

KHAI THÁC TẬP PHỔ BIẾN TỪ DỮ LIỆU LƯỖNG BẰNG CÁCH SỬ DỤNG THUẬT TOÁN DI TRUYỀN

Phạm Đức Thành¹, Lê Thị Minh Nguyễn¹

¹ Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ - Tin học TP.HCM

thanhpd@huflit.edu.vn, nguyentlm@huflit.edu.vn

TÓM TẮT - Bài báo này trình bày một nghiên cứu về việc khai thác các tập phổ biến từ dữ liệu giao dịch luồng trong bối cảnh có sự thay đổi khái niệm. Dữ liệu luồng, với tính chất không ổn định, đặt ra nhiều thách thức trong việc khai thác. Bài báo này sử dụng phương pháp thuật toán di truyền, mối quan hệ giữa sự thay đổi khái niệm, kích thước cửa sổ trượt và ràng buộc của thuật toán di truyền. Sự thay đổi khái niệm được xác định thông qua sự thay đổi trong các tập phổ biến. Sự độc đáo của đề tài này nằm ở việc xác định sự thay đổi khái niệm bằng cách sử dụng các tập phổ biến để khai thác dữ liệu luồng, với việc sử dụng khung là thuật toán di truyền. Công thức được trình bày để tính toán số lần của độ hỗ trợ tối thiểu trong dữ liệu luồng bằng cách sử dụng cửa sổ trượt. Việc thử nghiệm đã chỉ ra rằng tỷ lệ giữa kích thước cửa sổ và số giao dịch cho mỗi thay đổi là yếu tố quan trọng để đạt hiệu suất tốt. Việc đạt được kết quả tốt khi kích thước cửa sổ trượt quá nhỏ là một thách thức vì những biến động bình thường trong dữ liệu có thể xuất hiện như là sự thay đổi khái niệm. Kích thước cửa sổ phải được quản lý cùng với giá trị của độ hỗ trợ và độ tin cậy để đạt được kết quả hợp lý. Phương pháp này phát hiện sự thay đổi khái niệm đã hoạt động tốt khi sử dụng kích thước cửa sổ lớn hơn.

Từ khóa — Tập phổ biến, Dữ liệu luồng, Thay đổi khái niệm, Cửa sổ trượt, Khai thác luật kết hợp, Thuật toán di truyền, Khai thác dữ liệu, Số giao dịch mỗi lần trượt (batch).

I. GIỚI THIỆU

Trong thế giới kỹ thuật số ngày nay, dữ liệu liên tục được tạo ra từ các cảm biến giao thông, cảm biến sức khỏe, giao dịch khách hàng và các thiết bị Internet of Things (IoT) khác. Luồng dữ liệu liên tục và không ngừng tạo ra những thách thức mới từ góc nhìn của việc khai thác dữ liệu. Việc khai thác dữ liệu tĩnh theo từng khoảng khắc thời gian không còn hữu ích nữa. Dữ liệu luồng, với tính chất động hoặc không ổn định, có các mẫu thay đổi theo thời gian, và đây được gọi chính xác hơn là sự thay đổi khái niệm. Các thuật toán phát triển để khai thác dữ liệu luồng phải có khả năng phát hiện và làm việc với sự thay đổi khái niệm, do đó cần có các phương pháp mới để khai thác dữ liệu luồng. Bài báo này tìm hiểu về một kỹ thuật khai thác dữ liệu quan trọng, khai thác tập phổ biến, được áp dụng cho dữ liệu giao dịch luồng, trong bối cảnh có sự thay đổi khái niệm.

Khai thác tập phổ biến, một công đoạn tiền đề cho khai thác luật kết hợp, thường đòi hỏi sức mạnh xử lý đáng kể vì quá trình này liên quan đến nhiều lần xử lý dữ liệu trong cơ sở dữ liệu (CSDL), và điều này có thể là một thách thức trong các tập dữ liệu luồng lớn. Mặc dù đã có nhiều tiến bộ trong việc tìm kiếm tập phổ biến và luật kết hợp trong CSDL tĩnh hoặc cố định [1] [2], vì cảnh quan dữ liệu đang thay đổi, các thuật toán phải được mở rộng hiệu quả để khai thác dữ liệu luồng với sự thay đổi khái niệm. Có một số vấn đề được đặt ra để khám phá điều này: (i) Có thể khai thác tập phổ biến trong khi dữ liệu luồng xuất hiện mà không cần tham chiếu đến một CSDL chính cố định; (ii) Có các kỹ thuật để giảm độ phức tạp về thời gian của việc khai thác tập phổ biến để quản lý dữ liệu động; (iii) Có những thách thức nào xuất hiện do môi trường luồng dữ liệu? Để giải quyết vấn đề độ phức tạp thời gian, việc sử dụng thuật toán di truyền đã được đề cập; kỹ thuật này đã được chứng minh giảm độ phức tạp thời gian của việc khai thác tập phổ biến và luật kết hợp trong CSDL tĩnh [2].

Sử dụng thuật toán di truyền để khai thác tập phổ biến trong dữ liệu luồng với sự thay đổi khái niệm là một lĩnh vực nghiên cứu được khám phá tương đối ít, và có nhiều vấn đề quan trọng cần được xem xét. Một vấn đề quan trọng cụ thể được khám phá trong đề tài này là mối quan hệ giữa tốc độ thay đổi khái niệm trong dữ liệu luồng và sự hội tụ của thuật toán di truyền được sử dụng để khai thác tập phổ biến từ dữ liệu đó.

Trong nghiên cứu này, thuật toán di truyền được sử dụng để khai thác tập phổ biến từ dữ liệu luồng trong bối cảnh có sự thay đổi khái niệm, với sự hỗ trợ của cửa sổ trượt. Điểm độc đáo của bài báo này nằm ở việc xác định sự thay đổi khái niệm bằng cách sử dụng các tập phổ biến để khai thác dữ liệu luồng, với việc sử dụng khung thuật toán di truyền.

Việc thử nghiệm được thực hiện bằng cách sử dụng một chương trình kiểm tra, cho phép sử dụng dữ liệu luồng theo yêu cầu. Chương trình kiểm tra cũng cho phép thay đổi một số biến cơ bản, bao gồm dữ liệu được truyền, số giao dịch trước khi có sự thay đổi khái niệm được giới thiệu và kích thước của cửa sổ trượt. Các biến

¹ Coressponding Author

khác, liên quan đến thuật toán di truyền, bao gồm kích thước quần thể, xác suất lai ghép, xác suất đột biến của thuật toán di truyền.

Phần còn lại của bài báo được trình bày như sau. Phần hai trình bày về các công trình liên quan. Để cung cấp nền tảng và ngữ cảnh, trong phần 3 trình bày các khái niệm, trình bày cách nhìn tổng quan về dữ liệu luồng trong cửa sổ trượt, thay đổi khái niệm, tập phổ biến trong cửa sổ trượt và cuối cùng là thuật toán di truyền. Phần 4 trình bày phương pháp đề xuất. Tiếp theo, trình bày kết quả thực nghiệm về phương pháp đề xuất khác biệt so với thuật toán T_Apriori chuẩn [3]. Cuối cùng, trình bày phần kết luận và tương lai của đề tài.

II. CÔNG TRÌNH LIÊN QUAN

Việc khai thác luật kết hợp và tập phổ biến đã có một bề dày lịch sử và thực tế [4] [5] [6] đã thảo luận về các thuật toán để thực hiện các nhiệm vụ này trong bối cảnh dữ liệu luồng. Mô hình tích lũy, mô hình cửa sổ trượt và mô hình tích lũy có trọng số được trình bày bởi Yu và Chi [6] như các cách để xử lý dữ liệu luồng. Mô hình tích lũy và mô hình tích lũy có trọng số giữ tất cả dữ liệu trong một kho dữ liệu. Tuy nhiên, trong mô hình tích lũy có trọng số, dữ liệu trở nên ít quan trọng hơn khi nó càng cũ. Nghiên cứu của Krempel và đồng sự [7] trình bày về các thách thức đối mặt với loại nghiên cứu này là thiếu một tập hợp các môi trường, giao thức và phương pháp đánh giá chuẩn chung cần thiết để kiểm tra và so sánh. Trong ngữ cảnh khai thác tập phổ biến, Gama [8] cho biết rằng thay đổi khái niệm là vấn đề khó nhất. Hoens và đồng sự [9] trình bày một phương pháp phát hiện và thích ứng với thay đổi khái niệm trong quá trình học phân loại. Kim và Park [10], Wang và Abraham [11] cũng đề cập đến việc phát hiện thay đổi khái niệm.

Bull [12] cho rằng thuật toán di truyền là một trong những thuật toán tối ưu hóa hữu ích nhất, nhưng chỉ khi hàm được tối ưu hóa là đơn giản để tính toán. Ngược lại, Rabinovich và Wigderson [13] tin rằng chúng có thể được sử dụng trong các vấn đề thực tế khó khăn. Có nhiều nghiên cứu về tốc độ hội tụ của thuật toán di truyền. Forrest và Mitchell [14], Ruholla và Smith [15], Aldallal [16], Lin và đồng sự [17], Angelova và Pencheva [18], Pellerin và đồng sự [19], đều xem xét cách cải thiện tốc độ hội tụ trong thuật toán di truyền. Những nghiên cứu này ảnh hưởng đến tốc độ hội tụ bằng cách điều chỉnh các toán tử di truyền. Theo Aldallal [16], chỉ số phổ biến nhất của sự hội tụ trong thuật toán di truyền là số thế hệ cần thiết để một cá thể đáp ứng yêu cầu về độ thích nghi. Tuy nhiên, He và Lin [20] coi "trung bình hình học chuẩn hóa của tỷ lệ giảm khác biệt về độ thích nghi mỗi thế hệ" là một chỉ số tốt hơn mà vẫn dễ tính toán. Ghosh và đồng sự [2] và Rangaswamy và Shobha [21] thảo luận về khai thác tập phổ biến và luật kết hợp bằng thuật toán di truyền.

Trong khi tất cả các thành phần liên quan đến nghiên cứu này đã được đề cập độc lập, không có nghiên cứu nào trực tiếp giải quyết tất cả các vấn đề này cùng nhau. Nghĩa là không có công trình nào giải quyết vấn đề khai thác tập phổ biến từ dữ liệu luồng với thuật toán di truyền trong bối cảnh biến đổi khái niệm, đó chính là vấn đề được giải quyết trong đề tài này.

III. ĐỊNH NGHĨA VÀ KHÁI NIỆM

A. ĐỊNH NGHĨA

Cho $I = \{a_1, a_2, \dots, a_m\}$ là một tập các mục, và CSDL giao dịch $DB = \langle T_1, T_2, \dots, T_n \rangle$, trong đó T_i (i thuộc $[1..n]$) là một giao dịch chứa một tập các mục trong I .

Định nghĩa 1. Độ hỗ trợ được định nghĩa là các hạng mục được tìm thấy với tần suất tối thiểu trong toàn bộ tập dữ liệu. Độ hỗ trợ (hoặc tần suất xuất hiện) của một mẫu A , trong đó A là một tập con của I , là số lượng giao dịch chứa A trong DB . Công thức (1) tính độ *support* như sau:

$$support(A) = \frac{n(A)}{|DB|} \quad (1)$$

Trong đó: $n(A)$ là số lượng giao dịch chứa A trong DB

Định nghĩa 2. Một tập phổ biến là một tập mục đáp ứng ngưỡng hỗ trợ tối thiểu. Một mẫu A được coi là phổ biến nếu độ hỗ trợ của A không nhỏ hơn một ngưỡng hỗ trợ tối thiểu min_sup được xác định trước, ξ . Công thức (2) tính tập phổ biến như sau:

$$support(A) \geq \xi \times |DB| \quad (2)$$

Định nghĩa 3. Một luật kết hợp là một câu lệnh có điều kiện nói rằng, nếu hạng mục A tồn tại trong giao dịch thì có khả năng hạng mục B cũng có trong giao dịch đó. Một mẫu $A = \{X, Y\}$, ta có luật kết hợp $X \rightarrow Y$. Độ tin cậy được định nghĩa là tần suất của một hạng mục liên quan đến tập mục chứa các hạng mục được hỗ trợ. Độ tin cậy (confidence) là xác suất xảy ra Y khi đã biết X. Công thức (3) được tính như sau:

$$confidence(X \rightarrow Y) = P(Y|X) = \frac{n(X \cup Y)}{n(X)} \tag{3}$$

Trong đó: $n(X \cup Y)$ là số giao dịch chứa $(X \cup Y)$ và $n(X)$ là số giao dịch chứa X

Định nghĩa 4. Ta có một tập mẫu A là tập con của I. Đối với dữ liệu luồng, sử dụng cửa sổ trượt, với kích thước cửa sổ trượt là win_size, ta có **support_count** để xác định tập phổ biến với công thức tính (4) là:

$$support_count(A) = (win_size * support(A)) \tag{4}$$

Giá trị support_count của thuật toán di truyền được tính theo công thức (5) như sau:

$$support_count_GA(A) = (win_size * support(A) * confidence(A)) \tag{5}$$

B. KHÁI NIỆM

1. DỮ LIỆU LUỒNG TRONG CỬA SỔ TRƯỢT

Trong hầu hết môi trường dữ liệu hiện đại, một kho lưu trữ dữ liệu tĩnh như CSDL được cho là có sẵn, nhưng điều này có thể không phải luôn luôn đúng. Dữ liệu có thể liên tục và thường xuyên được truyền đến máy tính. Hơn nữa, sự truyền đến của các dữ liệu này là một phần tất yếu trong mô hình kinh doanh, giống như tình huống khi các tàu đưa hàng hóa đến một cảng tại một thành phố. Nếu cảng cố gắng lưu trữ tất cả hàng hóa, không gian lưu trữ có thể nhanh chóng trở thành vấn đề đáng quan tâm. Ngoài ra, trong khi một số hàng hóa thường được lưu trữ, nó phải được xử lý và loại bỏ nhanh chóng với tốc độ trung bình ít nhất bằng tốc độ hàng hóa đến. Hệ thống máy tính xử lý sự đến liên tục này đối mặt với một vấn đề tương tự về việc đến, gồm có không gian lưu trữ và xử lý.

Việc xử lý dữ liệu luồng thường liên quan đến việc trích xuất một số thông tin hữu ích từ dữ liệu. Có một số ràng buộc đặc biệt cần được xem xét khi làm việc với dữ liệu luồng, ví dụ như dữ liệu có thể bị giới hạn chỉ cho truy cập một lần, dữ liệu có thể không giới hạn về khối lượng và phản hồi thời gian thực có thể cần thiết để dữ liệu trở nên hữu ích [6] [7] [8] [22]. Các công trình này thảo luận về các mô hình xử lý dữ liệu luồng, trong đó một trong số đó là mô hình cửa sổ trượt. Cửa sổ trượt giữ thông tin về các giao dịch hiện tại, cho phép giao dịch cũ nhất bị xóa khi giao dịch mới đến. Do đó, n giao dịch mới nhất luôn xuất hiện trong cửa sổ trượt, trong đó n là kích thước cửa sổ trượt.

Bảng 1 Cửa sổ trượt

Luồng giao dịch		
Những giao dịch chưa vào hệ thống	-	
	-	
	-	
	{A, F, H}	
	{B, C}	
Giao dịch mới nhất trong cửa sổ trượt	{A, D, E, I}	Cửa sổ trượt Kích thước = 5 giao dịch
	{B, F}	
	{A, B, C, G}	
	{A, G, I}	
Giao dịch cũ nhất trong cửa sổ trượt	{C, D}	
Những giao dịch đã xóa, đã đi qua hệ thống	{B, D, E}	
	{A, C, D, G, I}	
	-	
	-	

	-	
--	---	--

Bảng 1 minh họa một cửa sổ trượt có kích thước là 5. Cột ở giữa chứa các giao dịch của dữ liệu luồng. {Hạng mục B, Hạng mục D, Hạng mục E} không còn trong cửa sổ. Dữ liệu này đã bị xóa. Khu vực được đánh khung là cửa sổ hiện tại. Các giao dịch mới là hai giao dịch ở trên cửa sổ. Chúng chưa được đưa vào trong cửa sổ. Khi {Hạng mục B, Hạng mục C} được đưa vào cửa sổ, {Hạng mục C, Hạng mục D} sẽ bị loại bỏ.

2. SỐ GIAO DỊCH MỖI LẦN TRƯỢT (LÔ - BATCH)

Là số lượng giao dịch trong mỗi lần mà cửa sổ trượt, được gọi là lô (batch). Bảng 2 minh họa một batch, mỗi batch gồm hai giao dịch. Khi cửa sổ trượt mỗi lần 1 batch, nghĩa là sẽ gồm 2 giao dịch được đưa vào hệ thống xử lý, đồng thời sẽ có hai giao dịch sẽ bị đưa ra khỏi hệ thống

Bảng 2 Minh họa một batch gồm hai giao dịch

Luồng giao dịch		
Những giao dịch chưa vào hệ thống	-	
	-	
	-	
	{A, F, H}	Chung một batch (gồm 2 giao dịch)
	{B, C}	
Batch mới nhất trong cửa sổ trượt	{A, D, E, I}	Cửa sổ trượt Kích thước = 5 giao dịch
	{B, F}	
	{A, B, C, G}	
Batch cũ nhất trong cửa sổ trượt	{A, G, I}	
	{C, D}	
Những giao dịch đã xóa, đã đi qua hệ thống	{B, D, E}	
	{A, C, D, G, I}	
	-	
	-	
	-	

3. SỰ THAY ĐỔI KHÁI NIỆM

Khai thác thông tin từ dữ liệu tĩnh dẫn đến thông tin tĩnh. Trong dữ liệu luồng, thông tin đã học không tĩnh. Nó là động. Khi bản chất của thông tin trong dữ liệu thay đổi theo thời gian, điều này được gọi là sự thay đổi khái niệm. Nó cũng quan trọng để nắm bắt sự thay đổi của một khái niệm. Trong bài báo này, sự thay đổi khái niệm được đo bằng cách sử dụng tập phổ biến, thông qua sự thay đổi trong tập phổ biến. Các công trình khác cũng đã xem xét việc sử dụng tập phổ biến cho việc khai thác dữ liệu luồng [8]. Gama [8] nhắc đến rằng vấn đề khó nhất trong việc khai thác tập phổ biến từ dữ liệu luồng là các tập ít phổ biến trong quá khứ có thể trở nên phổ biến, và các tập phổ biến trong quá khứ có thể trở nên ít phổ biến. Do đó, ngoài việc phát hiện sự thay đổi khái niệm, quản lý các thay đổi khái niệm cũng là một khía cạnh quan trọng của việc khai thác dữ liệu luồng.

4. TẬP PHỔ BIẾN TRONG CỬA SỔ TRƯỢT.

Trong dữ liệu luồng với việc sử dụng cửa sổ trượt, vấn đề cần giải quyết là tìm các tập phổ biến. Trước khi trình bày các công thức được sử dụng để tính toán số lượng hỗ trợ trong các cửa sổ trượt, ta xem xét về các thông tin cơ bản về thuật toán Apriori chung.

Ví dụ: Giả sử ta có CSDL giao dịch như bảng 3 sau:

Bảng 3 Cơ sở dữ liệu giao dịch

Tid	Items
-----	-------

T ₁	A, F, H
T ₂	A, B, C
T ₃	A, D, E, I
T ₄	B, F
T ₅	A, B, C, G
T ₆	A, B, G, I
T ₇	C, D
T ₈	B, D, E
T ₉	A, C, D, G, I

Cho độ dài cửa sổ trượt $win_size=6$ (từ giao dịch T_1 đến T_6), độ hỗ trợ tối thiểu $min_sup=0.3$, độ tin cậy tối thiểu $min_conf=0.6$. Ta nhận thấy có 4 giao dịch chứa (A), do đó độ hỗ trợ của A là 5/6. Có 4 giao dịch chứa (B), do đó độ hỗ trợ của B là 4/6. A và B xuất hiện cùng nhau trong một giao dịch 3 lần, như vậy:

- Độ hỗ trợ $(A \Rightarrow B) = 3/6$ hoặc 0.5.
- Độ tin cậy $(A \Rightarrow B) = P(B|A) = (\text{số lần xuất hiện } (A \cup B)) / (\text{số lần xuất hiện } (A)) = 3/5$ hoặc 0.6.

Độ tin cậy $(A \Rightarrow B) >$ độ tin cậy tối thiểu (0.6), vì vậy đây là một luật kết hợp mạnh. Vì vậy, đối với dữ liệu tĩnh, một tập phổ biến sẽ là bất kỳ tập mục nào có độ hỗ trợ > 0.3 .

Đối với dữ liệu luồng, sử dụng cửa sổ trượt, ta có $support_count_slide$ để xác định tập phổ biến theo công thức (4). Giả sử $X= \{AB\}$ là tập phổ biến. Ta có, tập phổ biến X trong cửa sổ trượt sẽ được tính theo công thức (6) như sau:

$$support_count_slide(X) > (win_size * \xi)$$

(6)

Công thức (7) tính giá trị $support_count$ của thuật toán di truyền sẽ là:

$$support_count_slide_GA(X) = (win_size * support_count_slide(X) * confidence(X))$$

(7)

Vì vậy, tập mục $(A \cup B)$ sẽ được coi là một tập phổ biến của thuật toán di truyền theo công thức (8):

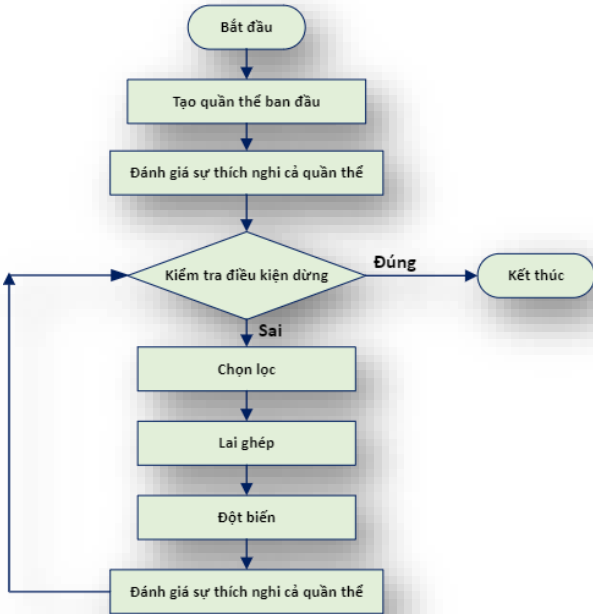
$$support(A \Rightarrow B) = (win_size * support_count_slide(A \Rightarrow B) * confidence(A \Rightarrow B))$$

(8)

Độ đo (*kích thước cửa sổ trượt * độ hỗ trợ * độ tin cậy*) cũng được sử dụng trong hàm đánh giá thích nghi của thuật toán di truyền.

5. THUẬT TOÁN DI TRUYỀN

Thuật toán di truyền mô phỏng quá trình tự nhiên chọn lọc. Đầu tiên, cấu trúc gen của một cá thể được định nghĩa. Chúng được biểu diễn dưới dạng một danh sách các số 0 và 1. Sau đó, các cá thể được đánh giá về độ thích nghi bằng cách sử dụng một hàm thích nghi. Sau đó, các cá thể tốt nhất được kết hợp lại, tức là một số gen của



Hình 1 Thuật toán Di truyền

chúng được hoán đổi. Tiếp theo, các cá thể trải qua quá trình đột biến, tức là một hoặc nhiều gen của chúng có thể được đảo ngẫu nhiên từ 1 thành 0 hoặc ngược lại. Khi các toán tử di truyền đã được sử dụng để xác định một quần thể mới, hàm thích nghi được áp dụng lại. Quá trình này lặp lại cho đến khi tìm thấy một cá thể thành công hoặc đạt đến số lượng thế hệ tối đa.

- Bước 1.** Tạo ra một quần thể ngẫu nhiên của cá thể.
- Bước 2.** Đánh giá độ thích nghi của mỗi cá thể bằng cách sử dụng một hàm thích nghi.

Bước 3. Áp dụng phép lai ghép dựa trên xác suất lai ghép, và cá thể thích nghi hơn có cơ hội cao hơn được chọn để lai ghép.

Bước 4. Điều này tạo ra một quần thể mới.

Bước 5. Quần thể mới này được tiến hành đột biến, với xác suất đột biến của mỗi gen được xác định bởi xác suất đột biến, thường là một xác suất rất thấp.

Bước 6. Đánh giá độ thích nghi của quần thể mới. Nếu độ thích nghi của cá thể tốt nhất chưa đạt đến một mức mong muốn, và số thế hệ tối đa chưa đạt được, thì quá trình bắt đầu lại từ bước 3.

a) Cá thể và gen

Trong cài đặt này, các cá thể được tạo thành từ một danh sách các giá trị 0 và 1, tương ứng với danh sách được sắp xếp của các mục phổ biến có thể có, ví dụ:

Tập phổ biến: [táo, chuối, kẹo, tã, xà phòng]

Cá thể ngẫu nhiên: [0,1,1,0,1]

Đưa ra một tập phổ biến tiềm năng bao gồm {chuối, kẹo, xà phòng}.

b) Kích thước quần thể

Để tìm nhiều tập mục đủ thay vì chỉ một tập mục tốt nhất, việc duy trì một quần thể các cá thể duy nhất là quan trọng. Do đó, kích thước quần thể được dựa trên độ dài của tập mục tiềm năng hoặc một kích thước tối đa nào đó. Quy tắc được sử dụng cho kích thước của quần thể là:

$minimum(2^{độ\ dài\ gen} - 1, 100)$

Điều này giữ cho kích thước quần thể nhỏ trong khi đảm bảo sự duy nhất của các cá thể trong toàn bộ quần thể.

c) Hàm thích nghi

Để xác định độ thích nghi của một cá thể, hàm thích nghi sẽ lấy các mục được ánh xạ với giá trị 1 trong cá thể, sau đó lặp qua một bản sao của cửa sổ trượt để kiểm tra xem tập mục của cá thể có là một phần con của từng giao dịch hay không và đếm số lần xảy ra điều này. Hàm thích nghi cuối cùng là: *đếm số lượng giao dịch mà có chứa cá thể đang xử lý theo cách tính sau:*

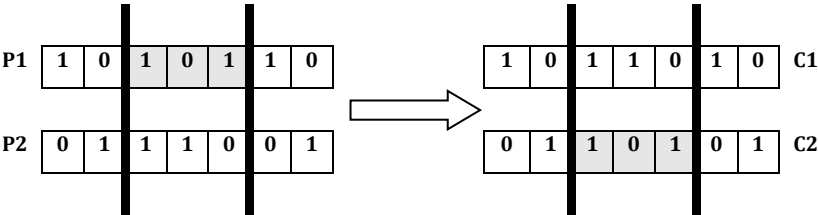
$(kích\ thước\ cửa\ sổ\ trượt * độ\ hỗ\ trợ * độ\ tin\ cậy)$

d) Lựa chọn cá thể ứng viên cho thế hệ tiếp theo

Lựa chọn cá thể ứng viên cho thế hệ tiếp theo: chọn cá thể có độ thích nghi đạt ngưỡng *support_count* và đưa nó vào quần thể mới. Điều này được thực hiện mà không xóa các cá thể ban đầu khỏi quần thể cũ.

e) Lai ghép hai điểm

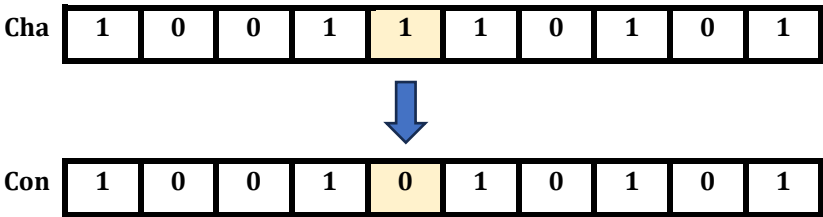
Sau khi chọn các cá thể cho lai ghép, chúng được tiến hành phép lai ghép hai điểm. Điều này bao gồm việc tìm một phần giữa ngẫu nhiên từ hai cá thể và trao đổi hai phần đó. Hai số ngẫu nhiên được chọn, số đầu tiên lớn hơn chỉ số 0 và sau đó từ chỉ số đó đến chỉ số trước cuối cùng trong danh sách. Khi tìm thấy phần giữa, nó có thể được trao đổi với cùng một phần từ cá thể khác (xem Hình 2).



Hình 2 Lai ghép hai điểm

f) Đột biến

Phép toán di truyền cuối cùng được áp dụng trước khi đánh giá lại độ thích nghi của mỗi cá thể là phép toán đột biến (xem Hình 3). Thông thường, một số ngẫu nhiên được tạo ra cho mỗi bit trong gen và nếu số ngẫu nhiên nhỏ hơn ngưỡng đột biến, thì bit đó sẽ bị đảo ngược. Xác suất đột biến thường rất thấp, tức là dưới 5%. Tuy nhiên, do yêu cầu về tính duy nhất, phép toán đột biến được sử dụng để loại bỏ các bản trùng nhau trong quần thể. Mỗi cá thể mới được thêm vào quần thể sẽ được chuyển thành một chuỗi và nếu chuỗi đó đã có trong tập hợp tạm thời các chuỗi, cá thể sẽ bị đột biến lại. Mặc dù không giữ lại dữ liệu về tỷ lệ đột biến, kiểm tra từng điểm đã cho thấy điều này dẫn đến một tỷ lệ đột biến không thường xuyên cao.



Hình 3 Đột biến

IV. KHAI THÁC CÁC TẬP PHỔ BIẾN DỮ LIỆU LUỒNG VỚI THUẬT GIẢI DI TRUYỀN

A. THUẬT TOÁN ĐỀ XUẤT:

1. THUẬT TOÁN GA.

Algorithm 1 genetic algorithm

Input Số thế hệ *generations*, kích thước quần thể *population_size*, bộ dữ liệu vào *dataset*, độ hỗ trợ *support_count*.

Output Các tập phổ biến

```
1. item_candidate ← Candidate(dataset, support_count)
2. population ← CreatePopulation_BitArr(population_size, len(item_candidate))
3. for generation ← 0 to generations do
4.     fitness_scores ← [fitness_function(dataset, individual, item_candidate) for individual in population]
5.     selected_parents ← Select(population, fitness_scores, support_count)
6.     next_generation ← CopyPopulation(selected_parents)
7.     while len(next_generation) < population_size do
8.         parent1, parent2 ← randomSelected(selected_parents)
9.         if random.random() < crossover_probability then
10.            child1, child2 ← crossover(parent1, parent2)
11.            next_generation.append(child1)
12.            next_generation.append(child2)
13.         end if
14.     end while
15.     for individual in next_generation do
16.         if random.random() < mutation_probability then
17.             mutate(individual, next_generation)
18.         end if
19.     end for
20.     population ← CopyPopulation(next_generation)
21. end for
22. frequent_itemsets ← find_frequent_itemsets(population, dataset, support_count, item_candidate)
23. Return frequent_itemsets
```

- Dòng 1: Đi tìm tập các ứng viên item_candidate dựa trên dữ liệu dataset đầu vào và support_count.
- Dòng 2: Tạo quần thể với BitArray với kích thước quần thể population_size và độ dài của item_candidate.
- Dòng 3: Lặp số lượng thế hệ là generations

Dòng 4: Tính độ thích nghi của từng cá thể trong quần thể population thông qua hàm thích nghi.

Dòng 5: Lựa cho các cá thể parent để lai ghép dựa trên độ thích nghi và support_count.

Dòng 6: Sao chép các cá thể vừa chọn vào thế hệ tiếp theo.

Dòng 7: Lặp cho đến khi số lượng cá thể trong thế hệ tiếp theo bằng với kích thước quần thể.

Dòng 8: Lựa cho ngẫu nhiên cặp cá thể cha mẹ.

Dòng 9: Nếu vượt ngưỡng xác suất của lai ghép crossover_probability thì.

Dòng 10: Lai ghép thành 2 cá thể con.

Dòng 11, 12: Đưa 2 cá thể con vào quần thể kế tiếp.

Dòng 15: Duyệt từng cá thể trong quần thể kế tiếp.

Dòng 16, 17: Nếu vượt ngưỡng xác suất đột biến thì sẽ đột biến.

Dòng 20: Sao chép quần thể cho lần lặp tiếp theo.

Dòng 22: Lưu lại những tập phổ biến kết quả của thuật toán 1

Dòng 23: Trả về kết quả tìm được thuật toán.

2. THUẬT TOÁN GA_BSW (GENETIC ALGORITHM WITH BATCH SLIDING WINDOW).

Algorithm 2 GA_BSW

Input Dữ liệu *transactions*, kích thước quần thể *population_size*, kích thước cửa sổ trượt *window_size*, độ hỗ trợ *support_count*, số giao dịch trong một lô *batch*.

Output Danh sách lưu các tập phổ biến của mỗi lần trượt

```
1. all_frequent_itemsets ← [ ]
2. i ← 0
3. while i < len(transactions) - window_size + 1 do
4.     window_data ← transactions[i:i+window_size]
5.     frequent_itemsets ← GA(generations, population_size, window_data, support_count)
6.     all_frequent_itemsets.extend(frequent_itemsets)
7.     i ← i + batch
8. end while
9. Return all_frequent_itemsets
```

Dòng 1: Khởi tạo danh sách tất cả các tập phổ biến là rỗng.

Dòng 2, 3: Thực hiện việc dùng cửa sổ trượt với *window_size*, thực hiện trượt cho đến khi hết dữ liệu *transactions*.

Dòng 4: Lấy ra các giao dịch trong cửa sổ trượt *window_data*.

Dòng 5: Gọi hàm GA (thuật toán 1) với tham số là số lượng thế hệ *generations*, kích thước quần thể *population_size*, dữ liệu trong cửa sổ trượt *window_data* và *support_count*. Trả về các tập phổ biến *frequent_itemsets* của *window_data*.

Dòng 6: Ghi nhận lại các tập phổ biến *frequent_itemsets*.

Dòng 7: thực hiện việc trượt cửa sổ thêm một batch.

Dòng 9: Trả về kết quả của thuật toán 2.

B. VÍ DỤ MINH HOẠ:

Cho một CSDL giao dịch:

Bảng 4 CSDL giao dịch với lần trượt thứ nhất

T _{id}	Itemset
T ₁	2, 4, 6
T ₂	2, 4, 1, 5
T ₃	6, 4, 1, 3
T ₄	2, 6, 4, 1

T ₅	2, 6, 4, 3
T ₆	1, 3, 5
T ₇	2, 3, 5
T ₈	1, 4, 6

Giải thuật di truyền được đề xuất sử dụng ngưỡng bao gồm tập phổ biến của giải thuật di truyền là $\text{support_count} = (\text{kích thước cửa sổ trượt} * \text{support} * \text{confidence})$. Cho kích thước cửa sổ trượt $\text{win_size}=5$, $\text{support}=0.5$ và $\text{confidence}=0.6$. Xét cửa sổ trượt đầu tiên từ T_1 đến T_5 . Ta có: support_count là: $5 * 0.5 * 0.6=1.5$.

Khi từng giao dịch được xử lý, sẽ được ghi nhận số lần xuất hiện của mỗi hạng mục (item) trong các giao dịch, và những hạng mục không đáp ứng ngưỡng support_count tối thiểu sẽ bị loại bỏ. Trong ví dụ này, Mục 5 bị loại bỏ không đưa vào trong việc khởi tạo gene. Với số lượng hạng mục (item) ứng viên ánh xạ trong bộ dữ liệu của cửa sổ trượt này là năm, bước đầu tiên là tạo một quần thể ngẫu nhiên gồm $2^{(5-1)}$ hoặc 16 cá thể, mỗi cá thể có 5 gene và mỗi gene ánh xạ tới một trong các mục có thể có.

Một số cá thể được trình bày trong Bảng 5, với các ứng viên ánh xạ [1, 2, 3, 4, 6] cho lần trượt thứ nhất như hình ở bảng 4.

Bảng 5 Ví dụ về thuật toán di truyền

Cá thể (1)	Gồm ứng viên ánh xạ (2)	Kiểm tra sự thích nghi (3)	Chọn lọc để lai ghép (4)	Cá thể mới (5)	Đột biến cá thể mới (6)	Gồm (7)	Độ thích nghi (8)
[01101]	[2, 3, 6]	1	Cá thể 2,3	[01001] Con 1	[01001]	[2, 6]	3
[00011]	[4, 6]	4	Cá thể 2,3	[10010] Con 2	[10010] Không có gen đột biến	[1, 4]	3
[11000]	[1, 2]	2	Cá thể 2,4	[00011] Không lai ghép. Xác suất không vượt quá ngưỡng, cá thể 2 được chuyển sang thể hệ tiếp theo	[00011]	[4, 6]	4
[01010]	[2, 4]	4	Cá thể 2,4	[01010] Không lai ghép. Xác suất không vượt quá ngưỡng, cá thể 4 được chuyển sang thể hệ tiếp theo	[01010] Không có gen đột biến	[2, 4]	4

Trong bảng 5, cột 1 (cá thể) biểu diễn quần thể ban đầu được khởi tạo gồm 4 cá thể {01101, 00011, 11000, 01010} bằng cách phát sinh ngẫu nhiên 1 hoặc 0, với 1 là mục ánh xạ tương ứng có xuất hiện, 0 thì ngược lại. Cột 2 gồm các mục tương ứng với số 1 bên cột 1. Cột 3 tính *support* của từng cá thể ở cột 1 với lần lượt các giá trị là: 1, 4, 2, 4. Với *support_count* = 1.5 vừa tính ở trên thì cá thể đầu tiên bị loại. Ta còn lại 3 cá thể thứ hai, ba và tư. Cột 4 sau khi chọn lọc 3 cá thể còn lại, ta lai ghép các cặp như sau: 2-3, 2-4. Việc lai ghép cũng được chọn ngẫu nhiên và theo xác suất lai ghép cho trước. Nếu không vượt ngưỡng xác suất thì không lai ghép. Cột (5) biểu diễn các cá thể con đã lai ghép hoặc các quần thể trước đó. Cột (6) biểu diễn việc đột biến cá thể mới với xác suất đột biến cho trước và thực hiện tương tự như lai ghép. Cột (7) thể hiện các mục ánh xạ tương ứng như cột 2. Cột (8) tính *support* như cột (3).

Quá trình trên được lặp lại cho đến khi đạt được một điều kiện dừng nào đó. Điều kiện dừng này bao gồm số thế hệ tối đa hoặc một điều kiện về sự thích nghi. Thường sẽ xác định trước số lần lặp (số thế hệ). Ta ghi nhận được các tập phổ biến cho lần trượt thứ nhất này.

Tiếp theo, ta trượt cửa sổ cho lần trượt thứ hai với *batch* = 3. Bảng 6 minh họa cửa sổ trượt lần hai như sau:

Bảng 6 CSDL với lần trượt thứ hai

T _{id}	Itemset
T ₁	2, 4, 6
T ₂	2, 4, 1, 5
T ₃	6, 4, 1, 3
T ₄	2, 6, 4, 1
T ₅	2, 6, 4, 3
T ₆	1, 3, 5
T ₇	2, 3, 5
T ₈	1, 4, 6

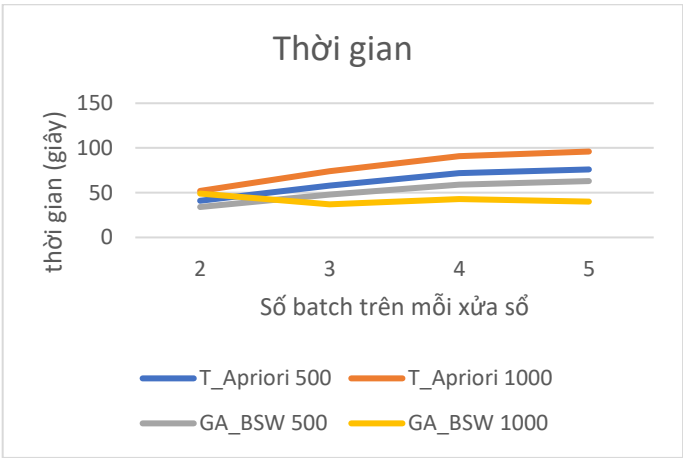
Thực hiện thuật toán di truyền tương tự như lần trượt cửa sổ thứ nhất, ta cũng thu hoạch được các tập phổ biến của dữ liệu trong cửa sổ này.

Bài báo này, chúng tôi chỉ có tạo ra các tập phổ biến, không hiển thị các luật kết hợp.

V. ĐÁNH GIÁ THỰC NGHIỆM

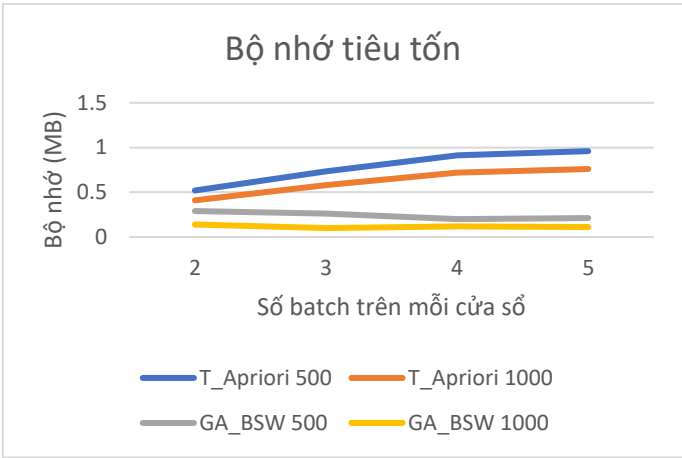
Các thử nghiệm được thực hiện trên máy tính CPU 2.90 GHz, 4 nhân và 32 GB bộ nhớ chạy trên hệ điều hành Microsoft Windows 10 64-bit. Chương trình được viết với ngôn ngữ lập trình Python, môi trường cài đặt IDLE (Python 3.11 64-bit). Với bộ dữ liệu Mushroom, có độ dài trung bình của giao dịch là 23, số lượng các hạng mục 119 và số lượng giao dịch là 8124.

Đầu tiên, bài báo tiến hành so sánh hiệu quả về mặt thời gian thực hiện của thuật toán đề xuất (*GA_BSW*) và thuật toán *T_Apriori* [3] trên bộ dữ liệu Mushroom với kích thước cửa sổ trượt khác nhau bằng cách thay đổi số lô trong một cửa sổ và số giao dịch trong một lô. Hình 5 cho thấy sự khác nhau về kích thước của cửa sổ với số lượng lô từ 2 đến 5. Mỗi thuật toán sẽ được thực hiện với kích thước mỗi lô là 500 và 1000 giao dịch. Giá trị ngưỡng *support_count* được chọn là 0.25%. Cùng với các tham số cơ bản của thuật toán di truyền: số thế hệ tối đa (*generations*) là 20, kích thước quần thể (*population_size*) là 100, xác suất lai ghép là 0.8 và xác suất đột biến là 0.2. Kết quả thực nghiệm cho thấy rằng thuật toán *GA_BSW* hiệu quả hơn rất nhiều so với thuật toán *T_Apriori* do không tốn thời gian phát sinh các tập con các ứng viên. Hoạt động phát sinh mẫu được thực hiện trực tiếp khi phát sinh quần thể của thuật toán đề xuất. Kết quả thực nghiệm cũng cho thấy rằng, khi số lượng giao dịch trong một lô là 500 sẽ tốn nhiều thời gian khai thác hơn do phải thực hiện hoạt động khi trượt sang cửa sổ khác nhiều lần hơn (Hình 4).



Hình 4 So sánh thời gian thực hiện thuật toán T_Apriori và GA_BSW

Tiếp theo, bài báo thực nghiệm so sánh về bộ nhớ của GA_BSW và T_Apriori. Kết quả trong Hình 5 thể hiện GA_BSW tiêu tốn ít bộ nhớ và ổn định hơn do phát sinh một số lượng xác định quần thể và số thể hệ ngay khi số lượng giao dịch trong một lô là 500.



Hình 5 Biểu diễn bộ nhớ tiêu thụ giữa T_Apriori và GA_BSW

VI. KẾT LUẬN

Bài báo này trình bày một nghiên cứu về khai thác tập phổ biến từ dữ liệu luồng. Một phương pháp sử dụng thuật toán di truyền được trình bày và khám phá các mối quan hệ giữa kích thước cửa sổ trượt và các ràng buộc của thuật toán di truyền. Support được trình bày để tính toán số lần xuất hiện tối thiểu để xác định tập phổ biến trong dữ liệu luồng sử dụng cửa sổ trượt.

Có hai cải tiến xử lý song song trong tương lai để thời gian thực thi của thuật toán nhanh hơn. Thứ nhất, thuật toán di truyền có thể được đa luồng để cho phép dòng dữ liệu tiếp tục trong khi các tập phổ biến đang được tạo ra, bằng cách xử lý từng lô (batch) vào từng tiểu trình. Thứ hai, có thể song song hóa hàm thích nghi khi tính support_count của từng ứng viên với các transactions trong cửa sổ trượt. Ngoài ra còn một giải pháp cải tiến nữa là dùng *BitWise* để thuật toán chạy nhanh hơn.

VII. TÀI LIỆU THAM KHẢO

[1] Agrawal, R., T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of ACM SIGMOD conference*, Washington, D.C., USA, 1993.

[2] Ghosh S, Biswas S, Sarkar D, Sarkar PP, "Mining frequent itemsets using GA," *Int J Artif Intell Appl*, vol. 1(4):133–43, 2010.


- [3] X. Yuan, "An improved Apriori algorithm for mining association rules," in *AIP Conference Proceedings* 1820, Shanghai, 2017.
- [4] Nedunchezian R, Geethanindhini K., "Association rule mining on Big Data—a survey," *Int J Eng Res Technol*, vol. 5(5):42–6, 2016.
- [5] Vijayarani S, Sathya P, "An efficient algorithm for mining frequent items in data streams," *Int J Innov Res Comput Commun Eng*, vol. 1(3):742–7, 2013.
- [6] Yu PS, Chi Y, "Association rule mining on streams," *Encyclopedia of Database Systems*, 2009.
- [7] Krempel G, Zliobaite I, Brzezinski D, Hullermeier E, Last M, Lemaire V, Noack T, Shaker A, Sievi S, Spiliopoulou M, Stefanowski J., "Open challenges for data stream mining research," *ACM SIGKDD Explor News*, vol. 16(1):1–10, 2014.
- [8] G. J., "A survey on learning from data streams: current and future trends," *Prog Arti Intell*, vol. 1(1):45–55, 2012.
- [9] Hoens TR, Polikar R, Chawla NV, "Learning from streaming data with concept drift and imbalance: an overview," *Prog Artif Intell*, vol. 1(1):89–101, 2012.
- [10] Kim Y, Park CH., "An efficient concept drift detection method for streaming data under limited labeling," *IEICE Trans Inf Syst*, vol. 100(10):2537–46, 2017.
- [11] Wang H, Abraham Z., "Concept drift detection for streaming data," in *International joint conference on neural networks (IJCNN)*, 2015.
- [12] B. AD., "Convergence rates of efficient global optimization algorithms," *J Mach Learn Res*, p. 12(88):2879–904., 2011.
- [13] Rabinovich Y, Wigderson A., "Techniques for bounding the convergence rate of GAs.," *Random Struct Algorithms*, vol. 14(2):111–38, 1999.
- [14] Forrest S, Mitchell M., "What makes a problem hard for a GA? Some anomalous results and their explanation," *GAs Mach Learn*, 1993.
- [15] Ruholla J-M, Smith BK., "Fluid GA (FGA)," *J Comput Des Eng*, vol. 4(2):158–67, 2017.
- [16] A. AS, "Avoiding premature convergence of GA in information retrieval systems," *Int J Intell Syst Appl Eng*, vol. 2, no. 4, p. 80, 2015.
- [17] Lin W-Y, Lee W-Y, Hong T-P, "Adapting crossover and mutation rates in GAs," *J Inf Sci Eng*, vol. 19(5):889–903, 2003.
- [18] Angelova M, Pencheva T, "Tuning GA parameters to improve convergence time.," *Int J Chem Eng*, 2011.
- [19] Pellerin E, Pigeon L, Delisle S., "Self-adaptive parameters in Genetic Algorithms," in *Proceedings SPIE 5433. Data mining and knowledge discovery: theory, tools, and technology VI*, 2004.
- [20] He J, Lin G., "Average convergence rate of evolutionary algorithms," *IEEE Trans Evol Comput*, vol. 20(2):316–21, 2016.
- [21] Rangaswamy S, Shobha G., "Optimized association rule mining using GA," *J Comput Sci Eng Inf Technol Res*, vol. 2(1):1–9, 2012.
- [22] Kolajo T, Daramola O, Adebisi A., "Big data stream analysis: a systematic literature review," *J Big Data*, p. 6:47, 2019.
- [23] Fortin F-A, De Rainville F-M, Gardner M-A, Parizeau M, Gagné C, "DEAP: evolutionary algorithms made easy," *J Mach Learn Res*, p. 13:2171–5., 2012.

EXTRACTING FREQUENT ITEMSETS FROM STREAMING DATA USING GENETIC ALGORITHM

Pham Duc Thanh, Le Thi Minh Nguyen

ABSTRACT - This paper presents a study on mining frequent sets of terms from streaming transaction data in the context of evolving concepts. Streaming data, characterized by its instability, poses numerous challenges in the mining process. A method utilizing a genetic algorithm is proposed, and the relationship between concept drift, sliding window size, and genetic algorithm constraints is explored. Concept drift is identified through changes in frequent itemsets. The uniqueness of this study lies in determining concept drift by leveraging frequent itemsets for streaming data mining, employing a genetic algorithm framework. An equation is presented to compute the minimum support count in streaming data using a sliding window. Experiments have indicated that the ratio between the window size and the number of transactions per drift is a critical factor for achieving good performance. Attaining satisfactory results with excessively small window sizes poses a challenge as normal fluctuations in data can manifest as concept drift. The window size must be managed alongside the support value and confidence level to achieve reasonable outcomes. This approach to concept drift detection has performed well when using larger window sizes.

Phạm Đức Thành nhận học vị Thạc sĩ năm 2006 tại Đại học Quốc gia Thành phố Hồ Chí Minh; hiện đang là Giảng viên công tác tại khoa Công nghệ Thông tin trường Đại học Ngoại ngữ Tin học Tp. Hồ Chí Minh; lĩnh vực nghiên cứu đang quan tâm là: khai thác dữ liệu.



Lê Thị Minh Nguyễn nhận học vị thạc sĩ Khoa học máy tính trường Đại học Quốc gia Thành phố Hồ Chí Minh năm 2007. Hiện là giảng viên khoa Công Nghệ Thông Tin trường Đại Học Ngoại Ngữ Tin Học thành phố Hồ Chí Minh. Lĩnh vực nghiên cứu đang quan tâm là: khai thác dữ liệu.

