KHAI THÁC CÁC TẬP MỤC HỮU ÍCH CAO DỰA TRÊN PHƯƠNG PHÁP TỐI ƯU BẦY ĐÀN DÙNG BITMAP

Phạm Đức Thành

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

phamducthanh@huflit.edu.vn

TÓM TẮT— Khai thác các tập mục hữu ích cao (HUIs) là một chủ đề nóng hổi hiện nay về khai thác dữ liệu. Các thuật toán tiến hóa trong tự nhiên đang ngày càng thu hút chú ý, vì chúng có lợi thế tránh bùng nổ tổ hợp không gian tìm kiếm. Trong số các thuật toán tiến hóa trong tự nhiên được sử dụng để khai thác HUIs, thuật toán tối ưu bầy đàn (PSO) là phổ biến nhất. PSO khai thác HUI dựa trên sự chuyển đổi vị trí không ngừng theo hàm sigmoid cho vận tốc. Trong bài báo này, chúng tôi đề xuất một thuật toán HUIM dựa trên bộ PSO (S-PSO) được goi là HUIM-SPSO, trong đó chủ yếu xem xét các phần tử ở các vị trí có vận tốc lớn. Chúng tôi thực hiện mô hình hóa HUIM bằng S-PSO và giải thích HUIM-SPSO một cách chi tiết. Để phản ánh được sự đa dạng của các kết quả khai thác, chúng tôi đề xuất thước đo bit để chỉnh sửa khoảng cách. Kết quả thực nghiệm cho thấy thuật toán HUIM-SPSO hiệu quả và có thể khám phá nhiều HUI hơn với mức độ đa dạng cao.

Từ khóa— Khai phá dữ liệu, tập mục hữu ích cao, tối ưu dựa trên bầy đàn, khoảng cách chỉnh sửa bit.

I. GIỚI THIỆU

Khai thác tập phổ biến hữu ích cao (HUIM) [9, 12] là một phần của khai thác tập phổ biến được sử dụng để khám phá các tập mục có lợi nhuận cao bằng cách xem xét cả số lượng và giá trị của một hạng mục. Hầu hết các thuật toán HUIM hiện có, được thiết kế để khám phá tất cả các tập mục hữu ích cao (HUIs). Vì sự bùng nổ tổ hợp các hạng mục trong không gian tìm kiếm của tất cả các HUI, nên hiệu suất của các thuật toán chính xác có xu hướng nhanh chóng suy giảm theo kích thước của cơ sở dữ liệu (CSDL) và trở nên không thể chấp nhận được trong CSDL lớn. Hơn nữa, đối với các ứng dụng như hệ thống khuyến nghị, không cần sử dụng tất cả các HUI [14].

Một số thuật toán tiếp cận HUIM đã được đề xuất sử dụng các thuật toán tiến hóa (EA), chẳng hạn như thuật toán di truyền (GA) [3], tối ưu hóa bầy đàn (PSO) [5, 6], và đàn ong nhân tạo (ABC) [10]. Mặc dù các thuật toán này có thể khám phá các tập phổ biến hợp lý trong thời gian có thể chấp nhận được, việc xác định làm thế nào để nhận diện nhiều hơn HUI trong một số lần lặp giới hạn là một thách thức lớn.

Thuật toán PSO là một trong những thuật toán EA được sử dụng rộng rãi trong các nghiên cứu trước đây [5, 6], vì vậy chúng tôi cũng xem xét bài toán HUIM theo hướng PSO. Khác với mã hóa nhị phân sơ đồ PSO [4] sử dụng cho HUIM, được dựa trên bộ PSO (S-PSO) do Chen và đồng sự đề xuất [1] được tận dụng trong thuật toán của chúng tôi. Đối với S-PSO, vị trí được cập nhật theo cấu trúc của bộ cắt, để duy trì các vị trí có tốc độ cao.

Sử dụng S-PSO, chúng tôi đề xuất một thuật toán HUIM hiệu quả được gọi là HUIM-SPSO. Trước tiên, chúng tôi xác định lại các hoạt động S-PSO cho vấn đề HUIM. Sau đó, chúng tôi mô tả thuật toán được đề xuất một cách chi tiết. Để thể hiện tính ưu việt của S-PSO đối với vấn đề HUIM, chúng tôi xác định khoảng cách chỉnh sửa bit để đo tính đa dạng của các kết quả khai thác. Các kết quả thử nghiệm cho thấy HUIM-SPSO không chỉ hiệu quả mà còn có thể khám phá nhiều HUI hơn với mức độ đa dạng cao.

Các phần tiếp theo được bố cục như sau. Phần 2 đánh giá các công trình liên quan. Phần 3 một số khái niệm về HUIM và SPSO. Phần 4 trình bày về thuật toán HUIM-SPSO. Phần 5 trình bày các thực nghiệm. Cuối cùng, phần 6 rút ra các kết luận.

II. CÁC CÔNG TRÌNH LIÊN QUAN

Các thuật toán trước đây theo quy trình hai giai đoạn. Giai đoạn đầu xác định các ứng viên HUI dùng mô hình TWU. Sau đó, giai đoạn hai lọc các HUI thực sự bằng cách quét CSDL gốc [9, 12]. Các thuật toán hai giai đoạn thường tạo ra nhiều ứng viên, dẫn đến không gian tìm kiếm khổng lồ và chi phí tính toán cao. Vì thế, những thuật toán 1 giai đoạn không có ứng viên được giới thiệu. Với mỗi thuật toán một pha, cấu trúc dữ liệu, chẳng hạn như danh sách hữu ích [8] và chuỗi danh sách hữu ích chính xác [7], được sử dụng khám phá rất hiệu quả các HUI.

Gần đây, các thuật toán tiến hóa được sử dụng để duyệt qua không gian tập mục ứng viên rộng lớn với thời gian chấp nhận được để khai thác các HUI. Hai thuật toán HUIM, HUPEUMU-GARM và HUPEWUMU-GARM, dựa vào thuật giải di truyền, được đề xuất trong [3]. Khác nhau giữa chúng là thuật toán thứ hai không yêu cầu một ngưỡng hữu ích tối thiểu. Vấn đề chính của hai thuật toán này là dễ dàng có xu hướng rơi vào tối ưu cục bộ.

Sử dụng tối ưu đàn kiến, Wu và đồng sự đề xuất một thuật toán HUIM khởi tạo một đồ thị định tuyến trước khi tất cả con kiến bắt đầu hành trình của chúng [13]. Một con kiến có thể phát sinh vài tập mục ứng viên trong suốt hành trình của nó. Vì thế, mỗi nút trong đồ thị định tuyến biểu diễn một tập mục cụ thể có thể được lượng giá để xác định xem có phải là HUI không.

Song và Huang nghiên cứu vấn đề HUIM từ quan điểm của thuật toán ABC [10]. Thuật toán HUIM-ABC được đề xuất để khám phá các HUI bằng cách lập mô hình các tập mục dưới dạng nguồn mật hoa. Với mỗi nguồn mật hoa, ba loại ong được sử dụng để lặp đi lặp lại việc tối ưu hóa tuần tự.

Mới đây nhất, Fournier và đồng sự đề xuất hai thuật toán HUIM dựa theo thuật toán leo đồi và luyện kim [15]. Thuật toán HUIM-HC khám phá các HUI bằng cách lập mô hình quần thể để tìm giải pháp tối ưu. Thuật toán HUIM-SA khám phá các HUI bằng cách lập mô hình quần thể để tìm giải pháp tối ưu, tuy nhiên tránh được tối ưu cục bộ của leo đồi. Song và đồng sự đề xuất thuật toán HUIM-FA dựa trên đàn cá [16] mô phỏng hành vi bầy đàn của chúng để khám phá các HUI.

Trong số các thuật toán tiến hóa khác nhau, thuật toán PSO là thuật toán được sử dụng rộng rãi trong HUIM. HUIM-BPSOsig [5] và HUIM-BPSO [6] là hai thuật toán dựa trên PSO để khai thác các HUI. HUIM-BPSO hoạt động tốt hơn HUIM-BPSOsig nhờ sử dụng cấu trúc cây OR/NOR.

Ngoài véc tơ vận tốc và vị trí trước đó được dùng bởi lược đồ mã nhị phân PSO trong các thuật toán hiện có, thì tập cắt cũng được dùng trong S-PSO để cập nhật vị trí. Vì thế, mỗi phần tử tương ứng đến các véc tơ vận tốc cao có xu hướng có nhiều cơ hội được giữ lại trong lần lặp tiếp theo, như thế có thể tạo những kết quả với đa dạng cao. Theo sự hiểu biết tốt nhất của chúng tôi thì S-PSO chưa được sử dụng trong HUIM.

III. MỘT SỐ KHÁI NIỆM CƠ BẢN

A. KHÁI NIỆM VỀ HUIM

Cho $I = \{i_1, i_2, \dots, i_m\}$ là một tập hợp hữu hạn các hạng mục. $X \subseteq I$ được gọi là tập mục. Cho $D = \{T_1, T_2, \dots, T_n\}$ là CSDL giao dịch. Mỗi giao dịch $T_i \in D$, với định danh duy nhất tid, là một tập con của I.

Độ hữu ích nội $q(i_p, T_d)$ biểu diễn cho số lượng hạng mục i_p trong giao dịch T_d. Độ hữu ích ngoại $p(i_p)$ là giá trị lợi nhuận của hạng mục i_p. Độ hữu ích của hạng mục i_p trong giao dịch T_d được định nghĩa như sau $u(i_p, T_d) = p(i_p) \times q(i_p, T_d)$. Độ hữu ích của tập mục X trong giao dịch T_d được định nghĩa như sau $u(X, T_d) = \sum_{i_p \in X \wedge X \subseteq T_d} u(i_p, T_d)$. Độ hữu ích của tập mục X trong D được định nghĩa như sau $u(X) = \sum_{X \subseteq T_d \wedge T_d \in D} u(X, T_d)$. Độ hữu ích giao dịch của giao dịch T_d được định nghĩa như sau $TU(T_d) = u(T_d, T_d)$. Để khai thác HUIs, ngưỡng hữu ích tối thiểu δ, được xác định bởi người dùng, được định nghĩa như là phần trăm của tổng giá trị TU của CSDL, trong đó giá trị hữu ích tối thiểu được định nghĩa như sau $min\_util = \delta \times \sum_{T_d \in D} TU(T_d)$. Một tập mục X được gọi là một HUI nếu $u(X) \geq min\_util$. Cho một CSDL giao dịch D, nhiệm vụ của HUIM là xác định tất cả các tập mục mà có độ hữu ích không nhỏ hơn min_util.

Độ hữu ích giao dịch có trọng số (TWU) của tập mục X là tổng của độ hữu ích giao dịch tất cả các giao dịch có chứa X, được định nghĩa như sau $TWU(X) = \sum_{X \subseteq T_d \wedge T_d \in D} TU(T_d)$ [9]. X là một tập mục TWU (HTWUI) nếu $TWU(X) \geq min\_util$. Một HTWUI với k hạng mục được gọi là k-HTWUI. Điều đó được chứng minh trong [9] rằng HTWUIs thỏa mãn thuộc tính đóng chặn dưới của giao dịch trọng số, và tất cả HUIs đều là HTWUIs.

Xem xét CSDL giao dịch trong bảng 1 và lợi nhuận trong bảng 2. Để thuận tiện, chúng tôi viết tập mục $\{C, E\}$ là CE. Trong CSDL ví dụ, độ hữu ích của hạng mục E trong giao dịch T_2 là $u(E, T_2) = 3 \times 1 = 3$, độ hữu ích của tập mục CE trong giao dịch T_2 là $u(CE, T_2) = u(C, T_2) + u(E, T_2) = 6 + 3 = 9$, và độ hữu ích của tập mục CE trong CSDL giao dịch là $u(CE) = u(CE, T_2) + u(CE, T_2) = 24$. Cho $min\_util = 35$, với $u(CE) < min\_util$, vậy CE không là một HUI. TU của T_2 là $TU(T_2) = u(ABCDE, T_2) = 18$, và những độ hữu ích của những giao dịch khác được trình bày trong cột thứ ba của bảng 1. TWU của tập mục CE là $TWU(CE) = TU(T_2) + TU(T_4) = 48$, vậy CE là một HTWUI.

Bảng 1. *CSDL ví dụ*

TID	Giao dịch	TU
T_1	$(B, 1), (C, 3), (D, 5)$	35
T_2	$(A, 4), (B, 1), (C, 1), (D, 1), (E, 1)$	18
T_3	$(A, 4), (C, 2), (D, 5)$	31
T_4	$(C, 2), (D, 5), (E, 1)$	30
T_5	$(A, 5), (B, 2), (D, 5), (E, 3)$	33
T_6	$(A, 3), (B, 1), (C, 1), (D, 1)$	14
T_7	$(D, 1), (E, 1), (F, 2)$	8

Bảng 2. *Bảng lợi nhuận*

Hạng mục	*A*	*B*	*C*	*D*	*E*	*F*
Lợi nhuận	1	2	6	3	3	1

B. KHÁI NIỆM VỀ S-PSO

PSO là thuật toán tiến hóa (EA) mô phỏng hành vi xã hội của đàn chim và đàn cá [4]. Trong thuật toán ban đầu, một vài cá thể được khởi tạo ngẫu nhiên. Mỗi cá thể di chuyển về phía giá trị tối ưu theo hai phương trình sau:

$$Vec_i(t+1) = w \times Vec_i(t) + c_1 \times r_1 \times \big(PBest_i - Pos_i(t)\big) + c_2 \times r_2 \times \big(GBest - Pos_i(t)\big), \quad (1)$$

$$Pos_i(t+1) = Pos_i(t) + Vec_i(t+1), \quad (2)$$

Trong đó $Vec_i(t)$ và $Vec_i(t+1)$ là những véc tơ vận tốc của cá thể thứ i tại lần lặp thứ t và t+1 tương ứng; $Pos_i(t)$ và $Pos_i(t+1)$ là những vị trí của cá thể thứ i tại lần lặp thứ t và t+1 tương ứng; $PBest_i$ là vị trí tốt nhất trước đó của cá thể thứ i; $GBest$ là vị trí tốt nhất hiện tại của tất cả các cá thể; 3 hằng w, c_1 và c_2 là các hệ số trọng số; và r_1, r_2 là những số phát sinh ngẫu nhiên trong phạm vi (0, 1).

Tất cả những cá thể cập nhật véc tơ vận tốc và vị trí liên tục cho đến khi giải pháp tốt nhất được tìm thấy hoặc đã đạt đến số lần lặp tối đa cho trước.

Việc cập nhật véc tơ vận tốc của S-PSO: trong S-PSO, một véc tơ vận tốc là một tập các xác suất khả năng của mỗi phần tử đang được dùng trong việc cập nhật vị trí [1]. Cho E là một tập của những giải pháp có thể. Vec được định nghĩa dựa trên E như sau:

$$Vec = \{p(e) | \ e \in E\}, \quad (3)$$

Sử dụng định nghĩa véc tơ vận tốc trong công thức 3, việc cập nhật của véc tơ vận tốc trong công thức 1 được thực hiện bởi bốn loại tính toán sau đây [1]:

Hệ số × véc tơ vận tốc: Công thức 1 gồm 3 phần. Phần đầu tiên thuộc về loại này và có thể được định nghĩa như sau:

$$c \times Vec = \{p*(e) | \ e \in E\}, \quad (4)$$

Trong đó:

$$p*(e) = \begin{cases} 1, if \ c \times p(e) > 1 \\ c \times p(e), otherwise \end{cases} \quad (5)$$

(1) Vị trí – vị trí: Trong công thức 1, phần thứ hai và phần thứ ba bao gồm tính toán sự khác biệt giữa vị trí tốt nhất cục bộ và vị trí hiện tại, giữa vị trí tốt nhất toàn cục và vị trí hiện tại. Cho Pos_A và Pos_B là 2 vị trí. Việc tính toán sự khác biệt giữa 2 vị trí được định nghĩa sau:

$$Pos_A - Pos_B = \{e \mid e \in Pos_A \wedge e \notin Pos_B\}, \quad (6)$$

(2) Hệ số × (vị trí – vị trí): Sau khi trừ đi vị trí hiện tại từ cả hai vị trí tốt nhất cục bộ và toàn cục, việc tính toán của 2 phần cuối cùng được phân vào loại này. Tương tự, có thể định nghĩa sau:

$$cE' = \{p'(e) \mid e \in E\}, \quad (7)$$

Trong đó:

$$p'(e) = \begin{cases} 1, if \ e \in E' \ and \ c > 1 \\ c, if \ e \in E' \ and \ 0 \le c \le 1 \\ 0, if \ e \notin E' \end{cases} \quad (8)$$

(3) Tổng của tập các xác suất: Tính toán cuối cùng là tổng của 3 véc tơ vận tốc. Đặt $Vec_1 = \{p_1(e) | e \in E\}$, $Vec_2 = \{p_2(e) | e \in E\}$ và $Vec_3 = \{p_3(e) | e \in E\}$ là 3 tập xác suất được xác định trong E, trong đó tổng của Vec_1, Vec_2 và Vec_3 được định nghĩa sau:

$$Vec_1 + Vec_2 + Vec_3 = \{max(p_1(e), p_2(e), p_3(e)) | \ e \in E\}, \quad (9)$$

Cập nhật vị trí của S-PSO. Sau khi cập nhật véc tơ vận tốc, một số phát sinh ngẫu nhiên $\alpha = (0,1)$ được khởi tạo cho mỗi cá thể. Với mỗi phần tử thứ j của Vec_i, nếu xác suất tương ứng của nó $p(e)$ không nhỏ hơn α, thì $p(e)$ được giữ lại, nghĩa là:

$$cut_\alpha(Vec_i^j) = \begin{cases} p(e), & if \ p(e) = Vec_i^j \wedge p(e) \ge \alpha \\ 0, & otherwise \end{cases} \quad (10)$$

Tập của véc tơ vận tốc Vec_i, với mỗi phần tử được tính toán bởi công thức 10, được gọi là tập cắt. Với tập cắt và vị trí trước đó, vị trí cập nhật trong S-PSO cũng được tính trong công thức 2.

C. MÔ HÌNH HUIM DÙNG S-PSO

Trong bài báo này, mỗi cá thể được biểu diễn bởi hai loại véc tơ để biểu diễn vận tốc và vị trí.

Định nghĩa 1. Cho SN là kích thước quần thể, N_c là số lượng của các 1-HTWUI, và tất cả các 1-HTWUI được sắp xếp theo một thứ tự tổng thể (ví dụ như thứ tự từ vựng) trong suốt toàn bộ việc xử lý khai thác dữ liệu. Một véc tơ vận tốc $V_i (1 \le i \le SN)$ là một véc tơ với N_c phần tử, và với mỗi phần tử V_i^j là một xác suất, tương ứng với vận tốc của phần tử 1-HTWUI thứ j, được dùng trong việc cập nhật vị trí; và một véc tơ vị trí $P_i (1 \le i \le SN)$ là một véc tơ nhị phân với N_c phần tử, và mỗi thành phần P_i^j hoặc là 0 hoặc là 1, cho biết phần tử 1-HTWUI thứ j nào là vắng mặt hay có mặt trong P_i tương ứng.

Với hai véc tơ này, một véc tơ vận tốc thay đổi theo vị trí trước đó, và một véc tơ vị trí thì thay đổi theo véc tơ vận tốc và đại diện một tập mục ứng viên mới. Cụ thể, nếu vị trí thứ j của một véc tơ vị trí chứa 1, hạng mục ở vị trí thứ j, dựa theo tổng thứ tự, thì sẽ có trong HUI tiềm năng; ngược lại, hạng mục này không được bao gồm và không phải là một HUI tiềm năng. Đối với một véc tơ vị trí P_i, bít thứ j của nó được khởi tạo hoặc là 1 hoặc 0 dùng bánh xe roulette để lựa chọn với xác suất.

$$p(P_i^j) = \frac{TWU(item_j)}{\sum_{k=1}^{N_c} TWU(item_k)}, \quad (11)$$

Trong đó N_c là số lượng của các 1-HTWUI.

Trong mỗi lần lặp của S-PSO, hàm thích nghi được tính toán để mô tả đặc trưng vấn đề tối ưu hóa. Gọi X là một tập mục được biểu diễn bởi một véc tơ vị trí P_i. Độ hữu ích của X được sử dụng như hàm thích nghi sau:

$$fitness(P_i) = u(X), \quad (11)$$

Chúng tôi cũng cần định nghĩa lại việc tính toán của loại tính toán thứ hai (vị trí – vị trí). Gọi P_a và P_b là hai véc tơ vị trí với N_c phần tử. Chúng tôi định nghĩa sau:

$$dP = P_a - P_b = \{dP_i | 1 \le i \le N_c\}, \quad (12)$$

Trong đó

$$dP_i = \begin{cases} 1, & if\ P_a^i = 1\ and\ P_b^i = 0 \\ 0, & otherwise \end{cases} \quad (13)$$

IV. THUẬT TOÁN HUIM-SPSO

A. BIỂU DIỄN THÔNG TIN HẠNG MỤC BẰNG BITMAP

Trong HUIM-SPSO chúng tôi dùng bitmap, là một biểu diễn hiệu quả thông tin của hạng mục trong thuật toán HUIM [11]. Cụ thể, các tập mục được biểu diễn bởi 1 bitmap tổng bao. Trong 1 bitmap tổng bao, có một bit cho mỗi giao dịch trong CSDL. Nếu hạng mục i xuất hiện trong giao dịch T_j thì bit j của bitmap tổng bao đối với hạng mục i được bật lên 1; ngược lại là 0. Điều này áp dụng cho tất cả tập mục. Cho X là một tập mục. $Bit(X)$ tương ứng với bitmap tổng bao để biểu diễn tập giao dịch đối với tập mục X. Gọi X và Y là hai tập mục. $Bit(X \cup Y)$ được tính toán như là $Bit(X) \cap Bit(Y)$, đó là phép toán bitwise-AND của $Bit(X)$ và $Bit(Y)$. Do đó, các giá trị hữu ích của các tập mục đích có thể được tính toán một cách hiệu quả bằng việc sử dụng toán tử bitwise.

B. THUẬT TOÁN HUIM-SPSO

Theo trình bày ở trên, thuật toán HUIM-SPSO dựa trên S-PSO (HUIM-SPSO) được trình bày trong thuật toán 1 dưới đây.

Algorithm 1 **HUIM-SPSO**
Input Kích thước quần thể SN, giá trị hữu ích tối thiểu min_util, số lần lặp tối đa max_iter.
Output Các hạng mục có độ hữu ích cao HUI

```
1.  Gọi hàm Init()
2.  iter = 1
3.  while iter ≤ max_iter do
4.      for i=0 to SN do
5.          Tạo ngẫu nhiên r1 và r2
6.          Tính Vi dùng công thức 1.
7.          Tạo ngẫu nhiên ∝
8.          for j=1 to Nc do
9.              if Vi^j <∝ then
10.                 Vi^j = 0
11.             end if
12.         end for
13.         Gọi hàm Position_update(Pi)
14.         X = IS(Pi)
```

15. **if** $u(X) \geq min\_util$ **and** $X \notin SHUI$ **then**
16. $X \rightarrow SHUI$
17. **end if**
18. $X_i = IS(Pbest_i)$
19. **if** $u(X) > u(X_i)$ **then**
20. $Pbest_i = P_i$
21. **end if**
22. **end for**
23. Tìm $GBest$ trong số SN cá thể
24. tăng $iter$ lên 1
25. **end while**
26. Trả về tất cả các HUI trong $SHUI$

Trong thuật toán 1, thủ tục khởi tạo (được mô tả trong thuật toán 2) được gọi trong bước 1. Sau đó, số bước lặp được gán bằng 1 (bước 2). Vòng lặp chính (từ bước 3 đến bước 25) lặp lại việc cập nhật các véc tơ vận tốc và vị trí cho đến khi đạt được số lần lặp tối đa. Vòng lặp từ bước 4 đến bước 22 xử lý từng cá thể riêng lẻ. Đối với mỗi cá thể P_i, bước 5 tạo ra hai số ngẫu nhiên trong phạm vi (0,1). Lưu ý chúng tôi thiết lập giá trị của w, c_1 và c_2 trong công thức 1 là 1 trong thuật toán này. Véc tơ vận tốc được tính toán trong bước 6. Bước 7 phát sinh 1 số ngẫu nhiên $\propto$ để hiệu chỉnh véc tơ vận tốc. Khi đó, véc tơ vận tốc V_i của từng cá thể được liệt kê được cập nhật trong vòng lặp từ bước 8 đến bước 12. Vị trí thì được cập nhật bằng cách gọi thủ tục được mô tả trong thuật toán 3. Bước 14 xác định tập mục tương ứng với cá thể đang liệt kê. Hàm $IS(\,)$ trả về tập mục X bằng cách hợp nhất những hạng mục trong P_i nếu giá trị của nó là 1. Tập mục được xác định mới được lưu trong $SHUI$ nếu nó là 1 HUI và chưa được phát hiện trước đây. $SHUI$ là tập của các HUI được khám phá. Bước 18 đến bước 21 cập nhật giá trị cục bộ tốt nhất của các cá thể đang liệt kê. $GBest$ được cập nhật cá thể tương ứng với các HUI được khám phá với giá trị độ hữu ích cao trong bước 23. Bước 24 tăng số lần lặp lên 1. Cuối cùng, bước 26 trả về các HUI được khám phá.

Algorithm 2	**Thủ tục Init( )**
Input	CSDL giao dịch ***D***, kích thước quần thể SN, giá trị hữu ích tối thiểu min_util
Output	Những cá thể của quần thể đầu tiên

1. Quét CSDL ***D*** lần một
2. Xoá những hạng mục không là 1-HTWUI
3. Biểu diễn tái cấu trúc lại CSDL như một bitmap
4. **for** i=0 to SN **do**
5. **for** j=0 to N_c **do**
6. Khởi tạo P_i^j với 0 hay 1 dùng công thức 11
7. **If** $P_i^j \neq 0$ **then**
8. $v_i^j = rand(\,)$
9. **end if**
10. **end for**
11. $PBest_i = P_i$
12. $X = IS(P_i)$
13. **if** $u(X) \geq$ min_$util$ **then**
14. $X \longrightarrow SHUI$
15. **end if**
16. **end for**
17. Tìm được $GBest$ trong SN cá thể.

Trong thuật toán 2, CSDL giao dịch được quét lần thứ nhất để xác định các 1-HTWUI (bước 1-2). Trong bước 3, biểu diễn bitmap của CSDL đã lược bớt được xây dựng. Vòng lặp chính (bước 4-16) phát sinh các cá thể riêng lẻ ban đầu. Vòng lặp từ bước 5 đến bước 10 khởi tạo từng phần tử véc tơ vị trí và vận tốc của các cá thể đang liệt kê. Hàm $rand(\,)$ trả về một số ngẫu nhiên trong khoảng (0,1). Lưu ý chúng tôi chỉ khởi tạo những phần tử mà giá trị của chúng là 1 trong véc tơ vị trí với 1 vận tốc ngẫu nhiên. P_i cũng được khởi tạo như $PBest_i$ trong bước 11. Bước 12 xác định tập mục tương ứng với cá thể đang liệt kê. Nếu cá thể hiện tại có thể thoả điều kiện là một HUI X (bước 13), bước 14 ghi nhận tập mục này. Cuối cùng, $GBest$ được khởi tạo trong bước 17.

Algorithm 3	**Position_update(*P*)**
Input	véc tơ vị trí P
Output	véc tơ vị trí P được cập nhật mới

1. Khởi tạo new_P với tất cả phần tử bằng 0
2. Phát sinh ngẫu nhiên 1 số dương k không lớn hơn N_c
3. **if** $|V| \geq k$ **then**
4. Phát sinh new_P bằng việc thiết lập k số 1 với những vị trí tương ứng với những véc tơ vận tốc không là 0.
5. **else**
6. Phát sinh new_P bằng việc thiết lập $|V|$ số 1 trong những vị trí tương ứng với véc tơ vận tốc không là 0
7. Thay đổi giá trị của $(k - |V|)$-bit của new_P từ 0 sang 1
8. **end if**
9. $P = new\_P$

Trong thuật toán 3, vị trí đã được xử lý được khởi tạo dưới dạng N_c bit 0 trong bước 1. Sau đó, vị trí mới được xây dựng theo cách thức xây dựng sau. Bước 2 xác định con số những bit được gán là 1. Vị trí mới đầu tiên học từ những phần tử trong véc tơ vận tốc tương ứng (bước 3-4). $|V_i|$ là số phần tử có giá trị không 0 trong V_i. Nếu quá trình xây dựng new_P chưa hoàn thành bằng cách chỉ xem xét phần tử bít 1 trong véc tơ vận tốc của nó, các 1-HTWUI khác được sử dụng để xây dựng một véc tơ vị trí mới (bước 5-8). Vị trí mới được cập nhật được xác định trong bước 9 cuối cùng.

C. VÍ DỤ MINH HOẠ

Chúng tôi dùng CSDL giao dịch ở bảng 1 và bảng lợi nhuận ở bảng 2 để giải thích. Sau lần quét CSDL đầu tiên, TWU của mỗi hạng mục được biểu diễn trong bảng 3 dưới đây.

Bảng 3. *TWU của mỗi hạng mục*

Hạng mục	A	B	C	D	E	F
TWU	96	100	128	169	89	8

Cho min_util = 35, khi $TWU(F) < min\_util$, hạng mục F bị xoá khỏi giao dịch T_7, và độ hữu ích của F bị loại khỏi TUs của T_7. CSDL được tái tổ chức lại rồi biểu diễn lại bởi 1 bitmap, như trong bảng 4 dưới đây.

Bảng 4. *Bitmap biểu diễn CSDL tái tổ chức lại*

	A	B	C	D	E
T'_1	0	1	1	1	0
T'_2	1	1	1	1	1
T'_3	1	0	1	1	0
T'_4	0	0	1	1	1
T'_5	1	1	0	1	1
T'_6	1	1	1	1	0
T'_7	0	0	0	1	1

Giả sử kích thước quần thể SN là 3. Vì số lượng của các 1-HTWUI là 5, có 5 phần tử trong cả hai véc tơ vận tốc và véc tơ vị trí. Theo công thức 11, có 3 véc tơ vị trí được phát sinh ngẫu nhiên: $P_1 = < 10111 >$, $P_2 = < 11001 >$ và $P_3 = < 11010 >$. Tiếp theo, 3 véc tơ vận tốc cũng được phát sinh ngẫu nhiên: $V_1 = \{0.52,0,0.87,0.01,0.15\}$, $V_2 = \{0.15,0.58,0,0,0.62\}$ và $V_3 = \{0.76,0.03,0,0.28,0\}$. Đối với quần thể đầu tiên, $PBest_i$ giống với P_i. Vì thế, $PBest_1 = < 10111 >$, $PBest_2 = < 11001 >$ và $PBest_3 = < 11010 >$. Với CSDL ví dụ, ta có thể thấy rằng 3 cá thể biểu diễn ACDE, ABE và ABD tương ứng. Ta có $u(ACDE) = 16$, $u(ABE) = 27$ và $u(ACDE) = 41$. Trong số 3 tập mục này, chỉ có ABD là một HUI, vậy $SHUI = \{ABD: 41\}$, trong đó con số sau dấu hai chấm chỉ độ hữu ích. Theo giá trị hữu ích này, $GBest$ là <11010>.

Tiếp theo chúng tôi lấy cá thể P_1 làm ví dụ. Theo công thức 13, $PBest_1 - P_1 = < 00000 >$ và $GBest - P_1 = < 01000 >$. Giả sử $r_1 = 0.15$ và $r_2 = 0.66$, theo công thức 7, $r(PBest_1 - P_1) = \{0,0,0,0,0\}$ và $r(GBest - P_1) = \{0,0.66,0,0,0\}$. Tiếp theo, dùng công thức 9, ta có véc tơ vận tốc mới: $V_1 = \{0.52,0,0.87,0.01,0.15\} + \{0,0,0,0,0\} + \{0,0.66,0,0,0\} = \{0.52, 0.66, 0.87, 0.01, 0.15\}$. Cho số phát sinh ngẫu nhiên $\propto$ là 0.04. Với V_1 hiện tại, chỉ phần tử thứ tư nhỏ hơn $\propto$, vì vậy V_1 được thay đổi là {0.52, 0.66, 0.87, 0, 0.15}. Tiếp theo k được khởi tạo ngẫu nhiên là 1, điều này chỉ ra rằng chỉ có 1 bit với giá trị 1 trong véc tơ vị trí mới. Vì phần tử thứ nhất, nhì, ba và cuối của V_1 thì không 0, chỉ 1 trong 4 bit này được gán thành 1 trong véc tơ vị trí mới. Giả sử bit đầu tiên được chọn ngẫu nhiên, véc tơ vị trí mới là $P_1 = < 10000 >$, biểu diễn tập mục A. Vì $u(A) = 16 < min\_util$, A không là HUI. Hơn nữa, vì $u(A) = u(ACDE)$ và $u(A) < u(ABD)$, nên $PBest_1$ và $GBest$ không thay đổi.

Trong cùng lần lặp, những cá thể thứ hai và ba cũng được xử lý tương tự. Sau đó lần lặp tiếp theo bắt đầu xử lý lần lượt các cá thể để khám phá các HUI cho đến khi đạt được số lần lặp đã cho trước.

V. ĐÁNH GIÁ THỰC NGHIỆM

A. MÔI TRƯỜNG THỬ NGHIỆM VÀ TẬP DỮ LIỆU

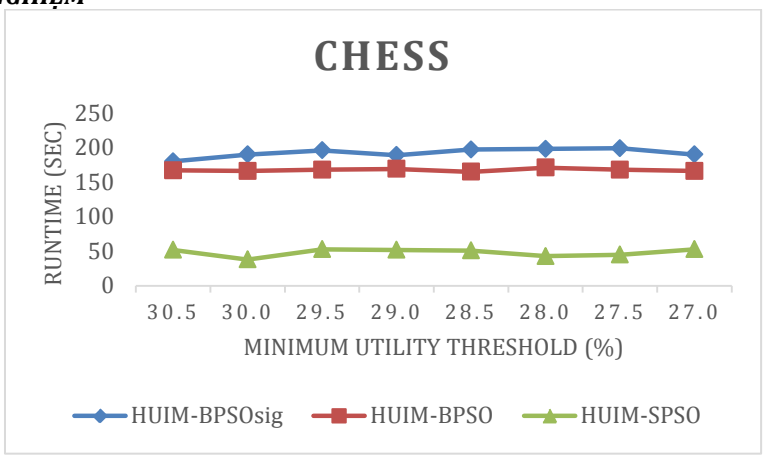
Các thử nghiệm được thực hiện trên máy tính CPU 2.90 GHz, 4 nhân và 32 GB bộ nhớ chạy trên hệ điều hành Microsoft Windows 10 64-bit. Chương trình được viết với ngôn ngữ lập trình Java, môi trường cài đặt Apache NetBeans IDE 12.5. Bốn bộ dữ liệu thực tế chúng tôi dùng để đánh giá thực nghiệm thuật toán. Các đặc trưng của các bộ dữ liệu được biểu diễn trong bảng 5 dưới đây.

Bảng 5. *Mô tả các CSDL được sử dụng*

Bộ dữ liệu	Độ dài trung bình của giao dịch	Số lượng các hạng mục	Số lượng giao dịch
Chess	37	76	3.196
Mushroom	23	119	8.124
Accidents_10%	34	469	34.018
Connect	43	130	67.557

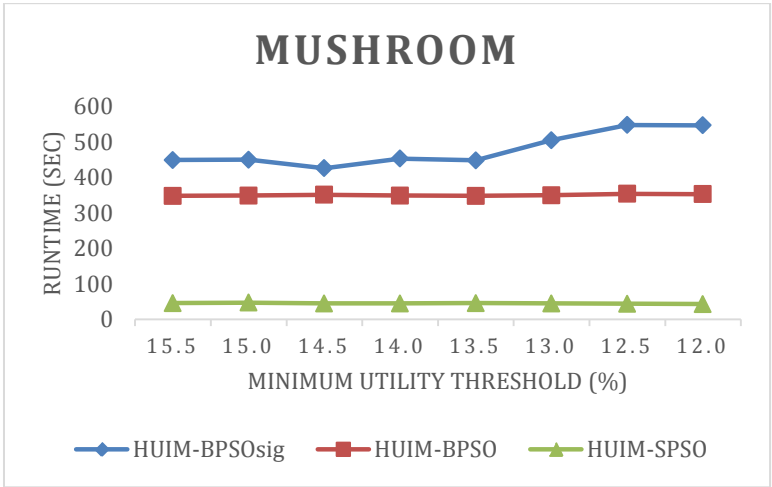
Hai thuật toán HUIM-BPSOsig [5] và HUIM-BPSO [6] được sử dụng để so sánh với thuật toán cải tiến HUIM-SPSO mà chúng tôi đề xuất, sử dụng từ thư viện SPMF [2] được xây dựng sẵn. Chúng tôi có hai đánh giá sẽ được thực hiện trên CSDL bảng 5, gồm có đánh giá thời gian thực thi chương trình và số lượng HUI được khám phá. Với kích thước quần thể là 20 và số lần lặp là 10000.

B. THỜI GIAN THỰC NGHIỆM



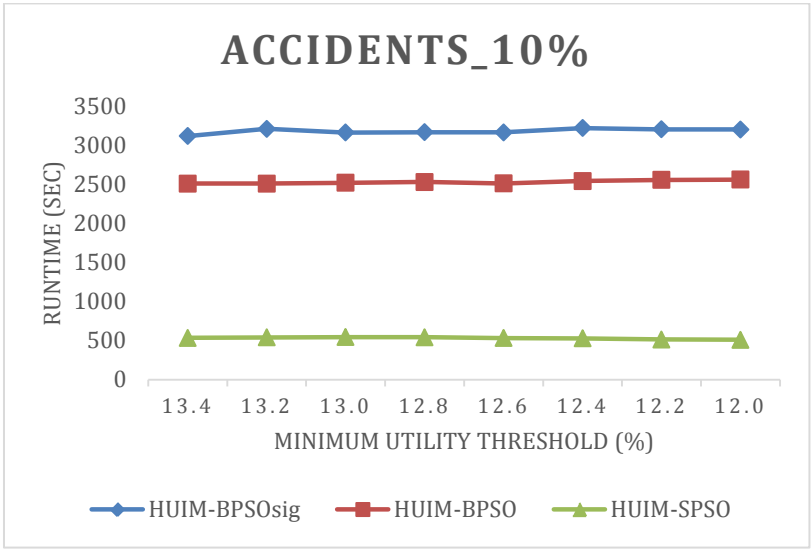
Hình 1. *Biểu diễn thời gian thực thi trên Chess dataset.*

Hình 1 biểu diễn thời gian thực thi của ba thuật toán: HUIM-BPSOsig (hình thoi màu xanh), HUIM-BPSO (hình vuông màu vàng) và HUIM-SPSO (hình tam giác màu xám) trên Chess dataset. Chess dataset là loại dữ liệu dày mật độ 49%, với ngưỡng minimun utility từ 30.5% đến 27%, HUIM-SPSO có thời gian chạy nhanh gấp 3 lần so với HUIM-BPSOsig và HUIM-BPSO.



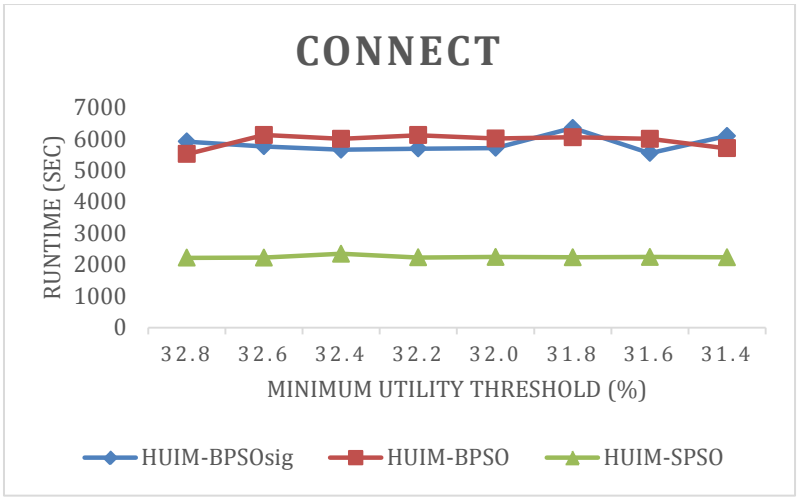
Hình 2. *Biểu diễn thời gian thực thi trên Mushroom dataset.*

Hình 2 biểu diễn thời gian thực thi của ba thuật toán: HUIM-BPSOsig (hình thoi màu xanh), HUIM-BPSO (hình vuông màu vàng) và HUIM-SPSO (hình tam giác màu xám) trên Mushroom dataset. Mushroom dataset là loại dữ liệu dày mật độ 19%, với ngưỡng minimun utility từ 15.5% đến 12%, HUIM-SPSO có thời gian chạy nhanh gấp 5 lần so với HUIM-BPSOsig và HUIM-BPSO.



Hình 3. *Biểu diễn thời gian thực thi trên Accidents_10% dataset.*

Hình 3 biểu diễn thời gian thực thi của ba thuật toán: HUIM-BPSOsig (hình thoi màu xanh), HUIM-BPSO (hình vuông màu vàng) và HUIM-SPSO (hình tam giác màu xám) trên Accidents_10% dataset. Do Accidents_10% dataset là loại dữ liệu dày mật độ 7%, với ngưỡng minimun utility từ 13.4% đến 12%, HUIM-SPSO có thời gian chạy nhanh gấp 5 lần so với HUIM-BPSOsig và HUIM-BPSO.



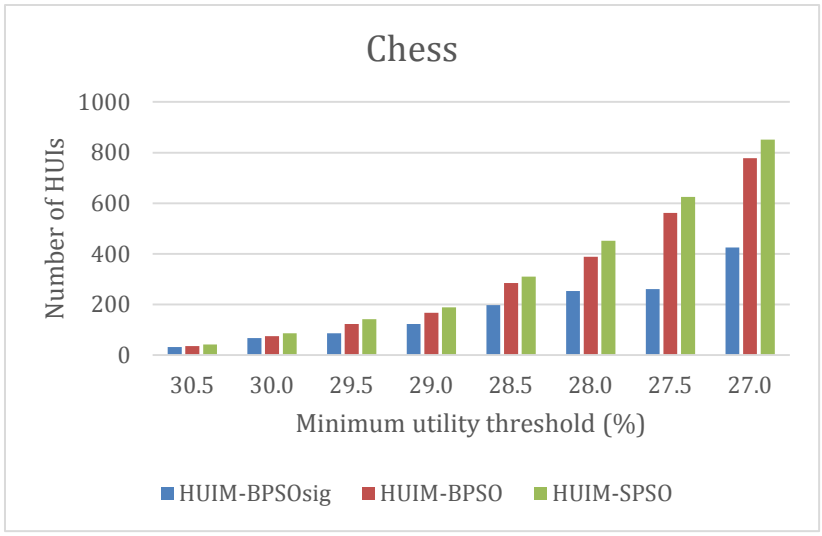
Hình 4. *Biểu diễn thời gian thực thi trên Connect dataset.*

Hình 4 biểu diễn thời gian thực thi của ba thuật toán: HUIM-BPSOsig (hình thoi màu xanh), HUIM-BPSO (hình vuông màu vàng) và HUIM-SPSO (hình tam giác màu xám) trên Connect dataset. Connect dataset là loại dữ liệu dày mật độ 33%, nên ngưỡng minimun utility từ 32.8% đến 31.4%, HUIM-SPSO có thời gian chạy nhanh gấp 3 lần so với HUIM-BPSOsig và HUIM-BPSO.

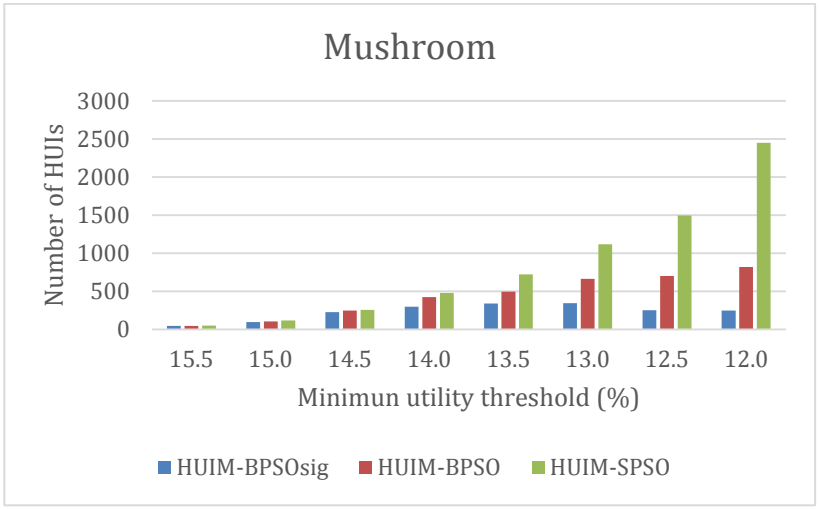
Thực nghiệm cho thấy thuật toán đề xuất của chúng tôi thực thi nhanh hơn hai thuật toán HUIM-BPSOsig [5] và HUIM-BPSO [6]. Với loại dữ liệu có mật độ thấp hơn thì thuật toán HUIM-SPSO vượt trội hơn về thời gian thực thi chương trình. Với việc sử dụng bitmap, thời gian thực hiện tỏ ra khá vượt trội.

C. SỐ LƯỢNG HUI ĐƯỢC PHÁT HIỆN

Hình 5 biểu diễn số lượng HUI được phát hiện của ba thuật toán: HUIM-BPSOsig (màu xanh), HUIM-BPSO (màu vàng) và HUIM-SPSO (màu xám) trên Chess dataset. Chess dataset với mật độ 49%, khi ngưỡng min_util giảm từ 30.5% xuống 27% thì số lượng HUI được phát hiện của HUIM-SPSO vượt trội hơn thuật toán HUIM-BPSOsig.

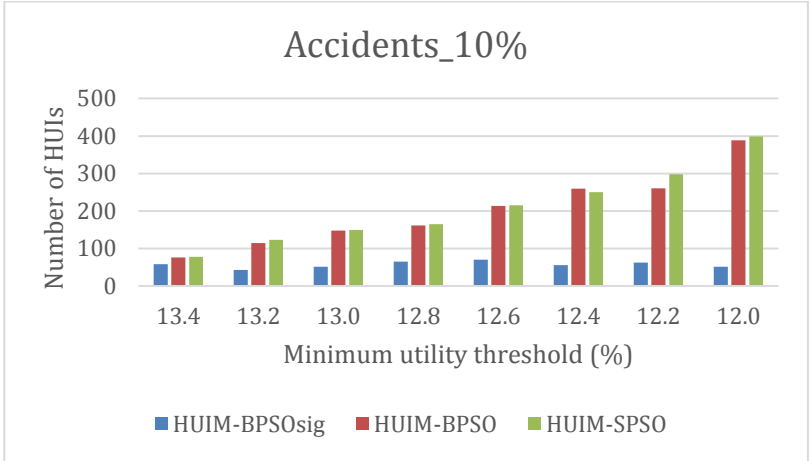


Hình 5. *Biểu diễn số lượng HUIs được khám phá trên Chess dataset.*



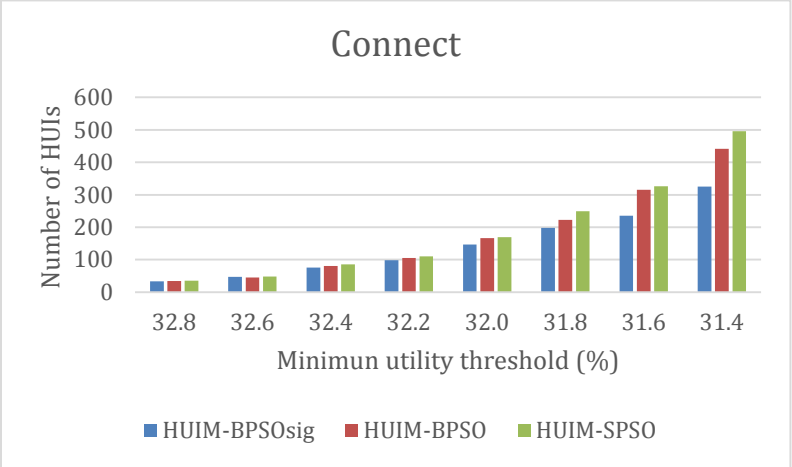
Hình 6. *Biểu diễn số lượng HUIs được khám phá trên Mushroom dataset.*

Hình 6 biểu diễn số lượng HUI được phát hiện của ba thuật toán: HUIM-BPSOsig (màu xanh), HUIM-BPSO (màu vàng) và HUIM-SPSO (màu xám) trên Mushroom dataset. Mushroom dataset với mật độ 19%, khi ngưỡng min_util giảm từ 15.5% xuống 12% thì số lượng HUI được phát hiện của HUIM-SPSO vượt trội hơn hai thuật toán HUIM-BPSOsig, HUIM-BPSO.



Hình 7. *Biểu diễn số lượng HUIs được khám phá trên Accidents_10% dataset.*

Hình 7 biểu diễn số lượng HUI được phát hiện của ba thuật toán: HUIM-BPSOsig (màu xanh), HUIM-BPSO (màu vàng) và HUIM-SPSO (màu xám) trên Accidents_10% dataset. Accidents_10% dataset với mật độ 7%, khi ngưỡng min_util giảm từ 13.4% xuống 12% thì số lượng HUI được phát hiện của HUIM-SPSO vượt trội hơn thuật toán HUIM-BPSOsig.



Hình 8. *Biểu diễn số lượng HUIs được khám phá trên Connect dataset*

Hình 8 biểu diễn số lượng HUI được phát hiện của ba thuật toán: HUIM-BPSOsig (màu xanh), HUIM-BPSO (màu vàng) và HUIM-SPSO (màu xám) trên Connect dataset. Connect dataset với mật độ 33%, khi ngưỡng min_util giảm từ 32.8% xuống 31.4% thì số lượng HUI được phát hiện của HUIM-SPSO vượt trội hơn thuật toán HUIM-BPSOsig.

Thực nghiệm cho thấy thuật toán đề xuất của chúng tôi có số lượng HUI được khám phá nhiều hơn hai thuật toán HUIM-BPSOsig [5] và HUIM-BPSO [6]. Đối với loại dữ liệu như: Chess và Connect dataset (mật độ cao hơn) thì số lượng HUI khai thác được không vượt trội nhiều hơn so với loại dữ liệu là: Mushroom và Accidents_10% dataset (mật độ thấp hơn). Ngưỡng minimum utility càng nhỏ (*min_util* càng nhỏ) thì số lượng HUI được khám phá càng nhiều.

VI. KẾT LUẬN

Trong bài báo này, một thuật toán HUIM mới được gọi là HUIM-SPSO được đề xuất dựa trên SPSO. Trái ngược với PSO điển hình, S-PSO có xu hướng thay đổi thành phần của véc tơ vị trí với vận tốc hơn chứ không phải chỉ đơn giản là dùng đến phép biến đổi hàm sigmoid. Mô hình xử lý của HUIM dùng S-PSO đã được mô tả. Để đo lường sự đa dạng của kết quả được phát hiện, khoảng cách chỉnh sửa bit đã được đề xuất. Với việc sử dụng bitmap và thực nghiệm trên 4 dataset dày nêu ở phần 5, kết quả thử nghiệm đã chứng minh rằng thuật toán được đề xuất có thời gian thực thi nhanh và việc khám phá số lượng HUI hiệu quả hơn hai thuật toán HUIM-BPSOsig [5] và HUIM-BPSO [6].

Hạn chế của đề tài là xử lý dữ liệu lớn là tốc độ còn chậm do phải duyệt CSDL hai lần. Do đó, hướng tới sẽ viết trên môi trường song song như: Hadoop (map, reduce của Python), Spark (ngôn ngữ Scala), multithread, multiprocess. Chúng tôi sẽ cải tiến để có thể duyệt qua CSDL một lần với một cấu trúc dữ liệu khác. Chúng tôi sẽ thực nghiệm trên loại dữ liệu thưa và lớn để có đánh giá đúng đắn hơn về việc sử dụng bitmap như trong đề xuất của chúng tôi.

VII. TÀI LIỆU THAM KHẢO

[1] W.-N. Z. J. C. H. S. H. Z. W.-L. W. W.-G. S. Y. Chen, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE T. Evolut. Comput,* vol. 14(2), pp. 278-300, 2010.

[2] P. L. C. W. G. A. G. T. S. A. D. Z. L. H. T. Fournier-Viger, "The SPMF open-source data mining library version 2," *Berendt, B., Bringmamn, B., Fromont, É., Garriga, G., Miettinen, P., Tatti, N., Tresp, V. (eds) ECML PKDD 2016. LNCS,* vol. 9853, pp. 36-40, 2016.

[3] S. P. K. Kannimuthu, "Discovery of high utility itemsets using genetic algorithm with ranked mutation," *Appl. Artif. Intell,* vol. 28(4), pp. 337-359, 2014.

[4] J. E. R. Kennedy, "A discrete binary version of particle swarm algorithm," *Proceedings of The 1997 IEEE International Conference on Systems, Man, and Cybernetics,* pp. 4104-4108, 1997.

[5] J. C.-W. Y. L. F.-V. P. W. M.-T. H. T.-P. W. S.-L. L. Z. J. Lin, "Mining high-utilitu itemsets based on particle swarm optimization," *Eng. Appl. Artif. Intel,* vol. 55, pp. 320-330, 2016.

[6] J. C.-W. Y. L. F.-V. P. H. T.-P. V. M. Lin, "A binary PSO approach to mine high-utility itemsets," *Soft Comput,* vol. 21(17), pp. 5103-5121, 2017.

[7] J. W. K. F. B. C. M. Liu, "Direct discovery of high utility itemsets without candidate generation," *Proceedings of The 12th IEEE International Conference on Data Mining,* pp. 984-989, 2012.

[8] M. Q. J.-F. Liu, "Mining high utility itemsets without candidate generatin," *Proceedings of The 21st ACM International Conference on Information and Knowledge Management,* pp. 55-64, 2012.

[9] Y. L. W.-K. C. A. N. Liu, "A two-phase algorithm for fast discovery of high utility itemsets," *Ho, T. B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS,* vol. 3515, pp. 689-695, 2005.

[10] W. H. C. Song, "Discovering high utility itemsets based on artificial bee colony algorithm," *Phung, D., Tseng, V., Webb, G., Ho, B., Ganji, M., Rashidi, L. (eds) PAKDD 2018. LNCS,* vol. 10939, pp. 3-14, 2018.

[11] W. L. Y. L. J. Song, "Vertical mining for high utility itemsets," *Proceedings of The 2012 IEEE International Conference on Granular Computing,* pp. 429-434, 2012.

[12] W. Z. Z. L. J. Song, "A high utility itemsets mining algorithm based on subsume index," *Knowl. Inf. Syst.,* vol. 49(1), pp. 315-340, 2016.

[13] J. M. T. Z. J. L. J. C. W. Wu, "An ACO-based approach to mine high-utility itemsets," *Knowl._Based Syst.,* vol. 116, pp. 102-113, 2017.

[14] R. X. M. J. P. S. N. Yang, "Real time utility-based recommendation for revenue optimization via an adaptive online top-k high utility itemsets mining model," *Proceedings of The 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery,* pp. 1859-1866, 2017.

[15] P. Y. U. W. Y. S. W. N. M. Fournier-Viger, "Mining High Utility Itemsets with Hill Climbing and Simulated Annealing," *ACM Transactions on Management Information Systems,* vol. 0, 2021.

[16] L. C. H. Song, "Artificial Fish Swarm Algorithm for Mining High Utility Itemsets," *Springer Natural Swizerland AG 2021,* pp. 407-419, 2021.

HIGH UTILITY ITEMSETS MINING BASED ON PARTICLE SWARM OPTIMIZATION USING BITMAP

Pham Duc Thanh

ABSTRACT— Mining high utility itemsets (HUIs) is a hot research topic nowadays in data mining. Algorithms that evolve in nature are attracting more and more attention because they have the advantage of avoiding combinatorial explosion of the search space. Among the natural evolution algorithms used to mine HUIs, the swarm optimization (PSO) algorithm is the most popular. PSO exploits HUI based on sigmoid non-stop position transformation for velocity. In this paper, we propose a HUIM algorithm based on the PSO set (S-PSO) called HUIM_SPSO, which mainly considers the elements at high-velocity positions. We perform HUIM modeling using S-PSO and explain HUIM-SPSO in detail. To reflect the diversity of mining results, we recommend a bit measure to correct the distance. The experimental results show that the HUIM-SPSO algorithm is efficient and can discover more HUIs with high diversity.

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 bộ môn Hệ thống Thông tin, thuộc 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.