PHƯƠNG PHÁP KHAI THÁC TẬP MỤC HỮU ÍCH CAO TỪ LUỒNG DỮ LIỆU DỰA TRÊN THUẬT TOÁN BẦY VOI

Lê Thị Minh Nguyện, Phạm Đức Thành, Trần Anh Duy, Trần Minh Thái

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

nguyenltm@huflit.edu.vn, thanhpd@huflit.edu.vn, duyta@huflit.edu.vn, thaitm@huflit.edu.vn

TÓM TẮT— Khai thác tập mục hữu ích cao (High Utility Itemset Mining - HUIM) là một hướng nghiên cứu nổi bật trong lĩnh vực khai phá dữ liệu. Các thuật toán HUIM truyền thống thường gặp khó khăn khi không thể xử lý sự bùng nổ theo cấp số mũ của không gian tìm kiếm, dẫn đến hạn chế về khả năng mở rộng. Để khắc phục vấn đề này, các thuật toán HUIM dựa trên phương pháp heuristic đã thu hút được nhiều sự quan tâm. Tuy nhiên, những phương pháp này lại dễ hội tụ sớm, gây ra hiện tượng bỏ sót các tập mục hữu ích tiềm năng. Nhằm khắc phục những hạn chế này, chúng tôi đề xuất một thuật toán mới có tên **SHUIM_HE**, dựa trên giải thuật bầy voi, để khai thác hiệu quả các tập mục có giá trị cao từ luồng dữ liệu trong môi trường tài nguyên hạn chế. Đổi mới cốt lõi của thuật toán là chiến lược tiến hóa vị trí dựa trên yếu tố voi cái, giúp giảm đáng kể không gian tìm kiếm và nâng cao hiệu suất thực thi của thuật toán. Các thí nghiệm thực hiện trên các tập dữ liệu thực tế cho thấy thuật toán đề xuất vượt trội hơn so với các thuật toán HUIM heuristic tiên tiến hiện nay.

Từ khóa— khai thác tập mục hữu ích cao, luồng dữ liệu, bảng băm, cửa sổ trượt, bầy voi.

I. GIỚI THIỆU

Khai thác tập mục hữu ích cao (HUIM) là một hướng nghiên cứu quan trọng trong lĩnh vực khai phá dữ liệu, mở rộng từ bài toán khai thác tập mục phổ biến truyền thống. Khác với các phương pháp chỉ dựa vào tần suất xuất hiện, HUIM đồng thời xem xét cả tần suất và độ hữu ích của các tập mục, nhằm phát hiện những mẫu có giá trị tiềm ẩn trong các tập dữ liệu lớn. Trong bối cảnh phân tích thị trường bán lẻ, mỗi giao dịch bao gồm số lượng các mặt hàng (độ hữu ích nội) và giá của từng mặt hàng (độ hữu ích ngoại). Độ hữu ích của một mặt hàng – hay còn gọi là lợi nhuận – được xác định bằng tích của độ hữu ích nội và độ hữu ích ngoại. Việc xem xét cả hai yếu tố này cho phép HUIM phát hiện ra những tập mục không chỉ phổ biến mà còn mang lại giá trị kinh tế cao, từ đó hỗ trợ hiệu quả cho các quyết định kinh doanh và quản trị.

Ban đầu, các nghiên cứu HUIM sử dụng thuật toán hai giai đoạn [1], đòi hỏi nhiều lần quét dữ liệu và sinh ra một lượng lớn tập ứng viên, làm giảm hiệu quả tính toán. Để cải thiện, các phương pháp dựa trên cấu trúc cây như IHUP [2], UP-tree [3] và HUITWU-tree [4] đã được đề xuất nhằm giảm số lần quét cơ sở dữ liệu. Tuy nhiên, các phương pháp này vẫn tiêu tốn nhiều bộ nhớ do số lượng lớn cây điều kiện được tạo ra. Một bước tiến quan trọng là công trình của Liu và cộng sự [5] đề xuất cấu trúc danh sách để tránh việc sinh tập ứng viên và quét lại dữ liệu, tuy nhiên hiệu năng giảm đáng kể khi dữ liệu mở rộng.

Gần đây, sự bùng nổ của các luồng dữ liệu từ cảm biến, giao dịch điện tử và mạng xã hội đã thúc đẩy nghiên cứu HUIM trong môi trường luồng dữ liệu. Khác với dữ liệu tĩnh, luồng dữ liệu có đặc điểm liên tục, tốc độ cao, thay đổi theo thời gian và khối lượng lớn, gây ra nhiều thách thức như bùng nổ tổ hợp. Để xử lý hiệu quả luồng dữ liệu, các mô hình cửa sổ như cửa sổ trượt và cửa sổ giảm dần [6] đã được áp dụng nhằm giới hạn phạm vi phân tích. Một số thuật toán tiêu biểu gồm SOHUPDS [7] sử dụng kỹ thuật chiếu cơ sở dữ liệu, IUDataListSW và HUIM-Stream [8] với bảng chỉ mục Ext-list giúp giảm độ phức tạp.

Tuy nhiên, các phương pháp truyền thống vẫn gặp vấn đề về tiêu thụ bộ nhớ và trùng lặp dữ liệu giữa các cửa sổ. Trong khi đó, các thuật toán heuristic tuy nhẹ hơn nhưng có nguy cơ bỏ sót các tập mục quan trọng, làm giảm độ chính xác và khả năng hội tụ. Nhằm khắc phục những hạn chế này, bài báo đề xuất thuật toán SHUIM_HE – kết hợp giữa khai thác tập mục hữu ích cao với giải thuật tối ưu bầy voi và mô hình cửa sổ trượt. SHUIM_HE được lựa chọn nhờ khả năng tìm kiếm toàn cục hiệu quả, ít yêu cầu điều chỉnh tham số và đã được chứng minh về hiệu suất. SHUIM_HE áp dụng chiến lược cập nhật quần thể kết hợp với bảng băm, giúp giảm trùng lặp dữ liệu, tăng tốc độ hội tụ và hạn chế mất mát các tập mục hữu ích quan trọng.

Phần còn lại của bài báo được trình bày như sau: Phần II trình bày về các công trình liên quan; để cung cấp nền tảng và ngữ cảnh, phần III trình bày các khái niệm và định nghĩa, trình bày cách nhìn tổng quan về dữ liệu luồng, độ hữu ích trong giao dịch và tối ưu hóa bầy voi; Phần IV trình bày phương pháp thực hiện; tiếp theo, trình bày kết quả thực nghiệm về phương pháp đề xuất SHUIM_HE khác biệt so với thuật toán HSLM [9] và thuật toán HUIM-stream [8]. Cuối cùng, trình bày phần kết luận và tương lai của đề tài.

II. NGHIÊN CỨU LIÊN QUAN

Thuật toán khai thác tập mục hữu ích cao (High Utility Itemset Mining – HUIM) là một nhánh quan trọng trong khai phá dữ liệu, được sử dụng rộng rãi trong các lĩnh vực như thương mại điện tử, phân tích giỏ hàng và ra quyết định kinh doanh. Khác với các phương pháp truyền thống chỉ xét đến tần suất, HUIM xem xét thêm yếu tố hữu ích như doanh thu, số lượng hoặc trọng số, từ đó phản ánh chính xác hơn giá trị thực tế của các mẫu khai thác.

A. HUIM TRÊN CƠ SỞ DỮ LIỆU TĨNH

Ban đầu, hầu hết các thuật toán HUIM được thiết kế cho cơ sở dữ liệu tĩnh. Các hướng tiếp cận chủ yếu dựa trên Apriori, danh sách hữu ích, cây tìm kiếm hoặc chiếu cơ sở dữ liệu (database projection). Trong đó, thuật toán CoIUM [10] là một cải tiến đáng chú ý khi kết hợp giữa độ hữu ích và độ tương quan của các mục, từ đó loại bỏ các tập mục hữu ích cao nhưng không có ý nghĩa thực tế. CoIUM sử dụng kỹ thuật chiếu cơ sở dữ liệu và nhiều chiến lược cắt tỉa (prune) để giảm đáng kể không gian tìm kiếm.

Hướng tiếp cận tiêu biểu trong khai thác tập mục hữu ích cao (HUIM) là nghiên cứu của Kannimuthu [11], người đầu tiên áp dụng thuật toán di truyền vào bài toán này thông qua hai mô hình: HUPEumu-GARM (dựa trên ngưỡng hữu ích tối thiểu) và HUPEwumu-GARM (không yêu cầu ngưỡng). Mặc dù các phương pháp này cho thấy khả năng tìm kiếm hiệu quả trong không gian tìm kiếm lớn, nhưng chúng vẫn gặp phải những hạn chế nhất định, đặc biệt là về tốc độ hội tụ chậm và nguy cơ rơi vào điểm tối ưu cục bộ, dẫn đến khả năng bỏ sót các thông tin có giá trị tiềm ẩn. Để cải thiện các vấn đề trên, TKHUIM-GA [12] được đề xuất với chiến lược sử dụng độ hữu ích của từng mục để hướng dẫn quá trình tiến hóa. Đồng thời, phương pháp này khai thác mô hình HUIM Top-k không cần ngưỡng, giúp tiết kiệm tài nguyên tính toán và giảm thiểu khả năng loại bỏ sớm những mục có giá trị. Tiếp nối cải tiến đó, HUIM-IGA [13] – một biến thể nâng cao của thuật toán di truyền – đã tích hợp chiến lược sửa lỗi dựa trên 1-HTWUI cùng với cơ chế tìm kiếm hàng xóm ưu tú. Nhờ vậy, không gian giải pháp được mở rộng và xác suất giữ lại các thông tin hữu ích được nâng cao, góp phần cải thiện độ chính xác và tính toàn diện của kết quả khai thác.

Một hướng nghiên cứu khác kết hợp HUIM với mục tiêu bảo vệ quyền riêng tư, điển hình là PPUMGAT+ [14], sử dụng khái niệm "pre-large" cùng ngưỡng hỗ trợ trên và dưới nhằm tạo vùng đệm an toàn, hạn chế mất mát các tập mục nhạy cảm. Trong khi đó, DcGA [15] và MCUI-Miner [16] tiếp cận bài toán bằng mô hình phân cụm, cho phép khai thác các tập mục hữu ích cao đóng (CHUIs) cục bộ và hợp nhất dần thành CHUIs toàn cục, giảm chi phí xử lý và bảo toàn thông tin trong quá trình tổng hợp. Gần đây, MOBGWO [17] tiếp tục mở rộng hướng khai thác tối ưu đa mục tiêu bằng cách áp dụng thuật toán Grey Wolf Optimizer có tính phân rã, cho thấy hiệu quả đáng kể trong các môi trường dữ liệu quy mô lớn.

B. HUIM TRÊN LUỒNG DỮ LIỆU

Trên luồng dữ liệu, các thuật toán cần đáp ứng yêu cầu xử lý theo thời gian thực, bộ nhớ giới hạn và khả năng thích ứng với dữ liệu đến liên tục. Các thuật toán như EGUI-tree [18] và THUI [19] mở rộng mô hình cửa sổ trượt (sliding window) để duy trì các tập mục hữu ích trong khoảng thời gian xác định. Hướng tiếp cận theo kiểu top-k, ví dụ như Vert_top-k_DS [20], giúp người dùng không cần xác định ngưỡng hữu ích trước, từ đó tăng tính linh hoạt. Trong trường hợp dữ liệu có tính tuần tự, thuật toán HUSP-Stream [21] đề xuất cấu trúc bảng DUI (Dynamic Utility Index) để khai thác các mẫu tuần tự có lợi ích cao trong luồng dữ liệu, đồng thời giảm thiểu độ trễ xử lý.

Ngoài ra, khai thác HUIM trên luồng dữ liệu nổi lên như một hướng tiếp cận đầy tiềm năng nhằm ứng phó với tính chất vô hạn, tốc độ cao và thay đổi liên tục của dữ liệu thời gian thực. SOHUPDS [7] là một trong những phương pháp tiên phong trong mô hình cửa sổ trượt, sử dụng cấu trúc IUDataListSW để lưu trữ thông tin về độ hữu ích, vị trí và giới hạn trên của mục, đồng thời tái sử dụng kết quả từ các cửa sổ trước nhằm giảm tiêu thụ bộ nhớ và thời gian xử lý. HUMHDT [22] phát triển kiến trúc hệ thống phân tán nhằm bảo toàn dữ liệu lịch sử mà không làm gián đoạn quá trình khai thác hiện tại. Bên cạnh đó, EGUI-tree [23] và HUIM-Stream [8] lần lượt đề xuất cấu trúc cây mở rộng và bảng chỉ mục Ext-list, kết hợp với cơ chế lưu trữ kết quả bằng bảng băm, giúp nâng cao hiệu quả cập nhật và truy xuất, giảm đáng kể mất mát thông tin giữa các phiên xử lý.

Nhìn chung, các phương pháp HUIM hiện đại đã có nhiều bước tiến trong việc giảm thiểu mất mát thông tin thông qua các chiến lược tiến hóa có định hướng, khai thác ngữ cảnh lịch sử, và tối ưu kiến trúc dữ liệu. Tuy nhiên, một số phương pháp vẫn gặp khó khăn khi đối mặt với dữ liệu động hoặc phân phối không đồng đều, cho thấy tiềm năng cho các nghiên cứu tiếp theo thích ứng theo thời gian và tự động điều chỉnh tham số khai thác nhằm bảo toàn tối đa các thông tin giá trị trong dữ liệu lớn.

III. CÁC ĐỊNH NGHĨA VÀ KHÁI NIỆM

A. ĐỊNH NGHĨA

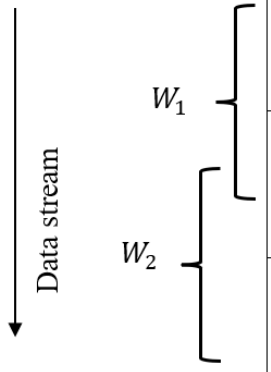
Cho $I = \{i_1, i_2, \dots, i_m\}$ là một tập hợp các mục khác nhau, $Ds = \{T_1, T_2, \dots, T_n\}$ là một chuỗi giao dịch, và mỗi giao dịch $T_q \in DS$ $(1 \le q \le n)$ là một tập con của I. Hàm $q(i_j, T_q)$ đại diện cho độ hữu ích nội của mục i_j trong giao dịch

T_q, và $p(i_j)$ đại diện cho độ hữu ích ngoại của mục i_j. Tập mục $X = \{i_1, i_2, \dots, i_k\}$ (1 ≤ k ≤ m) là một tập con khác rỗng của I. Trong khai thác tập mục hữu ích cao (HUIM) trên luồng dữ liệu dựa trên mô hình cửa sổ trượt, mỗi cửa sổ $W_b = \{B_{b+1}, B_{b+2}, \dots, B_{b+r}\}$ bao gồm một số lượng cố định r các lô (batch), mỗi lô $B_t = \{T_{t+1}, T_{t+2}, \dots, T_{t+s}\}$ bao gồm một số lượng cố định s các giao dịch.

Hình 1 minh họa một ví dụ đơn giản về luồng dữ liệu, trong đó kích thước cửa sổ là 2 và kích thước lô là 3. IU (Internal Utility) là độ hữu ích nội, và TU (Transaction Utility) là độ hữu ích của giao dịch cho các mục trong từng dòng giao dịch. Bảng 1 thể hiện độ hữu ích ngoại của từng mục trong luồng dữ liệu.

Bảng 1. *Giá trị hữu ích ngoại (EU)*

Item	a	b	c	d	e	f	g
EU	6	7	10	5	9	8	13



BID	TID	Items	IU	TU
B_1	T_1	a, b, d, e	2, 3, 3, 5	93
	T_2	a, b, c, d, e	3, 3, 4, 5, 2	122
	T_3	b, d, e, f	3, 3, 5, 4	113
B_2	T_4	a, c, e, f, g	1, 3, 4, 5, 4	164
	T_5	c, e, g	3, 4, 5	131
	T_6	a, b, f, g	3, 2, 2, 3	87
B_3	T_7	c, d, e	1, 3, 2	43
	T_8	b, c, e	3, 1, 5	76
	T_9	b, d, e, g	3, 4, 5, 6	164

Hinh 1. *Luồng dữ liệu*

Định nghĩa 1. Độ hữu ích của một mục [8]. Mục i_j trong giao dịch T_q được biểu diễn là $u(i_j, T_q)$. Độ hữu ích này được tính bằng cách nhân hữu ích nội của mục i_j trong giao dịch T_q với hữu ích ngoại của i_j, theo công thức (1).

$$u(i_j, T_q) = q(i_j, T_q) \times p(i_j) \quad (1)$$

Ví dụ như trong Hình 1 và Bảng 1, $u(a, T_1) = q(a, T_1) \times p(a) = 2 \times 6 = 12$

Định nghĩa 2. Độ hữu ích của tập mục trong giao dịch [8]. Độ hữu ích của tập mục X trong giao dịch T_q được biểu diễn là $u(X, T_q)$, là tổng hữu ích của tất cả các mục thuộc tập X có trong giao dịch T_q. Độ hữu ích này được tính theo công thức (2).

$$u(X, T_q) = \sum_{i_j \in X \wedge X \in t_q} u(i_j, T_q) \quad (2)$$

Ví dụ: $u(\{a, b\}, T_1) = u(a, T_1) + u(b, T_1) = 2 \times 6 + 3 \times 7 = 33$

Định nghĩa 3. Độ hữu ích của giao dịch [8]. Độ hữu ích của một giao dịch T_q được biểu diễn là $TU(T_q)$, là tổng hữu ích của tất cả các mục trong T_q, được tính theo công thức (3).

$$tu(T_q) = \sum_{i_j \in T_q} u(i_j, T_q) \quad (3)$$

Ví dụ: $TU(T_5) = u(c, T_5) + u(e, T_5) + u(g, T_5) = 3 \times 10 + 4 \times 9 + 5 \times 13 = 131$

Định nghĩa 4. Độ hữu ích trọng số giao dịch (TWU) [8]. Tập mục X trong cửa sổ W_b, Độ hữu ích trọng số giao dịch được biểu diễn là $TWU(X, W_b)$, là tổng hữu ích của tất cả các giao dịch chứa X trong cửa sổ W_b, được tính theo công thức (4)

$$TWU(X, W_b) = \sum_{X \in T_q \wedge T_q \in W_b} TU(T_q) \quad (4)$$

Ví dụ: $TWU(\{a, c\}, W_1) = TU(T_2) + TU(T_4) = 122 + 164 = 286$

Định nghĩa 5. Ngưỡng hữu ích tối thiểu [8]. Cửa sổ W_b có ngưỡng hữu ích tối thiểu được biểu diễn là min_util, là tích của tổng hữu ích của tất cả các giao dịch trong cửa sổ W_b với tỷ lệ phần trăm ngưỡng hữu ích tối thiểu do người dùng định nghĩa δ, được tính theo công thức (5).

$$\left(\min\_util_{W_b}\right) = \sum_{T_q \in W_b} TU\left(T_q\right) \times \delta \quad (5)$$

Ví dụ: với độ hỗ trợ δ=0.3, ngưỡng hữu ích tối thiểu của của sổ W1 có min_util=(93 + 122 + 113 + 164 +131 + 87) x 0.3 = 213.

B. PHÁT BIỂU BÀI TOÁN

Cho luồng dữ liệu DS = {T1, T2, … Tn} là một dãy các giao dịch, min_util là ngưỡng hữu ích tối thiểu do người dùng xác định, win_size là kích thước cửa sổ trượt do người dùng xác định, bat_size là kích thước lô, pop_size là kích thước quần thể và max_iter là số lần lặp tối đa. Mục tiêu của HUIM (khai thác tập mục có tiện ích cao) là sử dụng thuật toán tối ưu đàn voi (EHO) để tìm tất cả các tập mục hữu ích cao (HUIs) thỏa mãn ngưỡng min_util trong mỗi cửa sổ trượt từ luồng dữ liệu.

Bảng 2. *Ký hiệu và ý nghĩa*

Ký hiệu	Ý nghĩa	Ký hiệu	Ý nghĩa
DS	Data Stream (luồng dữ liệu)	min_util	Minimum Utility Threshold (Ngưỡng hữu ích tối thiểu)
W	Sliding Window (cửa sổ trượt)	win_size	Number of batches in a sliding window (số lô trong một cửa sổ trượt)
B	Batch (lô)	batch_size	Number of transactions in a batch (số giao tác trong một lô)
T	Transaction (giao tác)	pop_size	Number of in population (kích thước quần thể)
I	Item (mục)	max_iter	Maximum iterations (Số vòng lặp tối đa)
X	Itemset (tập mục)	SHUI	the set of HUIs (tập của các tập mục hữu ích cao)
U	Utility (độ hữu ích)	HUIs	High Utility Itemsets (tập mục hữu ích cao)
IU	Internal Utility (độ hữu ích nội)	HTWUI	High Transaction Weighted Utilization Itemset (Tập mục có độ hữu ích theo trọng số giao dịch cao)
EU	External Utility (độ hữu ích ngoại)	LTWUI	Low Transaction Weighted Utilization Itemset (Tập mục có độ hữu ích theo trọng số giao dịch thấp)
TWU	Transaction Weighted Utility (Độ hữu ích theo trọng số giao dịch)	1-HTWUI	High Transaction Weighted Utilization Itemset of 1-item (Tập mục 1 phần tử có độ hữu ích theo trọng số giao dịch cao)

C. TỐI ƯU HÓA BẦY VOI

Trong thiên nhiên, voi sống theo bầy đàn có tính xã hội. Cấu trúc của các nhóm này thường bao gồm các bầy (clan), mỗi bầy do một con voi cái dẫn đầu. Ngoài ra, trong quá trình trưởng thành, các con voi đực cũng có thể tách ra sống đơn lẻ, không thuộc về bất kỳ bầy nào. Để giải quyết các bài toán tối ưu toàn cục khác nhau, Wang và cộng sự [13] lần đầu tiên đề xuất thuật toán tối ưu bầy voi (EHO) vào năm 2019, trong đó hành vi sinh học của loài voi được trừu tượng hóa thành hai thao tác cơ bản: cập nhật bầy và tách bầy. Mỗi bầy sẽ lặp lại quá trình cập nhật vị trí của các con voi nhằm tìm ra lời giải tối ưu trong không gian tìm kiếm của bài toán.

1. CẬP NHẬT BẦY

Do các con voi trong bầy sống dưới sự lãnh đạo của voi cái, nên vị trí của mỗi con voi j trong bầy $\boldsymbol{c_i}$ sẽ bị ảnh hưởng bởi voi cái đầu đàn. Vị trí này được cập nhật theo công thức (6) như sau: [24]

$$x_{new,ci,j} = x_{ci,j} + \alpha.r(x_{best,ci} - x_{ci,j}) \quad (6)$$

Trong đó $x_{new,ci,j}$ là vị trí mới của con voi j trong bầy c_i; $x_{ci,j}$ là vị trí hiện tại (cũ) của con voi j trong bầy c_i, $x_{best,ci}$: vị trí của voi cái đầu đàn – cá thể tốt nhất trong bầy c_i;α∈[0,1]: hệ số khuếch đại, xác định mức độ ảnh hưởng của voi cái đến các thành viên khác trong bầy; r∈[0,1]: vector ngẫu nhiên được chọn từ phân phối đều.

Mỗi con voi sẽ di chuyển một bước về phía vị trí của con voi cái đầu đàn, có điều chỉnh bởi hệ số α và tính ngẫu nhiên r. Cơ chế này đảm bảo rằng bầy voi dần hội tụ về hướng lời giải tối ưu trong không gian tìm kiếm, trong khi vẫn duy trì sự đa dạng để tránh rơi vào cực trị cục bộ.

Tuy nhiên, khi $x_{ci,j} = x_{best,ci}$, thì không thể cập nhật bằng công thức (6). Lời giải tốt nhất trong không gian bài toán, voi cái sẽ được cập nhật theo công thức (7) [24], trong đó β∈ [0,1] là hệ số ảnh hưởng của $x_{center,ci}$ đến $x_{new,ci,j}$, $x_{center,ci}$ là vị trí trung tâm của bầy c_i.

$$x_{new,ci,j} = \beta \times x_{center,ci} \quad (7)$$

$$x_{center,ci,j} = \frac{1}{n_{ci}} \times \sum_{j=1}^{n_{ci}} x_{ci,j,d} \quad (8)$$

Thành phần chiều thứ d của $x_{center,ci}$ có thể được tính theo công thức (8) [24], trong đó $1 \leq d \leq d1$: biểu thị chiều thứ d; D: là tổng số chiều của không gian tìm kiếm, n_{ci}: là tổng số con voi trong bầy c_i; $x_{ci,j,d}$ là thành phần chiều thứ d của con voi $x_{ci,j}$.

2. TÁCH BẦY

Như đã đề cập trước đó, các nhóm voi được tổ chức thành các bầy, và mỗi bầy gồm một số lượng cố định voi đực và voi cái. Ngoài ra, một số lượng cố định voi đực ở mỗi thế hệ sẽ rời khỏi bầy và sống đơn lẻ. Khi giải các bài toán tối ưu, voi đực ở xa bầy được xem là lời giải tệ nhất trong không gian bài toán và sẽ được tách khỏi bầy theo công thức (9). Trong đó: x_{max}và x_{min}: lần lượt là giới hạn trên và giới hạn dưới của vị trí voi trong không gian tìm kiếm; $\boldsymbol{\gamma} \in [\boldsymbol{0}, \boldsymbol{1}]$: phân phối ngẫu nhiên áp dụng cho cá thể tệ nhất $\boldsymbol{x_{worst,ci}}$ trong bầy c_i [24]

$$x_{worst,ci} = x_{min} + (x_{max} - x_{min} + 1) \times \gamma \quad (9)$$

IV. PHƯƠNG PHÁP GIẢI QUYẾT

A. GIẢM THIỂU VIỆC MẤT CÁC TẬP MỤC QUAN TRỌNG

Trong bối cảnh khai thác dữ liệu luồng thời gian thực, việc duy trì các tập mục hữu ích cao (HUIs) trong quá trình cập nhật cửa sổ trượt là một thách thức lớn. Ví dụ, trong một hệ thống gợi ý sản phẩm của sàn thương mại điện tử, các giao dịch mua hàng liên tục được ghi nhận và xử lý để đưa ra các đề xuất cá nhân hóa kịp thời.

Giả sử trong một cửa sổ trượt đầu tiên, hệ thống thu thập được nhiều giao dịch chứa tổ hợp sản phẩm {Laptop, Chuột không dây} với tổng hữu ích cao, được xác định là một tập mục quan trọng. Tuy nhiên, khi cửa sổ trượt sang giai đoạn tiếp theo, tổ hợp này không còn xuất hiện trong dữ liệu mới. Nếu thuật toán không có cơ chế lưu trữ hoặc tái sử dụng thông tin, tập mục quan trọng này sẽ bị loại bỏ, dẫn đến mất mát thông tin giá trị và giảm hiệu quả gợi ý sản phẩm.

Bảng 3. *Cửa sổ trượt*

BID	Giao dịch	Sản phẩm	Giá trị lợi nhuận
W1	T1	Laptop, Chuột không dây	1000, 200
	T2	Laptop, Bàn phím cơ	1000, 300
	T3	Laptop, Chuột không dây	1000, 200
	T4	Tai nghe, Chuột không dây	150, 200
W2	T5	Bàn phím cơ, Tai nghe	300, 150
	T6	Tai nghe	150
	T7	Bàn phím cơ	300
	T8	Tai nghe, chuột khong dây	150, 200

Để giải quyết vấn đề này, bài báo đã sử dụng bảng băm lưu trữ kết quả (Hash Result Store – HRS). HRS cho phép lưu giữ các tập mục hữu ích cao đã được khai thác ở các cửa sổ trước, đồng thời đánh giá và quyết định giữ lại những tập mục này ngay cả khi chúng không xuất hiện trong cửa sổ hiện tại, dựa trên các ngưỡng hỗ trợ và lợi ích định trước. Cách tiếp cận này giúp giảm thiểu tối đa mất mát các tập mục quan trọng, đồng thời tăng cường độ chính xác và tính liên tục trong khai thác dữ liệu luồng.

Giả sử, ở Bảng 3 trong một khoảng thời gian 2 phút đầu tiên (cửa sổ trượt W1), các giao dịch sau được ghi nhận T1, T2, T3, T4. Thuật toán khai thác ra tập hữu ích cao là {Laptop, Chuột không dây} với tổng hữu ích 2400. Sau khi cửa sổ trượt di chuyển sang W2, các giao dịch mới đến là T5, T6, T7, T8. Trong W2, tổ hợp {Laptop, Chuột không dây} không còn xuất hiện, nếu không có chiến lược lưu trữ, thuật toán sẽ loại bỏ tập mục quan trọng này. Trước khi loại bỏ dữ liệu W1, thuật toán HUIM-Stream lưu {Laptop, Chuột không dây} vào bảng HRS. Khi cửa sổ W2 đến: Thuật toán không khai thác lại từ đầu, mà kiểm tra trong HRS để xác định xem tập này có nên giữ lại không. Nếu ngưỡng hữu ích và tần suất vẫn đủ lớn (theo ngưỡng pre-large), tập này được giữ lại, sẵn sàng sử dụng lại nếu xuất hiện trở lại.

B. LÝ DO CHỌN THUẬT TOÁN TỐI ƯU BẦY VOI (EHO) CHO BÀI TOÁN HUIM

Thuật toán Tối ưu bầy voi EHO là một thuật toán metaheuristic được lấy cảm hứng từ hành vi bầy đàn của voi trong tự nhiên, trong đó quần thể được chia thành các đàn, mỗi đàn có một con voi cái đầu đàn (matriarch) dẫn dắt các thành viên khác. Mỗi cá thể voi trong thuật toán đại diện cho một lời giải ứng viên. Đặc trưng này của EHO đặc biệt phù hợp để giải quyết bài toán HUIM trong môi trường dữ liệu luồng vì những lý do sau:

Không gian tìm kiếm rời rạc và siêu cấp số: không gian lời giải của bài toán HUIM là tập hợp tất cả các tổ hợp mục có thể có, tức là tăng theo cấp số mũ với số lượng mục. Đây là môi trường đặc biệt phù hợp với các thuật toán tối ưu bầy đàn, vì các chiến lược như phân đàn, di chuyển toàn cục và cục bộ giúp tìm kiếm hiệu quả hơn thay vì liệt kê toàn bộ tập ứng viên như các thuật toán truyền thống (Apriori, FHM, ...).

Khả năng song song hóa qua nhiều đàn: EHO chia quần thể thành nhiều đàn, cho phép đánh giá và cập nhật nhiều lời giải đồng thời. Điều này đặc biệt phù hợp với mô hình window-based data stream, nơi mà mỗi cửa sổ dữ liệu mới yêu cầu tính toán lại HUIs. Việc phân đàn cho phép xử lý song song nhiều tập ứng viên tương ứng với các khối dữ liệu mới.

Tối ưu dựa trên nhị phân và định hướng HUI: Trong thuật toán HUIM_EHO, mỗi cá thể voi được biểu diễn bằng một chuỗi nhị phân, đại diện cho việc chọn hoặc không chọn một mục (item). Điều này cho phép ánh xạ trực tiếp giữa không gian tìm kiếm của EHO và các tập mục trong HUIM. Ngoài ra, quá trình cập nhật vị trí trong EHO đã được điều chỉnh với các toán tử logic như XOR, giúp tránh kẹt cục bộ và duy trì đa dạng cá thể.

Cân bằng khai phá và khai thác: Trong bối cảnh dữ liệu luồng, việc liên tục cập nhật tập HUIs mới đòi hỏi thuật toán phải duy trì sự cân bằng giữa khai phá toàn cục và khai thác cục bộ. EHO làm tốt điều này qua hai cơ chế chính: (1) Tách đàn định kỳ: giúp các cá thể rời khỏi vùng kém hiệu quả và khởi tạo lại ở vị trí mới (cơ chế tránh kẹt cục bộ). (2)Ảnh hưởng trọng số từ đầu đàn: giúp tăng cường khả năng hội tụ về vùng lời giải tốt.

Phù hợp với dữ liệu cập nhật theo lô (batch): Do luồng dữ liệu được xử lý theo lô, EHO có thể khởi động và cập nhật lại vị trí đàn cho mỗi cửa sổ (window) hoặc mỗi batch dữ liệu mới. Điều này rất phù hợp với đặc trưng phân đoạn tự nhiên của luồng dữ liệu, không đòi hỏi thuật toán phải giữ trạng thái toàn cục giữa các batch – giúp tiết kiệm bộ nhớ và tính toán.

C. LƯU ĐỒ THUẬT TOÁN ĐỀ XUẤT SHUIM_HE [25]

Mỗi voi (cá thể) trong quần thể được biểu diễn dưới dạng vector nhị phân, tương ứng với một tập mục ứng viên. Giả sử tập mục là I = {A, B, C, D, E}.

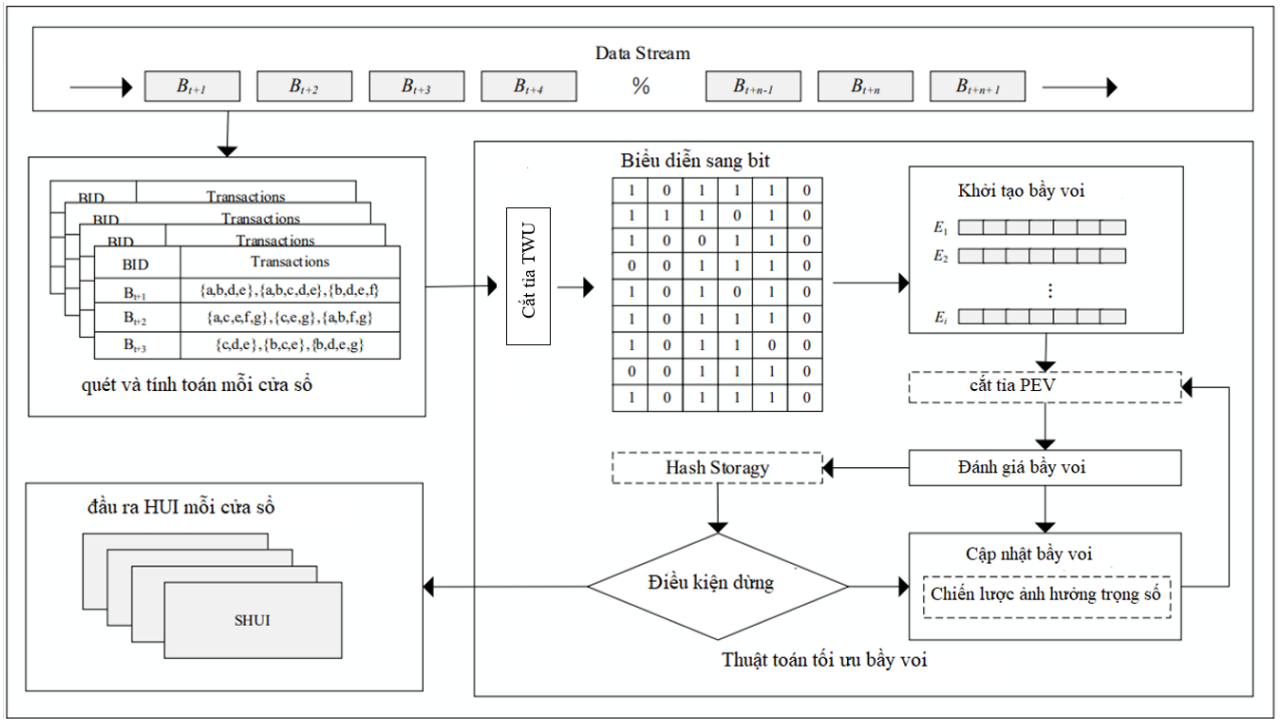
Voi x_1 = [1, 0, 1, 0, 0] đại diện cho tập mục {A, C}.

Voi x_2 = [1, 1, 0, 0, 1] đại diện cho tập mục {A, B, E}

Đối với luồng dữ liệu đầu vào DS đã cho, thuật toán trước tiên sẽ quét các dữ liệu mới đến và chèn các giao dịch vào cửa sổ trượt theo từng lô. Khi số lượng lô (batch_size) bằng với số lượng cửa sổ (win_size), thông tin của cửa sổ hiện tại sẽ được xử lý.

Đối với dữ liệu trong cửa sổ hiện tại, thuật toán trước tiên sử dụng chiến lược cắt tỉa TWU để loại bỏ các tập mục 1 phần tử có độ hữu ích theo trọng số giao dịch thấp (1-LTWUIs) và chuyển đổi cơ sở dữ liệu đã được tổ chức lại thành dạng bitmap.

Mỗi vector vị trí của voi trong bầy được khởi tạo, sử dụng chiến lược cắt tỉa PEV để loại bỏ các vector không tiềm năng.



Hình 2. *Cấu trúc và quy trình của SHUIM_HE*

Thuật toán đánh giá từng vector vị trí của voi, lưu trữ giá trị độ thích nghi (fitness) đã tính toán cùng với tập mục vào bảng băm (hash table), và xác định ra các cá thể ưu tú.

Thuật toán sử dụng chiến lược ảnh hưởng trọng số của các cá thể ưu tú để cập nhật bầy. Quá trình cập nhật và đánh giá bầy được lặp lại theo vòng lặp cho đến khi đạt đến số lần lặp tối đa, và sau đó các tập mục hữu ích (HUIs) của cửa sổ hiện tại sẽ được xuất ra.

Khi các giao dịch mới đến, các lô dữ liệu cũ sẽ bị xóa, và các lô mới sẽ được thêm vào cửa sổ. Thuật toán sẽ thực hiện một vòng đọc và xử lý dữ liệu mới như đã mô tả ở trên.

Mã giả của thuật toán được thể hiện trong thuật toán 1:

Thuật toán 1. SHUIM-EH

Input: DS, minutil_percent, pop_size, max_iter, batch_size, win_size
Output: HUIs

1. Wb = ∅
2. **for** each new batch Bt in DS **do**
3. numbatch++
4. Tính **TotalUtility** từng new batch Bt và đưa vào cửa sổ dữ liệu Wb,
5. Tính tổng **Độ hữu ích** của từng mục (item) trong Bt,
6. **if** numbatch = win_size **then**
7. Tính **TotalUtility** của cửa sổ dữ liệu Wb,
8. Tính **Độ hữu ích toàn phần (TWU)** của từng mục (item) trong Wb,
9. get min_util (Wb)= TotalUtility (Wb) × δ
10. Init (); //thuật toán 2
11. SHUI = ø, iter = 1

```
12.                    while iter≤max_iter do
13.                        for i=1 to pop_size do
14.                            X=IS(Ei);
15.                            if u(X)≥min_util &X ∉ SHUI then
16.                                X →SHUI và chọn ra những cá thể xuất sắc;
17.                            end if
18.                            Next_Clan (); // thuật toán 3
19.                        end for
20.                        iter++
21.                    end while
22.                    Xóa lô cũ;
23.                end if
24.        end for
```

Đầu vào của thuật toán gồm: Luồng dữ liệu: DS; ngưỡng hữu ích tối thiểu (theo phần trăm): minutil_percent; kích thước quần thể: pop_size; số vòng lặp tối đa: max_iter; số lượng lô dữ liệu trong cửa sổ trượt: win_size; số giao dịch trong mỗi lô: batch_size. Đầu ra. Tập các tập mục hữu ích cao trong mỗi cửa sổ trượt: HUIs. Khi luồng dữ liệu đến, các bước từ 1 đến 5 của thuật toán sẽ thêm các lô dữ liệu mới vào cửa sổ trượt Wb, tính tổng độ hữu ích của từng new batch Bt và tính độ hữu ích có trọng số theo giao dịch của các tập mục đơn (1-itemsets). Nếu số lượng lô đạt đến kích thước cửa sổ, thuật toán sẽ: tính tổng độ hữu ích của cửa sổ; và tính độ hữu ích có trọng số theo giao dịch của các tập mục đơn (1-itemsets).

Bước 10: Sử dụng hàm Init () để khởi tạo đối tượng thuật toán.

Bước 11: Khởi tạo tập SHUI rỗng và đặt số vòng lặp ban đầu là 1.

Bước 12 đến 17: thuật toán sử dụng vector vị trí từ đối tượng lưu trữ băm (hash storage strategy); đánh giá giá trị fitness (mức độ thích nghi); cập nhật tập kết quả các tập mục hữu ích cao. Nếu độ hữu ích không nhỏ hơn ngưỡng tối thiểu và không thuộc tập các tập mục hữu ích cao (HUIs) trong SHUI, thì vector vị trí sẽ được chuyển đổi thành tập mục tương ứng, và tập mục cùng với giá trị độ hữu ích tương ứng sẽ được lưu vào SHUI.

Bước 18 sử dụng chiến lược cập nhật bị ảnh hưởng bởi trọng số của con voi cái ưu tú trong hàm Next_Clan () để cập nhật bầy (clan). Vòng lặp while sẽ lặp lại quá trình cập nhật quần thể cho đến khi đạt đến số vòng lặp tối đa. Sau đó, thuật toán xóa lô dữ liệu cũ để tiếp tục với việc quét lô dữ liệu mới trong cửa sổ trượt tiếp theo.

Thuật toán 2. Init ()

Input: Transactions of (Wb), size of population(pop_size)

Output: Nhóm cá thể đầu tiên

```
1.      Quét DB để xác định 1-HTWUIs {ij | TWU (ij)≥min_util} và loại bỏ 1-LTWUIs để sinh ra tập giao dịch mới
được tổ chức lại thành TD;
2.      Chuyển TD thành dạng bitmap
3.      for i to pop_size do
4.          len=|1-HTWUIs |
5.          Sinh ngẫu nhiên số numi <len
6.          Tạo một vectơ bit Ei with numi bits to 1 lựa chọn bằng cách quay vòng
7.          if numi >1 then
8                  Ei=PEV_Check (Ei);
9.          end if
10.     end for
```

Thuật toán 2 xử lý dữ liệu cửa sổ của các lô mới đến để cung cấp quần thể khởi tạo ban đầu cho việc cập nhật bộ tộc. đầu vào Transactions of (Wb): Tập các giao dịch trong cửa sổ dữ liệu đang xử lý; pop_size: Kích thước quần thể. Đầu ra The first clan of solutions: Bộ tộc đầu tiên

Bước 1: Quét cơ sở dữ liệu để xác định tất cả các tập mục có độ hữu ích theo trọng số giao dịch cao (1-HTWUIs), tức là các mục đơn ij sao cho TWU (ij) ≥ min_util. Sau đó loại bỏ các tập mục có độ hữu ích thấp (1-LTWUIs). Kết quả của bước này là tập giao dịch mới đã được tổ chức lại, gọi là TD.

Bước 2: Biểu diễn lại TD dưới dạng bitmap, tức là mỗi giao dịch được mã hóa thành một dãy bit cho dễ xử lý và tính toán.

Bước 3–10: Lặp lại pop_size để tạo ra các cá thể (voi) cho quần thể ban đầu:

Bước 4: Tính độ dài len, là số lượng tập mục 1-HTWUI được tìm thấy.

Bước 5: Sinh một số ngẫu nhiên num_i nhỏ hơn len (tức num_i < len).

Bước 6: Tạo một vectơ bit E_i, trong đó có đúng num_i bit được chọn và đặt thành 1. Việc chọn các bit này được thực hiện bằng cách quay vòng, dựa vào xác suất liên quan đến độ hữu ích.

Bước 7–9: Nếu num_i > 1 (tức vectơ có từ 2 bit trở lên), thì thực hiện bước kiểm tra PEV (PEV_Check(Ei)) để đánh giá và loại bỏ các vectơ không hiệu quả.

Bước 10: Kết thúc vòng lặp. Sau pop_size lần lặp, ta thu được một quần thể đầu tiên gồm các vectơ vị trí tương ứng với các cá thể của bộ tộc đầu tiên.

Thuật toán 3. Next_Clan ()

Input: bộ tộc hiện tại

Output: bộ tộc tiếp theo

1. Tính fitness ($x_{ci,j}$) và tìm ra các cá thể voi cái xuất sắc// sử dụng công thức (11)
2. Tính w_i; // sử dụng công thức (12)
3. **for** i=1 to pop_size **do**
4. **while** k < N **do**
5. sinh ra n_i không trùng lắp; // sử dụng công thức (10)
6. Chọn ngẫu nhiên chọn n_i bit trong vectơ Ei để chuyển từ 0 thành 1 hoặc từ 1 thành0
7. end while
8. end for

Thuật toán 3 sử dụng chiến lược cập nhật bộ tộc có trọng số, trong đó trọng số chịu ảnh hưởng bởi các cá thể xuất sắc, nhằm tạo ra quần thể kế tiếp.

Ở bước đầu tiên, thuật toán tính giá trị độ thích nghi cho từng cá thể trong quần thể hiện tại bằng công thức (11). Những cá thể có độ thích nghi đạt ngưỡng độ hữu ích tối thiểu (min_util) sẽ được đưa vào tập SHUI, đồng thời các cá thể voi cái ưu tú cũng được lựa chọn từ đây.

Bước thứ hai, thuật toán áp dụng công thức (12) để tính trọng số cho các cá thể xuất sắc vừa được chọn.

Từ bước 4 đến bước 7, vòng lặp while sẽ sử dụng công thức (10) để từng bước sinh ra các cá thể mới cho quần thể kế tiếp. Quá trình này tiếp tục cho đến khi đạt đủ kích thước quần thể mong muốn, và thuật toán kết thúc.

D. CHIẾN LƯỢC CẬP NHẬT BỘ TỘC VỚI TRỌNG SỐ CHỊU ẢNH HƯỞNG BỞI CÁ THỂ XUẤT SẮC

Để khai thác hiệu quả các tập mục hữu ích cao (HUIs) từ luồng dữ liệu, bài báo này đề xuất một chiến lược cập nhật bộ tộc với trọng số chịu ảnh hưởng bởi các cá thể xuất sắc.

Chiến lược này thiết kế một công thức cập nhật bộ tộc mới (công thức 10). Trong đó, vectơ vị trí $x_{ci,j}$ của con voi thứ j trong bộ tộc c_i được so sánh với voi cái tốt nhất $x_{gbest,ci}$ thông qua độ chênh lệch bit. Như đã đề cập, vị trí của mỗi con voi j trong bộ tộc ci chịu ảnh hưởng từ các cá thể voi cái. Khi trong bộ tộc có nhiều cá thể voi cái, mỗi cá thể sẽ có mức độ ảnh hưởng khác nhau lên các thành viên còn lại. Nói chung, cá thể càng ưu tú thì trọng số ảnh hưởng càng cao. Vì vậy, thuật toán đã thiết kế công thức cập nhật (10) [24] để phù hợp với không gian tìm kiếm nhị phân

$$x_{n,ci,j} = k \times w_i \times \left| x_{gbest,ci} \oplus x_{ci,j} \right| + \eta \times x_{ci} \quad (10)$$

wi ∈ [0,1] là hệ số trọng số, phản ánh mức độ ảnh hưởng của các cá thể xuất sắc; k là một số nguyên dương ≥ 1, biểu thị tham số ảnh hưởng của cá thể xuất sắc; η là một số ngẫu nhiên nằm trong khoảng [0,1].

Công thức (11) [24] đánh giá từng cá thể voi trong bộ tộc hiện tại, trong đó $fitness(x_{ci,j})$ biểu thị giá trị độ thích nghi của con voi thứ j trong bộ tộc c_i, và giá trị này bằng với độ hữu ích của tập mục tương ứng với $x_{ci,j}$.

Nếu giá trị độ hữu ích của cá thể này lớn hơn hoặc bằng ngưỡng hữu ích tối thiểu (min_util), thì cá thể đó được xem là cá thể xuất sắc và được đưa vào tập các tập mục hữu ích (SHUI)

$$fitness(x_{ci,j}) = u(X) \quad (11)$$

$$w_i = \frac{fitness_{best,ci}}{\sum_{j=1}^{|SHUI|} fitness_{ci,j}} \quad (12)$$

Giả sử kích thước quần thể hiện tại là các vectơ vị trí như $x_{ci,1}$ = 101101 → tương ứng với tập mục <acdf>; $x_{ci,2}$ = 101110 → tương ứng với tập mục <acde>; $x_{ci,3}$ = 101101 → tương ứng với tập mục <bdf>. Các giá trị độ hữu ích tương ứng được tính theo công thức (12) là: $x_{ci,1}$: 68; $x_{ci,2}$: 126; $x_{ci,3}$: 89. Giả sử ngưỡng hữu ích tối thiểu (min_util) là 80, thì: $x_{ci,2}$, $x_{ci,3}$ là các cá thể xuất sắc trong quần thể hiện tại, và voi cái tốt nhất $\boldsymbol{x_{gbest,ci}}$ là $x_{ci,2}$ = 101110.

E. VÍ DỤ MINH HỌA THUẬT TOÁN SHUIM_EH

Để minh họa quá trình hoạt động của thuật toán SHUIM_EH, ta xét một luồng dữ liệu được phân chia thành các cửa sổ trượt. Giả sử cửa sổ B_{t+1} bao gồm ba giao dịch như Bảng 4:

Bảng 4. *Ví dụ dữ liệu giao dịch*

BID	Giao dịch
T1	{a, b, c, d, e}
T2	{a, b, c, d, e}
T3	{b, d, e, f}

Giả sử giá trị hữu ích của mỗi mục được định nghĩa: **a = 5, b = 4, c = 2, d = 3, e = 1, f = 6.** Và ngưỡng hữu ích tối thiểu được đặt là **min_util = 18**

Bước 1: Quét và tính TWU

Thuật toán tính tổng hữu ích của mỗi giao dịch và sau đó xác định **TWU** (Transaction Weighted Utility) của từng mục dựa trên tổng hữu ích của các giao dịch chứa mục đó.

Tổng hữu ích mỗi giao dịch: T1, T2: 5 + 4 + 2 + 3 + 1 = 15, T3: 4 + 3 + 1 + 6 = 14.

TWU từng mục: TWU(a) = 15 + 15 = 30, TWU(b) = 15 + 15 + 14 = 44, TWU(c) = 15 + 15 = 30, TWU(d) = 15 + 15 + 14 = 44, TWU(e) = 15 + 15 + 14 = 44, TWU(f) = 14. Sau bước cắt tỉa TWU, loại bỏ mục f vì TWU(f) < min_util. Tập mục còn lại: X **= {a, b, c, d, e}**

Bước 2: Biểu diễn nhị phân:

Các giao dịch được chuyển sang dạng bit-vector theo tập mục đã cắt tỉa như Bảng 5

Bảng 5. *Bit - vector*

Giao dịch	**a**	**b**	**c**	**d**	**e**
T1	1	1	1	1	1
T2	1	1	1	1	1
T3	0	1	0	1	1

Bước 3: Khởi tạo bầy voi

Mỗi cá thể voi được khởi tạo là một vector nhị phân biểu diễn một tập mục ứng viên. Ví dụ:

E1= [1,0,1,0,0] → {a, c}

E2= [0,1,0,1,1] → {b, d, e}

Bước 4: Đánh giá bầy voi

Tiến hành tính utility cho từng cá thể dựa trên tập mục tương ứng:

E1= {a, c}: xuất hiện trong T1 và T2. Tổng utility = (5 + 2) × 2 = 14 < min_util → bị loại.

E2= {b, d, e}: xuất hiện trong cả ba giao dịch. Tổng utility = (4 + 3 + 1) × 3 = 24 ≥ min_util → **chấp nhận**

Bước 5: Cập nhật vị trí đàn voi

Cá thể không đạt min_util sẽ được cập nhật vị trí dựa trên cá thể đầu đàn theo công thức (10). Cơ chế cắt tỉa PEV (Pruned Elephant Vector) được áp dụng nhằm loại bỏ các tổ hợp kém tiềm năng trước khi đánh giá, giúp giảm thời gian xử lý.

Bước 6: Kiểm tra điều kiện dừng và xuất kết quả
Thuật toán dừng sau số vòng lặp xác định hoặc khi không còn cải thiện đáng kể. Các tập mục thỏa mãn min_util được xuất ra làm kết quả HUI của cửa sổ B_{t+1}. Kết quả tập mục **{b, d, e}** được ghi nhận là HUI.

V. ĐÁNH GIÁ THỰC NGHIỆM

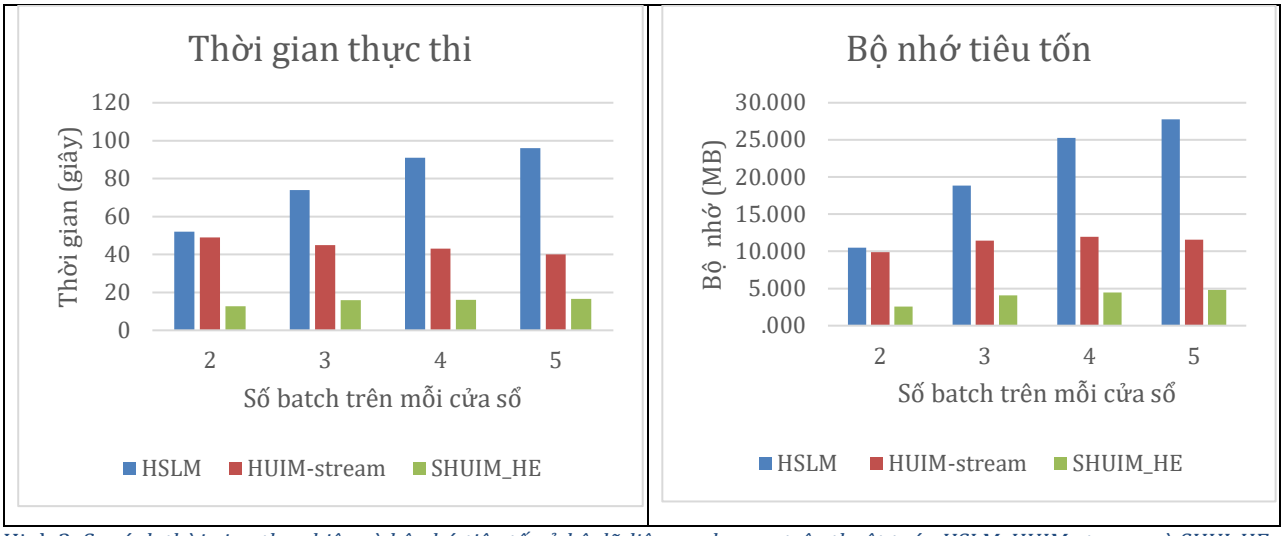
A. ĐÁNH GIÁ THỜI GIAN THỰC HIỆN VÀ TIÊU TỐN BỘ NHỚ

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 các bộ dữ liệu trên website: https://www.philippe-fournier-viger.com/spmf như ở Bảng 4.

Bảng 6. *Tập các bộ dữ liệu*

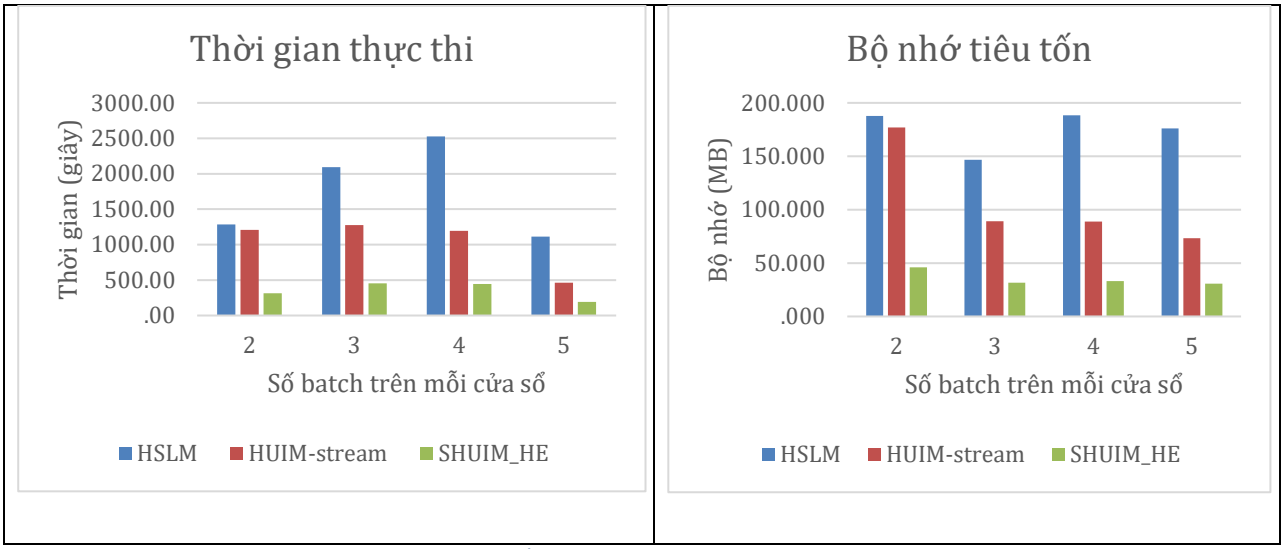
Dataset	Số lượng bit	Số item trung bình	Số dòng	Số transaction/batch
Mushroom	119	23	8416	1,000
Accidents	468	34	340,183	40,000
Retail	16,470	10	88,162	10,000
Chess	75	37	3,196	380
Chainstore	46,086	7.23	1,112,949	100,000

Đầu tiên, bài báo tiến hành so sánh hiệu quả về mặt thời gian thực hiện của thuật toán đề xuất (**SHUIM_HE**) với thuật toán ***HSLM*** [9] và thuật toán HUIM-stream [8] 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 3 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 tùy theo số lượng dòng của dataset thể hiện ở cột 5 của Bảng 3. Giá trị ngưỡng *minutil_percent* được chọn là 0.25%. Cùng với các tham số cơ bản của thuật toán: số thế hệ tối đa (generations) là 20, kích thước quần thể (*population_size*) là 100. Kết quả thực nghiệm cho thấy rằng thuật toán SHUIM_HE hiệu quả hơn rất nhiều so với thuật toán HSLM và HUIM-stream 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. Tiếp theo, bài báo thực nghiệm so sánh về bộ nhớ của HSLM, HUIM-stream và **SHUIM_HE** trên bộ dữ liệu của mushroom. Kết quả trong Hình 3 thể hiện tiêu tốn ít bộ nhớ và ổn định hơn do phát sinh một số lượng xác định quần thể và số thế hệ ngay khi số lượng giao dịch.

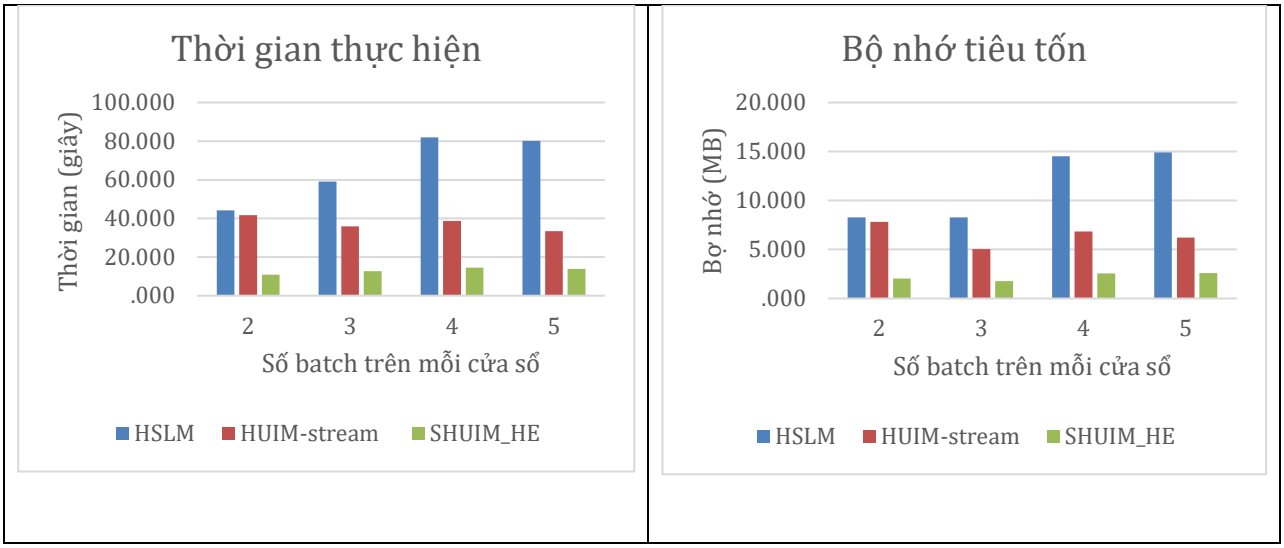


Hình 3. *So sánh thời gian thực hiện và bộ nhớ tiêu tốn ở bộ dữ liệu mushroom trên thuật toán HSLM, HUIM-stream và SHUI_HE*

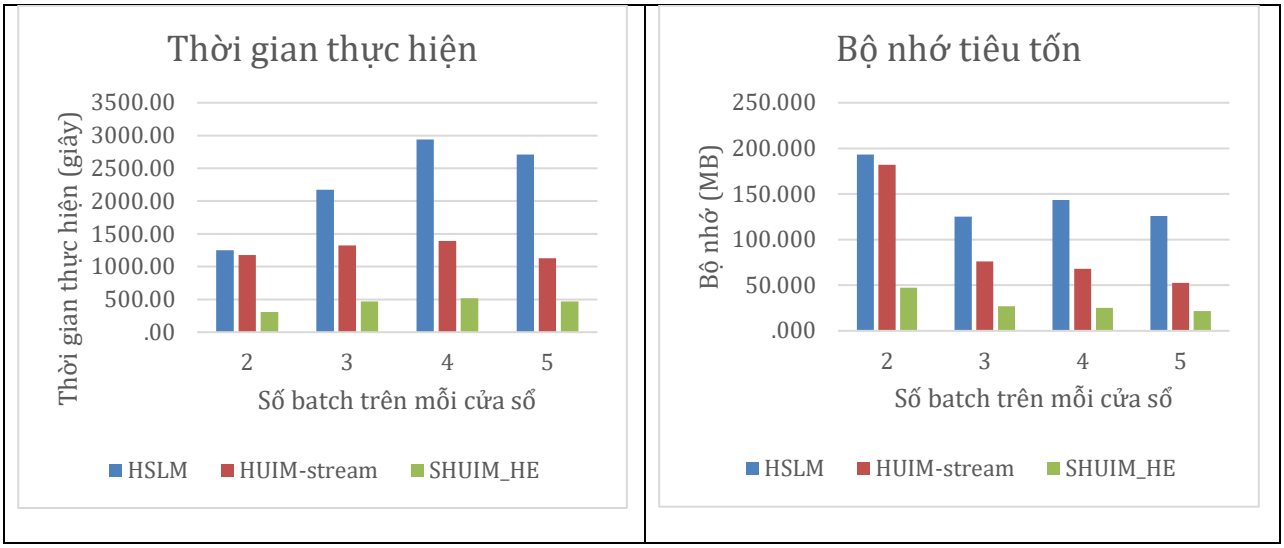
Tiếp tục so sánh trên các bộ dữ liệu Accidents, Retail va Chess đều cho thấy thời gian thực thi và bộ nhớ tiêu tốn của HUIM_GA nhỏ hơn nhiều so với HSLM và HUIM-stream.



Hình 4. *So sánh thời gian thực hiện và bộ nhớ tiêu tốn ở bộ dữ liệu accidents trên thuật toán HSLM, HUIM-stream và SHUIM_HE*



Hình 5. *So sánh thời gian thực hiện và bộ nhớ tiêu tốn ở bộ dữ liệu Chess trên thuật toán HSLM, HUIM-stream và SHUIM_HE*



Hình 6. *So sánh thời gian thực hiện và bộ nhớ tiêu tốn ở bộ dữ liệu retail trên thuật toán HSLM, HUIM-stream và SHUIM_HE*

B. ĐÁNH GIÁ ĐỘ CHÍNH XÁC VÀ CHẤT LƯỢNG KẾT QUẢ

Để đánh giá chất lượng phát hiện tập mục hữu ích cao (HUIs), chúng tôi so sánh:

Số lượng HUIs phát hiện được bởi mỗi thuật toán.

Precision = TP / (TP + FP).

Recall = TP / (TP + FN)

F1-score = 2 × (Precision × Recall) / (Precision + Recall)

Bảng 7. So sánh chất lượng kết quả giữa SHUIM_HE, HSLM và HUIM-Stream

Dataset	Thuật toán	#HUIs	Precision	Recall	F1-score
Mushroom	HSLM	920	0.954	0.968	0.961
	HUIM-stream	900	0.947	0.947	0.947
	SHUIM_HE	948	**0.997**	**0.998**	**0.998**
Accident	HSLM	800	0.961	0.940	0.950
	HUIM-stream	790	0.955	0.936	0.945
	SHUIM_HE	824	**0.993**	**0.992**	**0.993**
Retail	HSLM	720	0.976	0.970	0.973
	HUIM-stream	715	0.970	0.964	0.967
	SHUIM_HE	735	**0.993**	**0.993**	**0.993**
Chess	HSLM	1150	0.958	0.958	0.958
	HUIM-stream	1125	0.940	0.938	0.939
	SHUIM_HE	1175	**0.979**	**0.960**	**0.969**

Các kết quả thực nghiệm trên bốn tập dữ liệu benchmark (Mushroom, Accidents, Retail và Chess) ở Bảng 7 cho thấy rằng thuật toán đề xuất SHUIM_HE có hiệu năng vượt trội so với các phương pháp truyền thống HSLM và HUIM-Stream trên cả ba tiêu chí: thời gian thực thi, mức tiêu thụ bộ nhớ, và chất lượng tập mục phát hiện. Cụ thể, SHUIM_HE giúp giảm thời gian xử lý nhờ loại bỏ bước phát sinh tập con ứng viên, đồng thời tiêu tốn bộ nhớ thấp và ổn định hơn nhờ cấu trúc quần thể được xác định trước. Về độ chính xác, SHUIM_HE đạt Precision và Recall cao, với F1-score luôn lớn hơn 0.96 trên mọi bộ dữ liệu. Những kết quả này xác nhận rằng SHUIM_HE là một phương pháp hiệu quả và đáng tin cậy trong khai thác tập mục hữu ích cao trên luồng dữ liệu, đặc biệt phù hợp với môi trường có khối lượng dữ liệu lớn, tính động cao, và yêu cầu tối ưu tài nguyên tính toán.

VI. KẾT LUẬN

Bài báo này trình bày một nghiên cứu về khai thác tập hữu ích cao dựa trên bầy voi. Đối mặt với những thách thức của bài toán khai thác tập mục hữu ích cao (HUIM) bài báo đề xuất thuật toán có tên SHUIM_EH. Cụ thể: trước tiên sử dụng Kỹ thuật cửa sổ trượt để cập nhật nhanh thông tin mới nhất từ luồng dữ liệu. Tiếp theo, bài báo đề xuất một chiến lược cập nhật bộ tộc có trọng số dựa trên cá thể xuất sắc, giúp tăng tốc độ hội tụ của thuật toán và giảm

thiểu hiện tượng mất tập mục do yếu tố ngẫu nhiên trong thuật toán heuristic gây ra. Cuối cùng, để nâng cao hiệu quả của thuật toán hơn nữa, một chiến lược lưu trữ bằng bảng băm được đề xuất, nhằm giải quyết vấn đề đánh giá lặp lại các tập mục và xử lý việc kiểm tra kết quả ở các lô dữ liệu trùng nhau.

Trong tương lai, nhóm nghiên cứu của chúng tôi sẽ tiếp tục tối ưu hóa thuật toán SHUIM_EH và giải quyết bài toán trôi khái niệm (concept drift) trong luồng dữ liệu.

VII. TÀI LIỆU THAM KHẢO

[1] Pazhaniraja, N., Sountharrajan, S. and Suganya, 2023 "Optimizing high-utility item mining using hybrid dolphin echolocation and Boolean grey wolf optimization," *Ambient Intell Human Comput 14,* p. 2327–2339.

[2] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong and Young-Koo Lee, 2009 "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases," *IEEE Transactions on Knowledge and Data Engineering,* pp. vol. 21, no. 12, pp. 1708-1721.

[3] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu and Philip S. Yu, 2013, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Transactions on Knowledge and Data Engineering,* pp. vol. 25, no. 8, pp. 1772-1786.

[4] Guo, SM., Gao and H. HUITWU, 2016 "An Efficient Algorithm for High-Utility Itemset Mining in Transaction Databases," *Journal of Computer Science and Technology,* p. 776–786.

[5] Liu M and Qu J, 2012, "Mining high utility itemsets without candidate generation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*.

[6] WANG S F, HAN M and JIA T, 2020, "Survey of high utility pattern mining over data streams," *Application Research of Computers,* pp. 2571-2578.

[7] Bijay Prasad Jaysawal and Jen-Wei Huang, 2020, "SOHUPDS: a single-pass one-phase algorithm for mining high utility patterns over a data stream," in *Proceedings of the 35th annual ACM symposium on applied computing*.

[8] Han M, Li M and Chen Z, 2023, "High utility pattern mining algorithm over data streams using ext-list," *Applied Intelligence,* pp. 27072-27095.

[9] Minh-Thai Tran, Anh-Duy Tran, Duc-Thanh Pham and Minh-Nguyen Le, 2024 "Method for mining high-utility patterns in transaction stream data based on linked list structure," *Journal on Information Technologies & Communications*.

[10] J. Chen, Y. Lin, ,Y. Lin and P. Fournier-Viger, 2022 "CoIUM: An efficient correlated high utility itemset mining algorithm," *arXiv preprint*.

[11] Kannimuthu S and Premalatha K, 2014, "Discovery of high utility itemsets using genetic algorithm with ranked mutation," *Applied Artificial Intelligence,* pp. 337-359.

[12] Luna J M, Kiran R U and Fournier-Viger P, 2023, "Efficient mining of top-k high utility itemsets through genetic algorithms[J]," *Information Sciences,* pp. 529-553.

[13] Q. Zhang,, W. Fang, J. Sun and Q. Wang, 2019, "Improved Genetic Algorithm for High-Utility Itemset Mining," *in IEEE Access*.

[14] L. J. C. W, G. W and F.-V. P, 2016, "High utility-itemset mining and privacy-preserving utility mining," *Perspectives in Science*.

[15] Lin J C W, Djenouri Y, Srivastava G and et al, 2021, "A predictive GA-based model for closed high-utility itemset mining," *Applied Soft Computing*.

[16] Lin J C W, Djenouri Y, Srivastava G and et al, 2022, "Efficient evolutionary computation model of closed high-utility itemset mining," *Applied Intelligence,* pp. 10604-10616.

[17] Pazhaniraja N, Basheer S and Thirugnanasambandam K, 2023, "Multi-objective Boolean grey wolf optimization based decomposition algorithm for high-frequency and high-utility itemset mining," *AIMS Math,* pp. 18111- 18140.

[18] S. R. Bhandari and H. Tiwari, 2022 "Sliding window-based high utility itemset mining on stream data using EGUI-tree," *IJIGSP.*

[19] A. Dawar and A. K. Sharma, 2018, "Top-k high utility itemset mining over data stream," *The Knowledge Engineering Review*.

[20] L. H. e. al, 2023, "Efficient mining of top-k high utility itemsets over data streams," *Journal of Computer Science and Technology*.

[21] M. Meng and H. Zhao, 2024"High utility sequential pattern mining over data streams using DUI-table," *SAGE Open*.

[22] Chen X, Zhai P and Fang Y, 2021, "High utility pattern mining based on historical data table over data streams," in *2021 4th International Conference on Data Science and Information Technology*.

[23] Amaranatha Reddy P and Hazarath Murali Krishna Prasad, 2021, "High Utility Item-set Mining from retail market data stream with various discount strategies using EGUI-tree," *Journal of Ambient Intelligence and Humanized Computing,* pp. 1-12.

[24] Meng Han, Feifei He, Ruihua Zhang and Chunpeng Li, 2024, "Mining High Utility Itemsets with Elephant Herding Optimization," *DOI:10.21203/rs.3.rs-3881656/v1,* 2024.

[25] Đ. T. Phạm, D. A. Trần and Le Minh Nguyện, 2025, "Khai thác tập mục hữu ích cao từ các luồng dữ liệu dựa trên di truyền," *Tạp chí Khoa học Huflit*.

HIGH UTILITY ITEMSET MINING METHOD FROM DATA STREAMS BASED ON THE ELEPHANT HERDING ALGORITHM

Le Thi Minh Nguyen, Pham Duc Thanh, Tran Anh Duy, Tran Minh Thai

ABSTRACT— High Utility Itemset Mining (HUIM) is a prominent research direction in the field of data mining. Traditional HUIM algorithms often struggle with the exponential explosion of the search space, leading to limitations in scalability. To address this issue, heuristic-based HUIM algorithms have attracted significant attention; however, these methods are prone to premature convergence, resulting in the omission of potentially valuable itemsets. To overcome these limitations, we propose a new algorithm named SHUIM_HE, based on the Elephant Herding Optimization (EHO) algorithm, to efficiently mine high-utility itemsets from data streams in resource-constrained environments. The core innovation of the algorithm lies in a position evolution strategy based on female elephant, which significantly reduces the search space and improves the algorithm's execution performance. Experiments conducted on real-world datasets demonstrate that the proposed algorithm outperforms state-of-the-art heuristic HUIM algorithms.

Keywords — High Utility Itemset Mining; Data Stream; Hash Table; Sliding Window; Elephant Herding

Trần Anh Duy nhận học vị thạc sĩ Khoa học máy tính Trường Đại học Khoa học tự nhiên năm 2017; 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.

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 Thành phố Hồ Chí Minh; lĩnh vực nghiên cứu đang quan tâm là Khai thác dữ liệu.

Lê Thị Minh Nguyện nhận học vị thạc sĩ Khoa học máy tính Đại học Quốc gia Thành phố Hồ Chi 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.

Trần Minh Thái là tiến sỹ Công nghệ thông tin (CNTT). Hiện tại, TS. Thái là giảng viên và trưởng bộ môn Hệ thống Thông tin thuộc khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ -Tin học TP.HCM. Lĩnh vực nghiên cứu của TS. Thái liên quan đến vấn đề khai thác dữ liệu, ẩn dữ liệu, xử lý dữ liệu lớn và nhận dạng.