Name = "Prim"

2) Content = "Thuật toán Prim là thuật toán tìm cây khung tối tiểu trên đồ thị có hướng, hoặc vô hướng có trọng số. Thuật toán lấy ý tưởng từ việc bắt đầu từ một đỉnh bất kì, thực hiện dò tìm các đỉnh kề có thể đi và nạp chúng vào cây T, với điều kiện là trọng số của đỉnh sẽ đi là nhỏ nhất và không tạo ra chu trình. Quá trình này sẽ được lặp cho đến khi tất cả các đỉnh được nạp hết vào cây T, thì T là cây khung tối tiểu"

3) Pseudocode =“

```
Bước 1: Cây T = {}; Close = {bd};//bd là đỉnh bắt đầu
Bước 2: while Close != Vertices do
{
        2.1 Gọi T1 là tập các cạnh, có đỉnh kề với các đỉnh trong tập Close.
        2.2 Tìm cạnh có trọng số nhỏ nhất trong T1 và
        không tạo ra chu trình trong T.
        2.3. Đưa cạnh tìm được ở bước 2.2 vào cây T.
        2.4. Đưa đỉnh vừa tìm được vào tập Close.
        2.5 Loại cạnh vừa tìm được này ra khỏi tập cạnh T1.
}
Bước 3: Xuất cây T và trọng số cây T.”
```

4) ListTasks= [

```
[“id1”, “Bước 1: Cây T = {}; Close = {bd};”],
[“id2”, “Bước 2: while Close != Vertices do”],
[“id3”, “2.1 Gọi T1 là tập các cạnh, có đỉnh kề với các đỉnh trong tập Close.”],
[“id4”, “2.2 Tìm cạnh có trọng số nhỏ nhất trong T1 và không tạo ra chu trình trong T.”],
[“id5”, “2.3. Đưa cạnh tìm được ở bước 2.2 vào cây T.”],
[“id6”, “2.4. Đưa đỉnh vừa tìm được vào tập Close”],
[“id7”, “2.5 Loại cạnh vừa tìm được này ra khỏi tập cạnh T1.”],
[“id8”, “Bước 3: Xuất cây T và trọng số cây T”]
]
```

5) ListSpeechs = [

```
["s1", "Khởi tạo tập Close rỗng để lưu các đỉnh đã thuộc cây khung nhỏ nhất."],
["s2", "Chọn đỉnh bắt đầu {start} và thêm vào tập Close."],
["s3", "Khởi tạo tập Open chứa các cạnh nối từ {start} đến các đỉnh kề."],
["s4", "Chọn cạnh ({x},{y}) trong Open có trọng số nhỏ nhất."],
["s5", "Nếu {y} chưa thuộc Close thì thêm {y} vào Close và thêm cạnh ({x},{y}) vào cây khung."],
["s6", "Loại bỏ cạnh ({x},{y}) khỏi Open."],
["s7", "Thêm các cạnh nối từ {y} đến các đỉnh kề chưa thuộc Close vào Open."],
["s8", "Lặp lại cho đến khi tất cả các đỉnh đều thuộc Close."],
["s9", "Sau khi kết thúc, tập các cạnh đã chọn tạo thành cây khung nhỏ nhất."],
["s10", "Tổng trọng số cây khung nhỏ nhất là {totalWeight}."],
["s11", "Hiển thị cây khung nhỏ nhất trên đồ thị."],
["s12", "Thuật toán đã hoàn tất."]
]
```

6) Procedure được cài đặt bằng ngôn ngữ lập trình C#, với đầu vào gồm đồ thị G và đầu ra là các thành phần theo cấu trúc Visualization.

7) Cấu trúc kết quả trả về Visualization

Bảng 5. Cấu trúc của thành phần Visualization

List-Text-Sol	***list-task-sol***	***list-graph-sol***	***List-speech-sol***
Bước 1: T := {}; Close := {A};	id1	Tô đỉnh A	Speech 1.mp3
Bước 2: Close := {A} Vertices := {A, B, C, D, T, U, V, Y, Z} Ta có Close != Vertices nên vào vòng lặp	id2	-	Speech 2.mp3
Bước 2.1: Tập T1 có đỉnh kề với các đỉnh trong tập Close; T1 := {[{A, B}, 5], [{A, C}, 10], [{A, U}, 6]};	id3	-	Speech 3.mp3
Bước 2.2 T1 := {[{A, B}, 5], [{A, C}, 10], [{A, U}, 6]} Cạnh có trọng số nhỏ nhất là: {A, B}	id4	Tô cạnh A-B	Speech 4.mp3
Bước 2.3 Đưa cạnh [{A, B},5] vào trong cây T. T := [[{A, B}, 5]]	id5	-	Speech 5.mp3
Bước 2.4 Đưa tập B vào trong tập Close. Close := {A, B}	id6	-	Speech 6.mp3
Bước 2.5 Loại cạnh {A, B} khỏi T1. T1 := {[{A, C}, 10], [{A, U}, 6]}	id7	-	Speech 7.mp3
Quay lại vòng lặp. Close := {A, B} Vertices := {A, B, C, D, T, U, V, Y, Z} Ta có Close != Vertices	id2	-	Speech 8.mp3
Bước 2.1: Gọi T1 có đỉnh kề với các đỉnh trong tập Close. T1 := {[{A, C}, 10], [{A, U}, 6], [{B, C}, 9], [{B, D}, 20]}	id3	-	Speech 9.mp3
Bước 2.2 T1 := {[{A, C}, 10], [{A, U}, 6], [{B, C}, 9], [{B, D}, 20]} Cạnh có trọng số nhỏ nhất là: {A, U}	id4	Tô cạnh A-U	Speech 10.mp3

2.3 Đưa cạnh có trọng số nhỏ nhất này vào trong cây T. T := [[{A, B}, 5], [{A, U}, 6]]	id5	-	Speech 11.mp3
….	…		Speech 12.mp3
Bước 3: Ta có cây T = [[{A, B}, 5], [{A, U}, 6], [{C, U}, 2], [{C, V}, 8], [{D, V}, 5], [{D, Y}, 4], [{Y, Z}, 9], [{T, Z}, 4]] Tổng trọng số = 43	id8	Tô Cây T	Speech 13.mp3

Để hiện thực hóa việc minh họa trực quan từ các thành phần tri thức (Algorithms), hệ thống áp dụng các kỹ thuật xử lý dữ liệu đồng bộ sau:

Bóc tách công việc

Thuật giải Prim không được thực thi liên tục mà chia nhỏ thành các công việc độc lập thông qua ListTasks. Kỹ thuật này cho phép hệ thống kiểm soát luồng của thuật toán,hỗ trợ các tính năng tương tác như: Chạy từng bước (Step-by-step), tạm dừng hoặc quay lại bước trước đó theo yêu cầu của người học.

Đồng bộ hóa đa phương thức (Multi-modal Synchronization):

Đây là kỹ thuật cốt lõi để biến dữ liệu tính toán thành thông tin minh họa:

- Đồng bộ Thính giác - Thị giác: Khi một nhiệm vụ có mã id_i thực thi, hệ thống đồng thời truy xuất tệp âm thanh s_i từ **ListSpeeches** và thực hiện hiệu ứng làm nổi bật (highlight) dòng mã giả tương ứng.

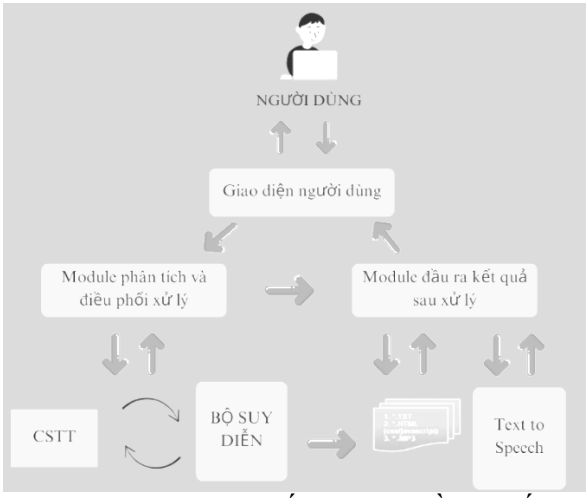
- Cập nhật trạng thái đồ họa (Graph State Update): Sử dụng thư viện D3.js để ánh xạ sự thay đổi của tập đỉnh Close và tập cạnh T lên giao diện. Khi một cạnh được nạp vào cây khung, hệ thống tự động thay đổi thuộc tính màu sắc và độ dày của cạnh đó trên đồ thị theo thời gian thực.

- Tự động sinh lời giải tự nhiên: Dựa trên các tham số của bước hiện tại, hệ thống thực hiện kỹ thuật ghép nối chuỗi để sinh ra các câu diễn giải (list-text-sol). Ví dụ: *"Tại bước này, cạnh (A, B) được chọn vì có trọng số nhỏ nhất (w=5) nối từ vùng đã xét ra ngoài"*.

3. KẾT QUẢ NGHIÊN CỨU VÀ BÀN LUẬN

3.1. Kiến trúc hệ thống

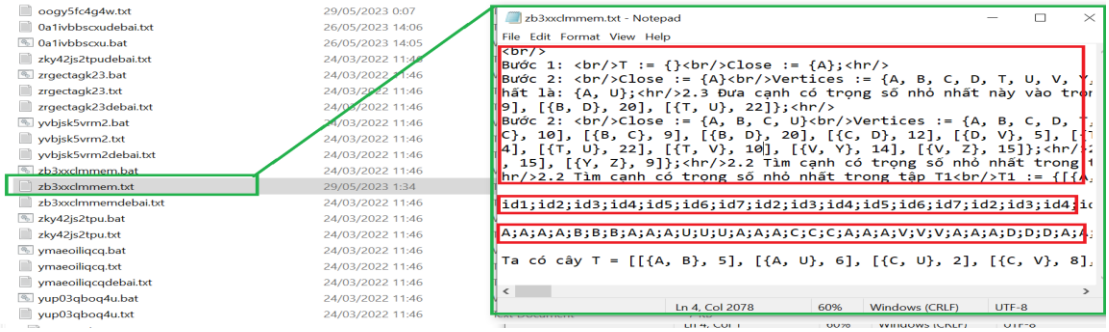
Hệ thống hỗ trợ học các thuật giải là một hệ thống được thiết kế dưới dạng môi trường Client-Server. Cho phép người dùng có thể truy cập hệ thống qua môi trường internet. Thành phần của hệ thống gồm các thành phần: Cơ sở tri thức (CSTT); Bộ suy diễn (BSD); Module xử lý các thông tin đầu vào điều khiển khác xử lý; Module xử lý việc hiển thị kết quả minh họa thuật giải. Ta có kiến trúc như hình dưới đây:



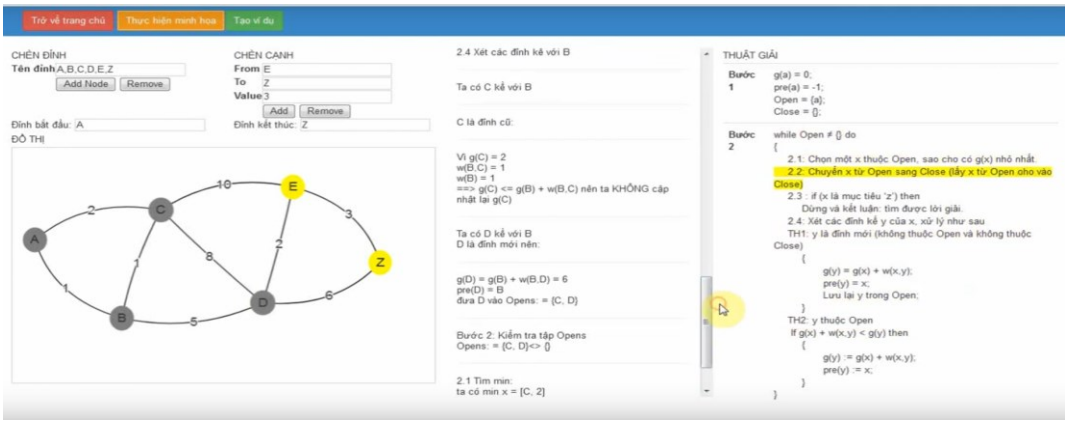
Hình 2. Hình ảnh kiến trúc sơ đồ hệ thống

3.2. Cài đặt thử nghiệm

Hệ thống được xây dựng trên công nghệ lập trình web hiện đại ASP.NET API và kết hợp với giao diện HTML CSS JS trong việc cài đặt thuật giải. Và đã cài đặt và thử nghiệm thành công một số thuật giải đồ thị như: BFS, DFS, tìm chu trình Euler, tìm đường đi Euler, Dijkstra, Prim, Kruskal.



Hình 8. Hiển thị cấu trúc nội dung của tệp lưu nội kết quả của thuật giải Prim



Hình 4. Giao diện hiển thị quá trình minh họa thuật toán tìm cây khung tối tiểu

Bảng 1. Bảng số liệu

Thuật toán	Dijkstra	BFS	DFS	Prim	Kruskal	Euler	Hamilton	Bellman
Số bài thử	10	10	10	10	10	10	10	10

3.3. Bàn luận

Trong nghiên cứu này, các ứng dụng hỗ trợ học thuật giải được đánh giá dựa trên nhóm tiêu chí sư phạm, bao gồm: Độ tin cậy và độ chính xác của nội dung diễn giải thuật giải; tính ổn định trong quá trình vận hành và hiển thị; mức độ phù hợp với chương trình đào tạo học phần Lý thuyết đồ thị; và khả năng hỗ trợ người học, thể hiện qua mức độ trực quan, tính tương tác, hỗ trợ nhập dữ liệu đầu vào và khả năng đồng bộ giữa các thành phần hiển thị. Các tiêu chí này nhằm đảm bảo hệ thống không chỉ đúng về mặt thuật toán mà còn hiệu quả trong việc hỗ trợ giảng dạy và học tập trong môi trường giáo dục đại học.

Bảng 2. So sánh các ứng dụng với kết quả của bài báo

Ứng dụng	Diễn giải	Trực quan	Nhập liệu	Speech	Đồng bộ
Hệ hỗ trợ [2]	Có hỗ trợ	Có hỗ trợ	Không hỗ trợ	Không hỗ trợ	Có hỗ trợ
Hệ đồ thị [11]	Có hỗ trợ	Có hỗ trợ	Có hỗ trợ	Không hỗ trợ	Có hỗ trợ
Gen AI	Có hỗ trợ	Không hỗ trợ	Có hỗ trợ	Không hỗ trợ	Không hỗ trợ
Web-based AV	Không xác định	Có hỗ trợ	Có hỗ trợ	Không hỗ trợ	Có hỗ trợ
Kết quả bài báo	Có hỗ trợ	Có hỗ trợ	Có hỗ trợ	Có hỗ trợ	Có hỗ trợ

4. KẾT LUẬN VÀ KIẾN NGHỊ

Bài báo đã đưa ra được một kiến trúc cho việc thiết kế các hệ thống hỗ trợ học thuật giải, bên cạnh đó bài báo cũng đã trình bày một cách rất chi tiết các vấn đề kĩ thuật, các xử lý, đặc biệt là kĩ thuật tích hợp xử lý giọng nói diễn giải trong quá trình minh họa. Dựa trên các kết quả nghiên cứu bài báo cũng đã cài đặt, thử nghiệm thành công một số thuật giải trên tri thức về Lý thuyết đồ thị như: BFS, DFS, tìm đường đi và chu trình Euler, tìm đường đi ngắn nhất Dijkstra, tìm cây khung tối tiểu Prim, Kruskal. Ứng dụng được cài đặt và thử nghiệm trên môi trường web giúp việc học tập trở nên thuận tiện và dễ dàng. Tuy cũng đã có đưa ra được kĩ thuật tích hợp thêm phần giọng nói diễn giải trong quá trình minh họa nhưng bài báo chưa có nhiều đầu tư thời gian để giải quyết các vấn đề trong giọng nói như: chưa có sự đồng bộ một cách chủ động thay vì việc diễn giải một lần cho toàn bộ quá trình. Cần phải tách ra từng bước sao cho phù hợp với cấu trúc lưu trữ lời giải list-text-sol. Hơn thế nữa, việc nghiên cứu xử lý giọng nói cho thật sự tự nhiên cũng là một vấn đề rất đáng quan tâm. Trong hướng phát triển tiếp theo, nhóm dự kiến tiến hành đánh giá hệ thống với sinh viên trong học phần Lý thuyết Đồ thị nhằm đo lường hiệu quả học tập và mức độ hài lòng của người dùng.

LỜI CẢM ƠN

Nghiên cứu này được Trường Đại học Quốc tế Hồng Bàng cấp kinh phí thực hiện dưới mã số đề tài SVTC19.76.

TÀI LIỆU THAM KHẢO

[1] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. Á. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," in *ACM SIGCSE Bull.*, vol. 35, no. 2, pp. 131–152, 2003.

[2] H. D. Nguyen, N. V. Do, T. T. Mai, and V. T. Pham, "A method for designing the intelligent system in learning of algorithms," in *Proc. 18th Int. Conf. New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT 2019)*, Seville, Spain, Sep. 2019. Amsterdam: IOS Press, pp. 303-316, 2019.

[3] S. Halim and F. Halim, "VisuAlgo: Visualising Data Structures and Algorithms through Animation," in *Proc. 19th ACM Annu. Conf. Innov. Technol. Comput. Sci. Educ. (ITiCSE 2014)*, Uppsala, Sweden, Jun. 2014. New York: ACM, pp. 261-266, 2014.

[4] S. Seng and J.-F. Serrette, "Visualization Techniques for the Design and Analysis of Dynamic Programming Algorithms," *J. Comput. Sci. Technol.*, vol. 19, no. 2, pp. 162-177, 2004.

[5] P. Agarwal, A. Sharma, and R. Singh, "Realizing Algorithms Using GUI," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 7, pp. 2365-2370, 2010.

[6] S. H. Rodger and T. W. Finley, "Algorithm Visualization: The State of the Field," *ACM Trans. Comput. Educ.*, vol. 2, no. 2, pp. 1-22, Jun. 2002.

[7] N. V. Do, T. T. Mai, and L. N. Hoang, "A Knowledge-Based Model for Designing the Knowledge Querying System in Education," in *Proc. 2022 RIVF Int. Conf. Comput. Commun. Technol. (RIVF)*, Hanoi, Vietnam, Dec. 2022. Piscataway: IEEE, pp. 1-6, 2022.

[8] W. H. Hsu, J. R. Bodily, C.-Y. Fang, and K. Mehrotra, "Enhancing learning and exploratory search with concept semantics in online healthcare knowledge management systems," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3059-3069, Apr. 2010.

[9] R. J. Trudeau, *Introduction to Graph Theory*. New York, NY, USA: Courier Corporation, 1993.

[10] N. Cam và C. Đ. Khánh, *Lý Thuyết Đồ Thị*. TP. Hồ Chí Minh, Việt Nam: NXB Đại học Quốc gia TP. HCM, 2008.

[11] L. H. Thái, "Ứng dụng web hỗ trợ học Lý thuyết đồ thị," Luận văn Thạc sĩ, Trường Đại học Công nghệ Thông tin, ĐHQG-HCM, TP. Hồ Chí Minh, 2020.