Journal of Computer Science and Cybernetics, V.30, N.3 (2014), 189–202
DOI:10.15625/1813-9663/30/3/2842

KHAI THÁC LUẬT PHÂN LỚP KẾT HỢP TRÊN CƠ SỞ DỮ LIỆU PHÂN TÁN

NGUYỄN THỊ THÚY LOAN[1], ĐỖ TRUNG TUẤN[2], NGUYỄN HỮU NGỰ[2]

[1]*Khoa công nghệ thông tin, Trường Đại học Ngoại ngữ - Tin học, Tp. HCM*
nthithuyloan@gmail.com
[2]*Khoa Toán – Cơ – Tin, Đại học Khoa học Tự nhiên Hà Nội*
tuandt@vnu.edu.vn; nguyenhngu@gmail.com

Tóm tắt. Bài báo đề nghị một phương pháp khai thác luật phân lớp kết hợp trên cơ sở dữ liệu phân tán dựa trên mạng ngang hàng. Phương pháp này tận dụng được năng lực tính toán của các máy trong mạng để xử lý thông tin tại mỗi vị trí và chỉ truyền các thông tin của các itemset có độ hỗ trợ thỏa ngưỡng độ hỗ trợ tối thiểu từ các bên tham gia cho bên cần khai thác. Chính vì vậy, phương pháp đề nghị giảm thiểu được không gian lưu trữ so với việc chuyển toàn bộ cơ sở dữ liệu về bên cần khai thác luật.

Từ khóa. CBA, CAR-Miner, luật phân lớp kết hợp, phân tán, mạng ngang hàng.

Abstract. This paper proposes a method for mining class association rules in distributed datasets by using peer-to-peer network. This method utilizes the computing performance of PCs in the network to process the local information at each site, and then only transfers the information of itemsets whose degree supports satisfy the minimum support threshold to the mining site. Therefore, the proposed method can reduce the memory usage more considerably than that of transferring all dataset's information to the mining site.

Keywords. CBA, CAR-Miner, class association rules, distributed, peer-to-peer network.

1. GIỚI THIỆU

Với sự bùng nổ thông tin như hiện nay, khối lượng dữ liệu phục vụ cho nhu cầu hằng ngày của mỗi tổ chức càng nhiều, càng đa dạng và phong phú. Thế nhưng những thông tin quí giá, những tri thức phục vụ cho nhu cầu quản lý, chiến lược hay định hướng cho tổ chức càng khó tìm thấy, nó bị chôn vùi sâu trong khối lượng dữ liệu khổng lồ của chính tổ chức đó. Khai thác dữ liệu được ra đời và ứng dụng nhằm phục vụ cho các nhu cầu khai thác các tri thức tiềm ẩn đó.

Trong các hướng khai thác dữ liệu thì khai thác luật kết hợp là một phương pháp mô tả dữ liệu, có nhiệm vụ phân tích nhằm phát hiện và đưa ra những qui luật tiềm ẩn, những mối liên hệ tương quan giữa các giá trị dữ liệu trong cơ sở dữ liệu tác nghiệp của tổ chức. Luật kết hợp thu được thường có dạng một mệnh đề có hai vế: $X \to Y$, trong đó X được gọi là vế trái của luật (tiền kiện), Y được gọi là vế phải của luật (hậu kiện). Luật kết hợp tuy khá đơn giản nhưng những thông tin mà luật đem lại mang nhiều ý nghĩa quan trọng, hỗ trợ không

nhỏ trong quá trình ra quyết định, quản lý và có tính định hướng. Khai thác luật kết hợp nhằm tìm ra những mối liên kết đáng quan tâm hoặc những quan hệ tương quan trong một tập lớn các đối tượng. Trong giao dịch thương mại khám phá mối quan hệ trong số lượng lớn các bản ghi giao dịch có thể giúp nhiều nhà kinh doanh xử lí giải quyết các vấn đề như: thiết kế catalog như thế nào? Bố trí các sản phẩm như thế nào? v.v.. Một ứng dụng quan trọng của luật kết hợp là phân tích thị trường. Đó là việc phân tích thói quen mua hàng của khách để tìm sự kết hợp giữa các mặt hàng khác nhau trong một lần mua hàng của họ.

Sự đa dạng và phong phú của dữ liệu hình thành nên nhiều mô hình dữ liệu khác nhau, đó cũng là một thách thức đối với việc khai thác dữ liệu. Một trong những bài toán quan trọng và đang được tập trung nhiều nghiên cứu hiện nay là phân lớp dữ liệu. Việc xây dựng một bộ phân lớp hay mô hình sao cho dự đoán đúng các mẫu chưa biết trước lớp là một nhu cầu cấp thiết trong các hệ thống hỗ trợ ra quyết định. Có thể thấy, luật phân lớp chính là luật kết hợp với vế phải chỉ chứa một giá trị của thuộc tính lớp. Phương pháp này được đề nghị vào năm 1998 bởi Liu và các đồng sự [8] và thường cho kết quả chính xác hơn so với C4.5 [13] và ILA [17, 18] (Theo [8, 19, 20, 21]).

Sự bùng nổ thông tin cùng với sự lớn mạnh của ngành công nghiệp phần cứng máy tính và công nghệ mạng, v.v... dẫn đến nhu cầu phân tán dữ liệu trên nhiều máy tính khác nhau, vừa giảm thiểu được các rủi ro khi vận hành, vừa chuyên biệt hóa được các nhu cầu xử lý, tận dụng được tối đa các nguồn lực máy tính, cũng lại vừa thích nghi được với nhiều mô hình tổ chức khác nhau. Chính vì thế, khai thác dữ liệu ngày nay không những thỏa mãn nhu cầu khai thác tri thức tiềm ẩn xuyên thời gian mà còn thỏa mãn nhu cầu khai thác sự tồn tại của những tri thức tiềm ẩn xuyên không gian, phân tán rải rác trên toàn bộ hệ thống máy tính của cả tổ chức, không phụ thuộc vào hệ thống lưu trữ.

Mục tiêu của bài báo là nghiên cứu các phương pháp khai thác luật kết hợp trên cơ sở dữ liệu (CSDL) phân tán, đặc biệt là phân lớp dựa vào khai thác luật kết hợp trên CSDL phân tán, từ đó đề nghị một thuật toán khai thác trên loại CSDL này.

2. CÁC NGHIÊN CỨU LIÊN QUAN

2.1. Phân lớp dựa vào khai thác luật kết hợp

Luật phân lớp đóng vai trò quan trọng trong các hệ thống ra quyết định, chính vì vậy, có rất nhiều phương pháp được phát triển như C4.5 [13], ILA [17, 18]. Các phương pháp này dựa trên kỹ thuật heuristic/tham lam nên độ chính xác thường chưa cao. Chính vì vậy vào năm 1998, Liu và các đồng sự đề xuất phương pháp phân lớp dựa vào khai thác luật kết hợp, được gọi là phân lớp kết hợp (CBA). Phương pháp này thường có độ chính xác cao hơn C4.5 và ILA. Lý do chính là nhờ nó khai thác tập luật đầy đủ hơn C4.5/ILA, có thể sử dụng đa luật để dự đoán nhãn của mẫu mới. Một số phương pháp nhằm nâng cao hiệu quả khai thác được đề nghị về sau như Phân lớp dựa trên luật kết hợp dự đoán (CPAR) [24], phân lớp dựa trên luật kết hợp đa nhãn (CMAR) [7], phân lớp dựa trên luật kết hợp đa lớp, đa nhãn (MMAC) [15], phân lớp dựa trên luật kết hợp đa lớp (MCAR) [16], khai thác luật phân lớp kết hợp dựa trên lớp tương đương và cây ECR (thuật toán ECR-CARM) [22] và khai thác luật phân lớp kết hợp dựa trên cây MECR (thuật toán CAR-Miner) [10]. Một số nghiên cứu chỉ ra bộ phân lớp được tạo ra từ luật phân lớp kết hợp thường có độ chính xác cao hơn các phương pháp truyền thống như C4.5/ILA cả về lý thuyết [17, 18, 19] lẫn thực nghiệm [8].

Phần này trình bày thuật toán CAR-Miner, một thuật toán được áp dụng để khai thác hiệu quả luật phân lớp kết hợp [10].

Gọi D là tập các dữ liệu huấn luyện với n thuộc tính $A_1, A_2, \ldots, A_n$ và $|D|$ đối tượng (mẫu). Gọi $C = \{c_1, c_2, \ldots, c_k\}$ là tập các nhãn lớp. Mỗi giá trị của thuộc tính A_i và thuộc tính lớp C được ký hiệu bởi các ký tự thường a và c tương ứng.

Định nghĩa 2.1. Một itemset là một tập các cặp (thuộc tính, giá trị), được ký hiệu bởi $\{(A_{i1}, a_{i1}), (A_{i2}, a_{i2}), \ldots, (A_{im}, a_{im})\}$.

Định nghĩa 2.2. Một luật phân lớp kết hợp r có dạng $\{(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})\} \rightarrow c$, trong đó $\{(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})\}$ là một itemset và $c \in C$ là một nhãn lớp.

Định nghĩa 2.3. Khả năng xảy ra của luật r, kí hiệu $ActOcc(r)$, là số dòng trên D chứa vế trái của r.

Định nghĩa 2.4. Đếm độ hỗ trợ của luật r, kí hiệu $Sup(r)$, là số dòng trên D chứa vế trái và vế phải của r.

Định nghĩa 2.5. Độ tin cậy của luật r là tỉ số giữa $Sup(r)$ chia cho $ActOcc(r)$, nghĩa là $\text{Conf}(r) = \frac{Sup(r)}{ActOcc(r)}$

Chẳng hạn, xét luật $r : \{(A, a1)\} \rightarrow y$ từ CSDL ở Bảng 1, ta có $ActOcc(r) = 3$ và $Sup(r) = 2$. Do có 3 đối tượng có giá trị $A = a1$, trong đó hai đối tượng có lớp là $y \Rightarrow$ Độ tin cậy của luật là $conf(r) = 2/3$.

Cấu trúc cây MECR: Mỗi nút trên cây gồm các thông tin sau

a) itemset.

b) Obidset: Tập các ID của đối tượng chứa itemset.

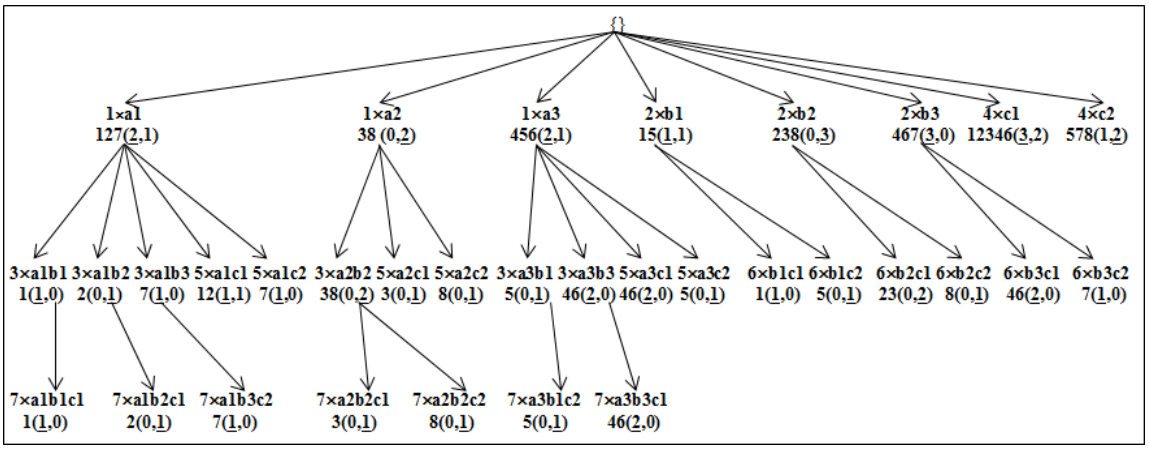
c) ($\#c_1$, $\#c_2$,..., $\#c_k$) – danh sách các số nguyên, trong đó $\#c_i$ là số dòng trong *Obidset* thuộc về lớp c_i.

d) *pos*: một số nguyên lưu trữ vị trí của lớp với số lần xuất hiện nhiều nhất, nghĩa là $pos = \underset{i \in [1,k]}{\arg\max}\{\#c_i\}$.

Chẳng hạn, xét nút chứa itemset $X = \{(A, a_3), (B, b_3)\}$ từ Bảng 1. Do X chứa trong các đối tượng 4, 6 và tất cả đều thuộc về lớp y. Vì vậy, nút $\{(A, a3), (B, b3)\}$ (hay viết đơn giản là $3 \times \underset{46(2,0)}{a3b3}$) được tạo ra trên cây. $pos = 1$ (được gạch chân tại vị trí 1 của nút này) do số đếm thuộc về lớp y là lớn nhất (2 so với 0). Biểu diễn sau khác cách biểu diễn đầu ở chỗ lưu trữ tập thuộc tính. Biểu diễn đầu lưu đầy đủ tập thuộc tính trong khi biểu diễn sau lưu tập thuộc tính dưới dạng bit cho tiết kiệm bộ nhớ (sử dụng cách biểu diễn bít như sau: A có giá trị là $2^0 = 1$; B có giá trị

OID	A	B	C	class
1	a1	b1	c1	y
2	a1	b2	c1	n
3	a2	b2	c1	n
4	a3	b3	c1	y
5	a3	b1	c2	n
6	a3	b3	c1	y
7	a1	b3	c2	y
8	a2	b2	c2	n

Bảng 1: Một CSDL huấn luyện mẫu



Hình 1: Cây MECR-tree được xây dựng từ CSDL của Bảng 1

là $2^1 = 2$; C có giá trị là $2^2 = 4$ nên $AB = 2^0|2^1 = 3$; tương tự $ABC = 2^0|2^1|2^2 = 7$). Với cách biểu diễn sau, itemset được chia thành 2 thành phần att × value.

Cung nối từ nút X đến nút Y sao cho X.itemset $\subset Y$.itemset và $|X.\text{itemset}| = |Y.\text{itemset}| - 1$. Hình 1 mô tả cây tìm kiếm để khai thác luật phân lớp kết hợp. Đầu tiên, mức 1 của cây chứa các nút với 1-itemset, sau đó mỗi nút trên cây sẽ được kết hợp với các nút đứng sau nó có cùng nút cha để tạo ra một tập các nút con của nút hiện hành. Quá trình này được thực hiện một cách đệ qui cho các nút con của nút hiện hành cho đến khi không còn nút nào được tạo ra.

Mệnh đề 2.1. *Cho trước hai nút* $\begin{matrix} att_1 \times value_1 \\ Obidset_1(c_{11}, ..., c_{1k}) \end{matrix}$ *và* $\begin{matrix} att_2 \times value_2 \\ Obidset_2(c_{21}, ..., c_{2k}) \end{matrix}$ *, nếu* $att_1 = att_2$ *và* $value_1 \neq value_2$*, thì* $Obidset_1 \cap Obidset_2 = \emptyset$.

Dựa vào mệnh đề 2.1, thuật toán không cần kết hợp hai itemset X và Y nếu chúng có cùng tập thuộc tính do $Sup(XY) = 0$. Chẳng hạn, xét hai nút $\begin{matrix} 1 \times a1 \\ 127(\underline{2}, 1) \end{matrix}$ và $\begin{matrix} 1 \times a2 \\ 38(\underline{1}, 1) \end{matrix}$, trong đó $Obidset(\{(A, a1)\}) = 127$ và $Obidset(\{(A, a2)\}) = 38 \Rightarrow Obidset(\{(A, a1),(A, a2)\}) = Obidset(\{(A, a1)\}) \cap Obidset(\{(A, a2)\}) = \emptyset$. Tương tự ta có $Obidset(\{(A, a1), (B, b1)\}) = 1$ và $Obidset(\{(A, a1); (B, b2)\}) = 2 \Rightarrow Obidset(\{(A, a1), (B, b1)\}) \cap Obidset(\{(A, a1); (B, b2)\}) = \emptyset$.

Mệnh đề 2.2. *Cho trước 2 nút* $\begin{matrix} itemset_1 \\ Obidset_1(c_{11}, ..., c_{1k}) \end{matrix}$ *và* $\begin{matrix} itemset_2 \\ Obidset_2(c_{21}, ..., c_{2k}) \end{matrix}$ *, nếu* $itemset_1 \subset itemset_2$ *và* $|Obidset_1| = |Obidset_2|$ *thì* $\forall i \in [1, k] : c_{1i} = c_{2i}$.

Từ mệnh đề 2, khi kết hai nút cha thành một nút con, có thể kiểm tra số phần tử của Obidset kết quả, nếu có kết quả bằng với một trong hai nút cha thì chép các thông tin của nút cha tương ứng cho nút con.

Hình 2 bên dưới mô tả chi tiết của thuật toán. Đầu tiên, nút gốc (L_r) chứa các nút 1-itemset phổ biến. Thủ tục CAR-Miner được gọi để khai thác tất cả các luật phân lớp kết hợp với đầu vào là L_r, minSupCount (là số đếm của minSup), minConf.

Thủ tục CAR-Miner (Hình 2) xét mỗi nút l_i với các nút sau nó l_j trên L_r sao cho $j > i$ (dòng 2 và dòng 5) để tạo ra một nút ứng viên l. Thủ tục này cũng sử dụng mệnh đề 1 và 2 để loại bỏ nhanh ứng viên và tính nhanh các thông tin của các nút con (dòng 6 và dòng 13).

Input: A dataset *D*, *minSupCount* and *minConf*
Output: all CARs satisfy *minSupCount* and *minConf*
Procedure:
CAR-Miner(L_r, *minSupCount*, *minConf*)

1. CARs = ∅;
2. for all $l_i \in L_r$.children do
3. **ENUMERATE-CAR**(l_i, *minConf*)
4. P_i = ∅;
5. for all $l_j \in L_r$.children, with j > i do
6. if l_i.att ≠ l_j.att then //Sử dụng mệnh đề 1
7. *O*.att = l_i.att ∪ l_j.att;
8. *O*.value = l_i.value ∪ l_j.value;
9. *O*.*Obidset* = l_i.*Obidset* ∩ l_j.*Obidset*;
10. if |*O*.*Obidset*| = |l_i.*Obidset*| then // Sử dụng mệnh đề 2
11. *O*.count = l_i.count;
12. *O*.pos = l_i.pos;
13. else if |*O*.*Obidset*| = |l_j.*Obidset*| then // Sử dụng mệnh đề 2
14. *O*.count = l_j.count;
15. *O*.pos = l_j.pos;
16. else
17. *O*.count = {count(x ∈ *O*.Obidset | class(x) = c_i, ∀i∈[1,k]};
18. *O*.pos = $\arg\max_{i\in[1,k]}\{l.count_i\}$;
19. if *O*.count[*O*.pos] ≥ *minSupCount* then
20. $P_i = P_i \cup O$;
21. **CAR-Miner**(P_i, *minSupCount*, *minConf*)

ENUMERATE-CAR(*l*, *minConf*)

22. conf = *l*.count[*l*.pos] / | *l*.*Obidset*|;
23. if conf ≥ *minConf* then
24. CARs = CARs ∪ {*l*.itemset → c_{pos} (*l*.count[*l*.pos], conf)}

Hình 2: Thuật toán CAR-Miner

2.2. Các phương pháp khai thác luật kết hợp trên cơ sở dữ liệu phân tán

Khai thác luật kết hợp trên mô hình xử lý song song tận dụng tốc độ tính toán cũng như dung lượng lưu trữ lớn, chuyển đến mô hình phân tán đòi hỏi phân chia cơ sở dữ liệu trên các bộ xử lý khác nhau, các module xử lý khác nhau [23]. Cheung và đồng sự [2] đề nghị thuật toán FDM (Fast Distributed Mining) để song song hóa thuật toán Apriori. Các quá trình thực hiện trên các phần cơ sở dữ liệu riêng lẻ rồi áp dụng các kỹ thuật cắt xén luật phân tán. Một thuật toán khác FPM (Fast Parallel Mining) – dựa trên hệ song song không chia sẻ [3]. Phương pháp này sử dụng cách tiếp cận đếm phân tán và kết hợp hai phương pháp cắt

xén phân tán và toàn cục. Có mô hình giao tiếp đơn giản, chỉ trao đổi những thông tin trong những lần lặp. Parthasarathy và đồng sự [12] đã khái quát các nghiên cứu gần đây về song song hóa khai thác luật kết hợp và kiến trúc chia sẻ bộ nhớ, cũng như xu hướng, khó khăn, sự thay thế trong khai thác song song. Các phương pháp hầu hết đặt nền tản trên Apriori. Tang và Turkia [14] cũng đề xuất mô hình song song dựa trên FP-tree.

Một trong những cách tiếp cận tính toán song song trên CSDL phân tán được trình bày trong [5], các tác giả dựa trên chặn trên và chặn dưới của độ phổ biến để xác định itemset nào có thể xác định được độ hỗ trợ (derivable) và itemset nào chưa xác định được độ hỗ trợ (non-derivable). Chỉ những itemset là non-derivable mới cần gửi đến các CSDL phân tán để xác định độ hỗ trợ. Bên cạnh đó, tính toán song song và phân tán theo phương pháp tăng cường cũng được đề xuất trong [11]. Trong công trình này, các tác giả đề xuất phương pháp khai thác tập phổ biến tối đại cục bộ ở từng vị trí và tổng hợp kết quả. Trong [6], tác giả đề xuất một thuật toán cải tiến cho việc khai thác song song tập phổ biến trên CSDL phân tán ngang. Ngoài ra, việc khai thác song song luật kết hợp trên CSDL phân tán dọc cũng được đề xuất trong [1]. Các tác giả sử dụng IT-tree với đặt điểm chỉ cần quét CSDL một lần nhằm khai thác nhanh tập phổ biến.

Các thuật toán kể trên được dùng trong việc phân tán dữ liệu để tính toán song song, chủ yếu là đề nghị giải pháp khai thác cục bộ trên từng vị trí sau đó chỉ tổng hợp dữ liệu. Việc phân tán dữ liệu trên nhiều vị trí làm cho khối lượng dữ liệu cần tính toán trên mỗi vị trí giảm đáng kể, dẫn đến khả năng tăng tốc của các thuật toán. Một vấn đề lớn cần giải quyết trên các hệ thống khai thác phân tán là thời gian truyền/nhận dữ liệu và kết quả khai thác. Đôi khi, thời gian này lớn hơn nhiều so với thời gian khai thác khi ngưỡng độ hỗ trợ tối thiểu (minSup) lớn hay số lượng tập phổ biến thu được ít. Để giảm chi phí truyền/nhận, các nghiên cứu thường tập trung khai thác tập phổ biến tối đại. Số lượng tập phổ biến tối đại thường ít hơn nhiều so với số lượng tập phổ biến nên cách tiếp cận này tỏ ra hiệu quả hơn việc tập trung dữ liệu để khai thác.

Mục tiêu của bài báo là nhằm khai thác luật phân lớp dựa vào khai thác luật kết hợp trên CSDL đã được phân tán với đặc điểm chứa rất nhiều luật. Chính vì vậy, thời gian truyền/nhận kết quả giữa các bên sẽ rất lớn nếu tiếp cận theo mô hình hiện tại được đề xuất cho khai thác tập phổ biến/tập phổ biến tối đại.

3. Phân lớp dựa vào khai thác luật kết hợp trên cơ sở dữ liệu phân tán

3.1. Nêu bài toán

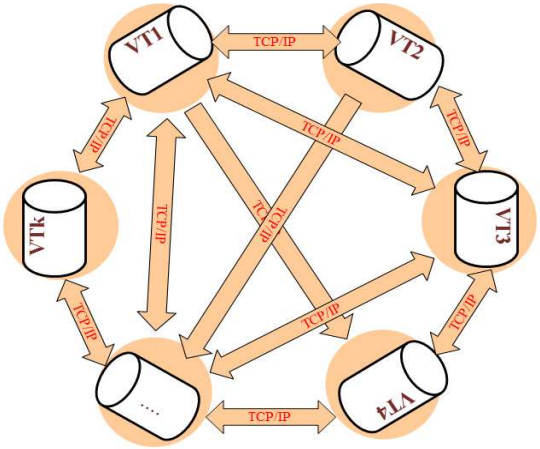
Một công ty có nhiều chi nhánh, mỗi chi nhánh sẽ có một CSDL riêng để quản lý các giao dịch của chi nhánh mình. Do khối lượng giao dịch là rất lớn và việc truy vấn trên toàn bộ CSDL ít xảy ra, vì vậy công ty quyết định lưu trữ dữ liệu độc lập ở mỗi chi nhánh. Vấn đề đặt ra là làm thế nào để có thể khai thác các luật phân lớp dựa vào khai thác luật kết hợp trên toàn bộ CSDL của công ty? Một cách tổng quát, bài toán có thể được phát biểu như sau: Giả sử công ty có một CSDL là DB được lưu trữ trên k chi nhánh với các CSDL con là $\{DB_1, DB_2, \ldots, DB_k\}$ không giao nhau, trong đó các DB_i có cùng cấu trúc (nghĩa là chúng có cùng tập thuộc tính $A_1, A_2, \ldots, A_n$) và DB không tồn tại dưới dạng logic. Bài toán đặt ra là: Cho trước hai ngưỡng minSup và minConf, hãy khai thác tất cả các luật phân lớp kết hợp trên DB.

3.2. Giải quyết vấn đề

Một trong những cách làm đơn giản nhất là tập trung toàn bộ các CSDL lại tại máy cần khai thác để khai thác luật. Cách làm này có ưu điểm là đơn giản, có thể tận dụng các thuật toán khai thác luật hiện nay để giải quyết bài toán. Tuy nhiên, phương pháp này có nhiều nhược điểm: Thứ nhất, trong các CSDL lớn, việc tập trung CSDL lại sẽ tốn nhiều không gian lưu trữ và thời gian truyền/nhận dữ liệu. Thứ hai, nhiều giá trị có tần số xuất hiện không thỏa ngưỡng cũng sẽ được truyền/nhận giữa các vị trí nên rất tốn thời gian. Cuối cùng, phương pháp này không tận dụng được khả năng tính toán của các bên tham gia (các vị trí).

Cách thứ hai là chỉnh sửa các thuật toán khai thác song song và phân tán sử dụng trong khai thác tập phổ biến cho khai thác luật phân lớp kết hợp. Muốn vậy, cần có hai yếu tố chính như sau: 1) Một hệ thống máy tính hiệu năng cao; 2) Cần có phương pháp giảm số lượng luật để giảm thời gian truyền/nhận kết quả. Cách này có ưu điểm là xử lý nhanh do được thực thi trên hệ thống máy chuyên nghiệp, các vị trí có thể xử lý độc lập và tổng hợp kết quả. Nhược điểm của phương pháp này là: 1) Công ty phải tốn chi phí đầu tư máy móc thiết bị và 2) Khó sinh luật do việc sinh luật phải được thực hiện khi đã có đầy đủ các thông tin về các itemset ở tất cả các vị trí, việc rút gọn luật ở từng vị trí cũng khó khăn do muốn rút gọn luật dư thừa phải xem xét trên tập luật toàn cục. Với mô hình tổ chức hiện hành của công ty, cách thứ 2 không tận dụng được năng lực tính toán của các máy đang chứa CSDL mà phải chuyển các CSDL này vào hệ thống máy tính hiệu năng cao để khai thác. Điều này không khả quan nếu việc khai thác được thực hiện thường xuyên (Do các CSDL thay đổi).

Một cách tiếp cận khả quan và đơn giản hơn là sử dụng mô hình mạng ngang hàng, bất kỳ máy nào có nhu cầu khai thác cũng có thể thực hiện được. Hình 3 mô tả mô hình khai thác luật phân lớp kết hợp dựa trên mạng ngang hàng. Mỗi máy (vị trí) có một CSDL cục bộ và giao tiếp với nhau theo giao thức TCP/IP, nghĩa là việc truyền nhận dữ liệu trên mạng cục bộ/internet thông qua chuẩn của giao thức này (chẳng hạn như FTP). Với mô hình như trên, người quản trị có thể thiết lập mối quan hệ giữa các bên tham gia. Chẳng hạn: Vị trí 1 (VT1) có mối quan hệ với tất cả các vị trí còn lại, nghĩa là nó có thể khai thác trên toàn bộ CSDL của công ty. Tuy nhiên, vị trí 2 chỉ có mối quan hệ với các vị trí 1, 3, 5 nên nó chỉ có thể khai thác trên tập dữ liệu của các vị trí 1, 2, 3, và 5.



Hình 3: Mô hình mạng ngang hàng

3.3. Thuật toán đề nghị

Dựa vào mô hình trên, một thuật toán khai thác luật phân lớp kết hợp được đề nghị trong phần này. Không mất tính tổng quát, giả sử vị trí i là máy cần khai thác, vị trí này có quan hệ với các vị trí $\{i_1, i_2, \ldots, i_m\}$ $(i \neq i_j)$. Vị trí i muốn khai thác trên $DB = DB_i \cup DB_{i1} \cup DB_{i2} \cup \ldots \cup DB_{im}$.

Đầu vào của thuật toán là $\{DB_i, DB_{i1}, DB_{i2}, \ldots, DB_{im}\}$, $minSup$ và $minConf$. Đầu ra là tập các luật phân lớp kết hợp chứa trong DB thỏa $minSup$ và $minConf$.

Các bước thực hiện của thuật toán như sau:

Bước 1	Vị trí i gửi minSup cho các vị trí có quan hệ và chờ phản hồi.
Bước 2	Các vị trí có quan hệ với vị trí i đọc CSDL của mình và gửi các giá trị có độ hỗ trợ lớn hơn hay bằng minSup cho vị trí i.
Bước 3	Vị trí i tổng hợp kết quả gửi lại cho các vị trí có quan hệ.
Bước 4	Các vị trí có quan hệ với vị trí i gửi các thông tin về cho vị trí i để tổng hợp kết quả.
Bước 5	Vị trí i tiến hành khai thác (sử dụng thuật toán CAR-Miner được trình bày trong phần 2).

Hình 4: Thuật toán khai thác CARs trên CSDL phân tán

Ví dụ minh họa: Giả sử có hai CSDL được lưu ở hai vị trí là VT1 và VT2, vị trí 1 gồm 6 mẫu, vị trí 2 gồm 3 mẫu như trong bảng 2 và bảng 3 bên dưới.

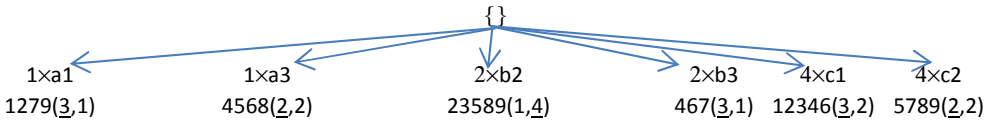
Giả sử $VT2$ muốn khai thác với $minSup = 20\%$ và $minConf = 60\%$, ta có quá trình khai thác như sau. Bước 1: VT2 gửi $minSup$ cho các VT có liên hệ (trường hợp này chỉ có $VT1$). Bước 2: $VT1$ tính $minSupCount_1 = 20\% \times 6 = 1.2$. Như vậy, nó sẽ gửi các giá trị có số lần xuất hiện ≥ 2 cho VT2, trong trường hợp này sẽ là $\{A : a1, a3; B : b2, b3; C : c1\}$. Tương tự, $VT2$ ($minSupCount_2 = 1$) cũng sẽ có kết quả là $\{A : a1, a3; B : b2, b3; C : c2\}$. Bước 3: $VT2$ tổng hợp kết quả $\{A : a1, a3; B : b2, b3; C : c1, c2\}$, sau đó gửi $\{a1, a3; b2, b3; c1, c2\}$ cho các vị trí liên quan ($VT1$). Bước 4: VT1 sẽ gửi về VT2 các thông tin $\{12, 456; 235, 46; 12346, 5\}$ đồng thời gửi $\{y, n, n, y, n, y\}$ (nhãn của các ID trong CSDL của $VT1$). Bên $VT2$ cũng sẽ đọc các thông tin liên quan và kết quả sẽ là $\{79, 8; 89, 7; \emptyset, 789\}$ và $\{y, n, y\}$. $VT2$ tổng hợp kết quả thành các nút của mức 1 trên cây như Hình 5.

OID	A	B	C	class
1	a1	b1	c1	y
2	a1	b2	c1	n
3	a2	b2	c1	n
4	a3	b3	c1	y
5	a3	b2	c2	n
6	a3	b3	c1	y

Bảng 2: CSDL của VT1

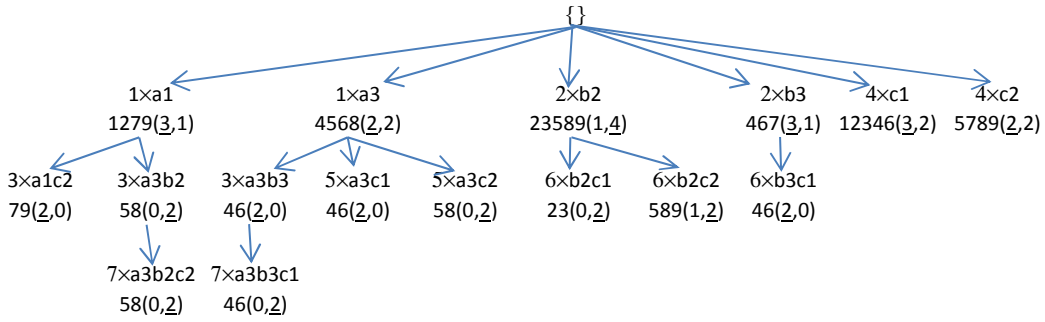
OID	A	B	C	class
7	a1	b3	c2	y
8	a3	b2	c2	n
9	a1	b2	c2	y

Bảng 3: CSDL của VT2



Hình 5: Mức 1 của cây MECR-tree

Bước 5: Sử dụng thuật toán CAR-Miner để khai thác với minSupCount = 2, ta có kết quả được trình bày trong Hình 6.



Hình 6: Cây MECR-tree minh họa quá trình khai thác luật

Hình 6 minh họa quá trình khai thác CARs dựa trên MECR-tree từ các itemset có được của $VT1$ và $VT2$. Đầu tiên, mức 1 của cây chứa các item đơn phổ biến (nghĩa là chứa các item có $count[pos] \geq 2$) gồm $\left\{ \begin{array}{cccccc} 1\times a1 & 1\times a3 & 2\times b2 & 2\times b3 & 4\times c1 & 4\times c2 \\ 1279(\underline{3},1) & 4568(\underline{2},2) & 23589(1,\underline{4}) & 467(\underline{3},0) & 12346(\underline{3},2) & 5789(\underline{2},2) \end{array} \right\}$. Sau đó thủ tục CAR-Miner được gọi với tham số đầu vào là L_r chứa 6 nút trên. Xét nút $\begin{array}{c} 1\times a1 \\ 1279(\underline{3},1) \end{array}$:

- Xét nút $\begin{array}{c} 1\times a3 \\ 4568(\underline{2},2) \end{array}$: Do chúng có tập thuộc tính bằng nhau nên theo mệnh đề 1, hai nút này không cần kết.

- Xét nút $\begin{array}{c} 2\times b2 \\ 23589(1,\underline{4}) \end{array}$: Đầu tiên tính $Obidset(3\times a1b2) = Obidset(1\times a1) \cap Obidset(2\times b2) = 1279 \cap 23589 = 29 \Rightarrow count = (\underline{1},1)$ và $pos = 1$. Ta có $count[pos] < minSupCount$ nên không tạo ra nút mới trên cây.

- Xét nút $\begin{array}{c} 2\times b3 \\ 467(\underline{3},0) \end{array}$: Đầu tiên tính $Obidset(3\times a1b3) = Obidset(1\times a1) \cap Obidset(2\times b3) = 1279 \cap 467 = 7 \Rightarrow count = (\underline{1},0)$ và $pos = 1$. Ta có $count[pos] < minSupCount$ nên không tạo ra nút mới trên cây.

- Xét nút $\begin{array}{c} 4\times c1 \\ 12346(\underline{3},2) \end{array}$: Đầu tiên tính $Obidset(5\times a1c1) = Obidset(1\times a1) \cap Obidset(4\times c1) = 1279 \cap 12346 = 12 \Rightarrow count = (\underline{1},1)$ và $pos = 1$. Ta có $count[pos] < minSupCount$ nên không tạo ra nút mới trên cây.

- Xét nút $\begin{array}{c} 4\times c2 \\ 5789(\underline{2},2) \end{array}$: Đầu tiên tính $Obidset(5\times a1c2) = Obidset(1\times a1) \cap Obidset(4\times c2) = 1279 \cap 5789 = 79 \Rightarrow count = (\underline{2},0)$ và $pos = 1$. Do $count[pos] \geq minSupCount$ nên nút này được thêm vào cây là một nút con của nút $\begin{array}{c} 1\times a1 \\ 1279(\underline{3},1) \end{array}$.

Tương tự cho các nút còn lại của cây.

3.4. Phân tích độ phức tạp thuật toán

Phương pháp trong bài báo này sử dụng CAR-Miner để khai thác luật phân lớp kết hợp nên độ phức tạp của thuật toán phụ thuộc nhiều vào CAR-Miner. Theo [4], độ phức tạp của CAR-Miner được tính theo công thức sau:

$$T_S = K_S \times m + a$$

Trong đó T_S là tổng thời gian khai thác của CAR-Miner, K_S là số lần lặp của thuật toán, m là thời gian để sinh một nút trên cây và a là thời gian để truy cập dữ liệu.

Cách tiếp cận trong bài báo này chủ yếu quan tân đến thời gian truy cập dữ liệu a. Trong các hệ thống phân tán, thời gian này có thể rất lớn khiến cho toàn bộ quá trình khai thác lớn.

Giả sử thời gian truy cập dữ liệu để chuyển hết dữ liệu trên máy thứ i sang máy cần khai thác là a_i, ta có $a = a_1 + a_2 + \ldots + a_k$. Nếu sử dụng ngưỡng $minSup$ để lọc bỏ bớt dữ liệu thì thời gian truy cập dữ liệu sẽ chỉ là thời gian chuyển các item có độ phổ biến thỏa ngưỡng. Gọi a'_i là thời gian truy cập dữ liệu khi có ngưỡng thì tổng thời gian truy cập trên các máy là $a' = a'_1 + a'_2 + \ldots + a'_k$. Do khi sử dụng ngưỡng, khối lượng dữ liệu cần truyền trên mỗi máy thường sẽ ít hơn nên $a'_i \leq a_i \Rightarrow a' \leq a$.

Nhận xét: Kết quả trên cho thấy thời gian để khai thác luật phân lớp sẽ giảm do thời gian truy cập dữ liệu giảm, đặc biệt với các trường hợp ngưỡng $minSup$ lớn. Tính hiệu quả này sẽ giảm dần khi $minSup$ càng giảm và đến một lúc nào đó, tất cả các item đều thỏa $minSup$ thì cách tiếp cận này không còn hiệu quả nữa.

4. THỰC NGHIỆM

Để thấy rõ tính hiệu quả của phương pháp đề nghị so với cách thứ 1 (gửi toàn bộ dữ liệu về cho máy cần khai thác), phần này trình bày khối lượng bộ nhớ yêu cầu để truyền dữ liệu giữa các bên so với việc chuyển tất cả dữ liệu cho bên khai thác.

4.1. Dữ liệu thực nghiệm

Các kết quả thực nghiệm được thực thi trên các CSDL được lấy từ UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/). Bảng 4 mô tả đặc điểm của các CSDL thực nghiệm.

Các CSDL có đặc điểm khác nhau: Breast, German chứa nhiều thuộc tính và giá trị phân biệt nhưng ít mẫu. Led7 chứa ít thuộc tính, giá trị phân biệt lẫn số mẫu. Đặc biệt, Lymph có số mẫu khá ít. Poker-hand chứa nhiều mẫu. Để có thể áp dụng được cho mô hình đề xuất, các CSDL được phân thành 5 phân mảnh (5 CSDL con), mỗi phân mảnh chứa 10%, 20%, 30% hoặc 40% tổng số dòng dữ liệu (tổng 5 phân mãnh chứa 100%).

Dataset	#thuộc tính	#lớp	#giá trị phân biệt	#mẫu
Breast[1]	10	2	737	699
German	20	2	1077	1000
Lymph	18	4	63	148
Led7	7	10	24	3200
Poker-hand	11	10	95	1000000

[1]Breast Cancer Wisconsin (Original)

Bảng 4: Đặc điểm của các CSDL thực nghiệm

4.2. So sánh về khối lượng bộ nhớ

Bảng 5 bên dưới trình bày kết quả so sánh về khối lượng bộ nhớ yêu cầu khi gửi/nhận dữ liệu của mô hình đề xuất (Các bước 2-4) và mô hình 1 (Các bên tham gia gửi hết dữ liệu cho bên khai thác). Kết quả từ Bảng 5 cho thấy khối lương bộ nhớ cần cho việc truyền/nhận thông tin giữa các bên tham gia thường ít hơn so với chuyển toàn bộ các CSDL cho bên cần khai thác. Điều này chứng tỏ tính hiệu quả của phương pháp được đề nghị. Ngoài ra, khi truyền dữ liệu theo mô hình đề xuất, bên khai thác tổng hợp thông tin cần thiết cho CAR-Miner nhanh hơn do mô hình đề xuất đã truyền Obidset của các item đơn cho bên khai thác. Chính vì vậy, thời gian khai thác sẽ giảm (Do mô hình 1 muốn sử dụng CAR-Miner phải chuyển dữ liệu thành các nút chứa các item đơn trên cây). Trong trường hợp chỉ tính khối lượng bộ nhớ cuối cùng (Bước 4) được gửi từ các bên thì khối lượng bộ nhớ càng ít hơn.

CSDL	*minSup%*	Khối lượng bộ nhớ (KB)		
		Mô hình đề nghị	Chỉ tính bên khai thác	Mô hình 1
Breast	10	14.83	13.92	30.04
	7	18.93	17.74	30.04
	4	22.75	20.75	30.04
	1	25.57	21.84	30.04
german	10	67.08	64.06	82.03
	7	67.39	64.06	82.03
	4	73.21	68.36	82.03
	1	76.23	68.59	82.03
Lymph	10	12.15	9.94	10.99
	7	12.18	9.94	10.99
	4	12.27	9.94	10.99
	1	12.39	9.94	10.99
Led7	10	88.09	87.5	100
	7	88.09	87.5	100
	4	88.09	87.5	100
	1	88.09	87.5	100
Poker-hand	10	27345	27343.75	42968.75
	7	39066.31	39062.5	42968.75
	4	39066.31	39062.5	42968.75
	1	39066.31	39062.5	42968.75

Bảng 5: So sánh khối lượng bộ nhớ giữa 2 phương pháp trên nhiều minSup khác nhau

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Công trình này nghiên cứu bài toán khai thác luật phân lớp kết hợp trên CSDL đã được phân tán. Có thể thấy đây là một bài toán thực tế cần giải quyết. Một trong những phương pháp có thể thực hiện được là cải tiến các thuật toán khai thác song song tập phổ biến cho bài toán khai thác luật phân lớp kết hợp. Cách làm này có ưu điểm là tận dụng được thế mạnh xử lý và nguồn tài nguyên của nhiều máy. Tuy nhiên, khi số lượng luật thu được lớn thì

phương pháp này tốn nhiều thời gian truyền/nhận kết quả và tổng hợp kết quả. Bên cạnh đó, việc tính toán song song cũng cần một hệ thống máy chuyên nghiệp trong khi nhu cầu là khai thác tức thời tận dụng năng lực xử lý của các máy hiện hành đang dùng. Với các máy PC kết nối qua LAN/WAN, việc truyền/nhận dữ liệu tương đối chậm do thường truyền theo cơ chế file. Với các phân tích như trên, việc đề nghị một mô hình xử lý có thể tận dụng được năng lực của các bên tham gia, giảm chi phí truyền thông là cần thiết. Mô hình đề xuất không gửi nhận tất cả các thông tin của CSDL cho bên cần khai thác mà chỉ nhận thông tin của các item có khả năng phổ biến (nghĩa là phổ biến ở ít nhất một vị trí). Chính đều này làm giảm thiểu số item cần gửi nhận giữa các bên tham gia. Việc xử lý cũng không phải tập trung hoàn toàn vào bên cần khai thác mà xử lý phân tán trên các bên tham gia, bên khai thác chỉ tổng hợp kết quả và tiến hành khai thác. Nhược điểm chính của phương pháp này là không tận dụng được năng lực tính toán của các bên tham gia trong giai đoạn khai thác và tốn thời gian chờ giữa các bên cho việc gửi/nhận thông tin của các bên tại các bước 1-3. Chính vì vậy, khi minSup rất nhỏ dẫn đến số lượng item thỏa minSup lớn thì giải pháp đề xuất có thể sẽ không hiệu quả.

Một trong những khó khăn của việc xử lý song song trong giai đoạn khai thác luật là việc tổng hợp kết quả, chính vì vậy trong thời gian tới, chúng tôi sẽ cố gắng nghiên cứu đề xuất giải pháp cho vấn đề này. Bên cạnh đó, việc tìm kiếm giải pháp để giảm chi phí truyền thông như sử dụng cấu trúc dữ liệu bit khi truyền dữ liệu cũng sẽ được quan tâm.

Lời cảm ơn: Nghiên cứu này được tài trợ bởi Quỹ phát triển khoa học và công nghệ quốc gia (NAFOSTED) trong đề tài mã số 102.01-2012.17

TÀI LIỆU

[1] Cao Tùng Anh, Khai thác luật kết hợp trên cơ sở dữ liệu phân tán dọc. Hội thảo "một số vấn đề chọn lọc của Công nghệ Thông tin và Truyền thông", Đại Lãi 9/2007, NXB Khoa học Tự nhiên và Công nghệ, pp. 169-180, 2008.

[2] D. Cheung, J. Han, V.T. Ng, A.V. Fu, Y. Fu, "A fast distributed algorithm for mining association rules", *Proc. of 1996 Int'l. Conf. on Parallel and Distributed Information Systems*, Miami Beach, Florida, pp. 31-44, 1996.

[3] D. Cheung, Y. Xiao, "Effect of data skewness in parallel mining of association rules. PAKDD '98", *LNCS* vol. 1394, pp. 48-60, 1998.

[4] D. Nguyen, B. Vo, B. Le, "Efficient strategies for parallel mining class association rules", *Expert Systems with Applications: An International Journal*, vol. 41, no. 10, pp. 4716-4729, 2014.

[5] M. Deypir, M. H. Sadreddini,"Distributed association rules mining using non derivable frequent patterns", *Iranian Journal of Science & Technology, Transaction B: Engineering* vol. 33, no. B6, pp. 511-526, 2009.

[6] Phạm Thị Hân, "Khai phá luật kết hợp trong cơ sở dữ liệu phân tán", Luận văn thạc sĩ chuyên ngành truyền số liệu và mạng máy tính, Học viện Bưu chính Viễn Thông, 2012.

[7] W. Li, J. Han, J. Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules", *The 1st IEEE International Conference on Data Mining*, San Jose, California, USA, pp. 369-376, 2001.

[8] B. Liu, W. Hsu, Y. Ma, "Integrating classification and association rule mining", *The 4th International Conference on Knowledge Discovery and Data Mining*, New York, USA, pp. 80-86, 1998.

[9] B. Liu, Y. Ma, C.K. Wong, "Improving an association rule based classifier", *The 4th European Conference on Principles of Data Mining and Knowledge Discovery*, Lyon, France, pp. 80-86, 2000.

[10] T. T. L. Nguyen, B. Vo, T. P. Hong, H. C. Thanh, "CAR-Miner: An efficient algorithm for mining class-association rules", *Expert Systems with Applications* vol. 40, no. 6, pp. 2305-2311, 2013.

[11] M. E. Otey, S. Parthasarathy, C. Wang, A. Veloso, W. M. Jr, "Parallel and distributed methods for incremental frequent itemset mining", *IEEE Transactions on Systems, Man, and Cybernetics, Part B* vpl. 34, no. 6, pp. 2439-2450, 2004.

[12] S. Parthasarathy, M. J. Zaki, M. Ogihara, "Parallel data mining for association rules on shared-memory systems", *Knowledge and Information Systems: An International Journal* vol. 3,, no. 1, pp. 1–29, 2001.

[13] J. R. Quinlan, *C4.5: program for machine learning*, Morgan Kaufmann Publishers, Inc., 1992.

[14] P. Tang, M. Turkia, "Parallelizing frequent itemset mining with FP-trees", Technical Report (www.ualr.edu/pxtang/papers/CATA06.pdf), Department of Computer Science, University of Arkansas at Little Rock, 2005.

[15] F. Thabtah, P. Cowling, Y. Peng, "MMAC: A new multi-class, multi-label associative classification approach", *The 4th IEEE International Conference on Data Mining*, Brighton, UK, pp. 217-224, 2004.

[16] F. Thabtah, P. Cowling, Y. Peng, "MCAR: Multi-class classification based on association rule", *The 3rd ACS/IEEE International Conference on Computer Systems and Applications*, Tunis, Tunisia, pp. 33-39, 2005.

[17] M. R. Tolun, S. M. Abu-Soud, "ILA: An inductive learning algorithm for production rule discovery", *Expert Systems with Applications*, vol. 14, no. 3, pp. 361–370, 1998.

[18] M.R. Tolun, H. Sever, M. Uludag, S.M. Abu-Soud, "ILA-2: An inductive learning algorithm for knowledge discovery", *Cybernetics and Systems*, vol. 30, no. 7, pp. 609 - 628, 1999.

[19] A. Veloso, W. M. Jr, M. J. Zaki,"Lazy associative classification", *The 2006 IEEE International Conference on Data Mining (ICDM'06)*, Hong Kong, China, pp. 645-654, 2006.

[20] A. Veloso, W. M. Jr, M. Goncalves, M. J. Zaki,"Multi-label lazy associative classification", *The 11th European Conference on Principles of Data Mining and Knowledge Discovery*, Warsaw, Poland, pp. 605-612, 2007.

[21] A. Veloso, W. M. Jr, M. Goncalves¸ H. M. Almeida, M. J. Zaki, "Calibrated lazy associative classification", *Information Sciences*, vol. 181, no. 13, pp. 2656-2670, 2011.

[22] B. Vo, B. Le, "A novel classification algorithm based on association rule mining", *The 2008 Pacific Rim Knowledge Acquisition Workshop (Held with PRICAI'08), LNAI 5465*, Hanoi, Vietnam, pp. 61-75, 2008.

[23] M. J. Zaki, "Parallel and distributed association mining: A survey", *IEEE Concurrency* vo. 7, no. 4, pp. 14-25, 1999.

[24] X. Yin, J. Han, "CPAR: Classification based on predictive association rules", *SIAM International Conference on Data Mining (SDM'03)*, San Francisco, CA, USA, pp. 331-335, 2003.