BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

TRỊNH MINH SỸ

KHAI THÁC MẪU TUẦN TỰ NÉN

LUẬN VĂN THẠC SĨ

Chuyên ngành : Công Nghệ Thông Tin

Mã số ngành: 60480201

TP. HỒ CHÍ MINH, tháng 10 năm 2015

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

----------oOo----------

TRỊNH MINH SỸ

KHAI THÁC MẪU TUẦN TỰ NÉN

LUẬN VĂN THẠC SĨ

Chuyên ngành : Công Nghệ Thông Tin

Mã số ngành: 60480201

CÁN BỘ HƯỚNG DẪN KHOA HỌC: PGS.TS. LÊ HOÀI BẮC

TP. HỒ CHÍ MINH, tháng 10 năm 2015

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

Cán bộ hướng dẫn khoa học : PGS.TS. LÊ HOÀI BẮC

Luận văn Thạc sĩ được bảo vệ tại Trường Đại học Công nghệ TP. HCM ngày 17 tháng 10 năm 2015

Thành phần Hội đồng đánh giá Luận văn Thạc sĩ gồm:

TT	Họ và tên	Chức danh Hội đồng
1	PGS.TS. Nguyễn Xuân Huy	Chủ tịch
2	PGS.TS. Quản Thành Thơ	Phản biện 1
3	TS. Nguyễn Thị Thúy Loan	Phản biện 2
4	TS. Võ Đình Bảy	Ủy viên
5	TS. Cao Tùng Anh	Ủy viên, Thư ký

Xác nhận của Chủ tịch Hội đồng đánh giá Luận Văn sau khi Luận Văn đã được sửa chữa (nếu có).

Chủ tịch Hội đồng đánh giá LV

TRƯỜNG ĐH CÔNG NGHỆ TP. HCM **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

PHÒNG QLKH – ĐTSĐH **Độc lập – Tự do – Hạnh phúc**

TP. HCM, ngày 03 *tháng 04 năm 2015*

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên:TRỊNH MINH SỸ. Giới tính: Nam

Ngày, tháng, năm sinh: 07/10/1960 Nơi sinh: Bình Định

Chuyên ngành:Công Nghệ Thông Tin MSHV: 1341860052

I- Tên đề tài:

KHAI THÁC MẪU TUẦN TỰ NÉN

II- Nhiệm vụ và nội dung:

Mã hóa dữ liệu tuần tự bằng cách gán các codeword đối với các khoảng cách nhỏ, rồi từ đó tiến hành xử lý trên mẫu với khoảng cách lớn hơn.

Tính toán độ phức tạp của quá trình khai phá mẫu nén trên cơ sở dữ liệu tuần tự.

Nghiên cứu thuật toán GoKrimp để khai phá trực tiếp trên mẫu đã được nén dựa trên thuật toán tham lam.

Tiến hành thực nghiệm trên các bộ dữ liệu khác nhau và đánh giá kết quả, đề xuất cải tiến.

III- Ngày giao nhiệm vụ: 03/04/2015

IV - Ngày hoàn thành nhiệm vụ: 07/09/2015

V - Cán bộ hướng dẫn: Phó Giáo Sư. Tiến Sĩ. Lê Hoài Bắc

CÁN BỘ HƯỚNG DẪN **KHOA QUẢN LÝ CHUYÊN NGÀNH**

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Luận văn này cũng như các trích dẫn hay tài liệu học thuật tham khảo đã được cảm ơn đến tác giả và các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Học viên thực hiện Luận văn

Trịnh minh Sỹ

LỜI CÁM ƠN

Trước hết, cho tôi được gửi lời cám ơn đến sự hướng dẫn và giúp đỡ tận tình của Thầy PGS.TS. Lê Hoài Bắc.

TÓM TẮT

Khai thác mẫu tuần tự đóng đối với dữ liệu dạng văn bản đã được áp dụng thành công trong nhiều bài toán khác nhau của khai thác dữ liệu. Tuy nhiên, kết quả khai thác vẫn còn một số mặt hạn chế như:

- Vấn đề dư thừa của mẫu đã trích ra.
- Mẫu trích ra bị trùng lắp.
- Một số mẫu tối nghĩa.

Để khắc phục những mặt hạn chế trên, ý tưởng dựa trên nguyên lý mô tả chiều dài tối thiểu (MDL minimum description length) được đề xuất nhằm khai thác mẫu tuần tự nén đối với cơ sở dữ liệu tuần tự.

Với dữ liệu là itemset, thuật toán Krimp [7] dựa trên mô tả chiều dài tối thiểu đã tỏ ra khá hiệu quả trong việc giải quyết vấn đề dư thừa của mẫu, trích ra những mẫu ít trùng lắp và dễ hiểu hơn.

Với dữ liệu tuần tự, đề tài đề xuất hai thuật toán SeqKrimp và GoKrimp. Trong đó, thuật toán chính của đề tài là GoKrimp.

- SeqKrimp là thuật toán khai thác mẫu nén gồm hai pha:

-Pha thứ nhất: lấy những mẫu tuần tự đóng bởi hàm GetCandidate() đã có.

-Pha thứ nhì: từ những mẫu tuần tự đóng đã lấy ở trên chọn lại chỉ còn những mẫu nén dữ liệu tốt nhất.

Vậy mẫu nén dữ liệu như thế nào là tốt nhất? Đề tài nghiên cứu hai phần:

-Cách thức nén dữ liệu: chọn nén dữ liệu theo phương pháp mã hóa Huffman[3], xây dựng cây nhị phân để mã hóa, cây nhị phân như vậy sẽ thỏa tính chất mã tiền tố nghĩa là không có từ mã nào là phần đầu của từ mã khác.

-Hiệu quả nén: là hiệu số giữa số bit trước khi nén và sau khi nén dữ liệu.

- GoKrimp là thuật toán khai thác trực tiếp mẫu nén, khác với SeqKrimp ở pha thứ nhất, nghĩa là, thuật toán không lấy mẫu tuần tự đóng đã có sẵn mà khai thác trực tiếp mẫu từ tập các từ phổ biến ban đầu. Từ đó dùng thuật toán tham lam nới rộng mẫu nhưng tránh không duyệt hết mọi trường hợp để nới rộng nhờ một kỹ thuật kiểm tra sự kiện liên quan đến mẫu.

Sau cùng nghiên cứu thực nghiệm trên tám tập dữ liệu để chỉ ra rằng GoKrimp tỏ ra hiệu quả nhất. So sánh GoKrimp với các thuật toán như SeqKrimp, BIDE, SQS để thấy ưu điểm ở các tính chất: dễ hiểu, thời gian thực thi, tỉ lệ nén và độ chính xác sự phân lớp.

ABSTRACT

Mining closed sequential pattern in text data has been successfully applied in many different problems of data mining. However, the mining result still has some drawbacks, include:

- The problem of redundancy extracted patterns.

- The duplication of extracting patterns.

- The existing of ambiguous patterns.

To overcome the above drawbacks, the approach that is based on the principles of the minimum description length (MDL) to exploit sequential pattern compression in sequence databases.

For the itemset, Krimp [7] was based on the minimum description length that has proved quite effective in solving the problem of redundant patterns. Thus, it extracts less duplicate patterns and easier to understand.

For the sequence data, the thesis proposed two algorithms, include SeqKrimp and GoKrimp. In which, GoKrimp is the main contribution.

• SeqKrimp mine the compressed pattern that consists of two phases:

The first phase: get the closed sequential pattern by using the exist GetCandidate() function.

The second phase: reselect the best compressed patterns from the closed sequential patterns.

Our work concentrates on how is the best compressed pattern which includes two parts:

- The data compression method: apply the data compression Huffman encoding method[3], built a binary tree for encoding which satisfies the prefixed code

property. That means there is no word code is the beginning part of others word codes.

-Effective compression: The difference between the number bits of data before and after the compression.

• GoKrimp allows to mine the compressed patterns directly. It differs from the first phase of SeqKrimp, i.e., The algorithm mines the patterns directly from the set of initial frequent patterns. The greedy algorithm then is used to extend the patterns. The algorithm avoids to scan all of cases by applying a related events checking technique.

Finally, the experimental studies are conducted on eight datasets to show the efficiency of GoKrimp. The comparisons with SeqKrimp, Bide, SQS to demonstrate the advantages of GoKrimp in the following characteristics: easy to understand, real-time enforcement, compression ratio, and accuracy classification.

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH

CHƯƠNG 1
GIỚI THIỆU ĐỀ TÀI

1.1. ĐẶT VẤN ĐỀ

Vấn đề khai thác mẫu tuần tự phổ biến từ cơ sở dữ liệu tuần tự đã được ứng dụng thành công trong lĩnh vực khai thác dữ liệu. Trong những năm gần đây nhiều công trình khai thác mẫu tuần tự đóng đã đóng góp nhiều ứng dụng cụ thể cho lĩnh vực khai thác dữ liệu. Tuy vậy khi quan sát Bảng 1.1[4] chỉ ra 20 mẫu tuần tự phổ biến đóng được trích ra từ tập dữ liệu Journal of Machine Learning Research (JMLR), bộ dữ liệu này chứa cơ sở dữ liệu của 787 cụm từ, mỗi cụm từ tương ứng với tóm tắt của một bài báo trong JMLR có nhận xét rằng:

- Mẫu trùng lắp mặc dù có độ hỗ trợ cao ví như mẫu algorithm algorithm
- Mẫu tối nghĩa ví như hai mẫu learn algorithm và algorithm learn nếu mẫu này có nghĩa thì mẫu kia tối nghĩa
- Mẫu dư thừa, rườm rà ví như mẫu algorithm algorithm algorithm

Qua quan sát trên rút ra nhận xét rằng có những mẫu có độ hỗ trợ cao nhưng lại xãy ra tình trạng mẫu trùng lắp, tối nghĩa, dư thừa không đáng quan tâm.

Để khắc phục tình trạng trên, đưa ra một ý tưởng mới là khai thác những mẫu dùng để nén bộ dữ liệu và ý tưởng này cơ sở dựa trên thuật toán Krimp[7], thuật toán khai thác mẫu nén đối với bộ dữ liệu là các tập mục (itemset).

Cơ sở của thuật toán Krimp là tính hiệu quả nén đó là số bit lợi được trước và sau khi nén, nguyên lý này gọi là nguyên lý mô tả chiều dài tối thiểu MDL (minimum description length).

Bảng 1.1 Mẫu tuần tự đóng [4]

Pattern	Support	Pattern	Support
algorithm algorithm	0.376	method method	0.250
learn learn	0.362	algorithm result	0.247
learn algorithm	0.356	data set	0.244
algorithm learn	0.288	learn learn learn	0.241
data data	0.244	learn prolem	0.239
learn data	0.263	learn method	0.229
model model	0.260	algorithm data	0.229
problem problem	0.258	learn set	0.228
learn result	0.255	problem learn	0.227
problem algorithm	0.251	algorithm algorithm algorithm	0.222

20 mẫu tuần tự phổ biến đóng không đơn từ bộ dữ liệu tóm tắt JMLR.

Đối với dữ liệu tuần tự, khác với dữ liệu là các tập mục và đặc biệt dữ liệu tuần tự dạng văn bản thì thứ tự các từ khác nhau mang nghĩa khác nhau cho nên việc khai thác mẫu tuần tự phải càng tránh trùng lắp, tối nghĩa hay ngược nghĩa, dư thừa.

Tiếp theo ý tưởng khai thác mẫu nén, đối với mẫu tuần tự dưa ra hai thuật toán:

Thuật toán SeqKrimp: gồm hai giai đoạn

- Giai đoạn một lấy những mẫu tuần tự đóng đã có.
- Chọn những mẫu nén tốt nhất trích ra.

Thuật toán GoKrimp: cũng gồm hai giai đoạn

- Giai đoạn một tự nới rộng mẫu để tìm mẫu.
- Giai đoạn hai chọn những mẫu nén tốt nhất trích ra.

Với hai thuật toán trên thì GoKrimp tỏ ra hiệu quả hơn và là thuật toán chính của đề tài.

1.2. MỤC TIÊU, NỘI DUNG VÀ PHƯƠNG PHÁP NGHIÊN CỨU

1.2.1. Mục tiêu của đề tài

Mục tiêu đề tài là đi tiếp theo của ý tưởng MDL (mô tả chiều dài tối thiểu) để đưa ra đặc tả ngữ cảnh của dữ liệu tuần tự. Và đặc biệt trọng tâm là việc lấy ý tưởng MDL để khám phá mẫu tuần tự nén đáng quan tâm, có nghĩa và khả dụng.

1.2.2. Nội dung nghiên cứu

Trong phạm vi này nghiên cứu giải thuật dựa trên nguyên lý MDL để giải quyết vấn đề dư thừa và tìm ra các mẫu có nghĩa, đáng quan tâm và khả dụng. Có thể tóm tắt công việc trong bốn chương như sau:

Chương 1: Giới thiệu đề tài

Giới thiệu đề tài, nghiên cứu các vấn đề liên quan trong vài năm gần đây, những kết quả đạt được và những mặt hạn chế như là việc trích ra những mẫu dư thừa và tối nghĩa. Đồng thời đưa ra cách hạn chế những khắc phục trên, nghiên cứu mô hình nén dữ liệu đối với cơ sở dữ liệu tuần tự để làm cho mẫu được trích ra sáng tỏ hơn, đáng quan tâm hơn.

Chương 2: Cơ sở lý thuyết

Nghiên cứu mô hình mã hóa dữ liệu tuần tự, mã Huffman, mã Elias từ đó tính hiệu quả nén dữ liệu với phương thức chọn mẫu nén, cách giải mã.

Tính toán độ phức tạp của quá trình khai phá mẫu nén trên cơ sở dữ liệu tuần tự. Kết quả chỉ ra rằng đây là bài toán có độ phức tạp cấp NP-khó và thuộc lớp bài toán không thể xấp xỉ, không thể duyệt hết mọi trường hợp.

Chương 3: Thuật toán khai thác mẫu nén

Nghiên cứu thuật toán SeqKrimp, là thuật toán lấy ứng viên của mẫu tuần tự đóng rồi chọn các mẫu nén có hiệu quả nén cao, thuật toán gồm hai pha, pha thứ nhất lấy những mẫu tuần tự đóng bằng các thuật toán trước đó sau đó từ những mẫu đóng chọn những mẫu nén tốt trích ra,thuật toán này lấy ý tưởng từ thuật toán Krimp.

Nghiên cứu thuật toán GoKrimp để khai phá trực tiếp trên mẫu nén dựa trên thuật toán tham lam, nới rộng mẫu bởi các sự kiện có liên quan bằng phương thức kiểm tra phụ thuộc, thử dấu rồi chọn mẫu có hiệu quả nén dương cao nhất đưa vào từ điển, trích ra mẫu nén.

Trong mỗi thuật toán, giải thích ý nghĩa của thuật toán và có ví dụ minh họa từng bước giải quyết của thuật toán đi kèm.

Chương 4: Thực nghiệm và kết luận.

Tiến hành thực nghiệm trên các 8 bộ dữ liệu khác nhau và đánh giá kết quả, trong đó xem xét đến bốn tính chất là tính dễ hiểu, tính phân lớp, tính nén và thời gian thực thi.

Tóm tắt công việc đã nghiên cứu, những việc đã đạt được, đề xuất cho những việc sắp tới.

1.2.3. Phương pháp luận và phương pháp nghiên cứu

1.2.3.1. Phương pháp luận

Nghiên cứu về lý thuyết các thuật toán có liên quan đến đề tài cùng với các khái niệm đi kèm, trong mỗi thuật toán giải thích ý nghĩa, cho ví dụ minh họa từng bước giải quyết của thuật toán.

1.2.3.2. Phương pháp nghiên cứu

Đọc hiểu về lý thuyết cách mã hóa dữ liệu, tìm chọn mô hình mã hóa tối ưu dữ liệu, từ đó sẽ cho hiệu quả nén tốt để kết quả đưa ra những mẫu dễ hiểu hơn, giảm dư thừa và khả dụng.

1.3. Ý TƯỞNG NÉN DỮ LIỆU

Đối với dữ liệu tuần tự dạng văn bản, để trích ra những mẫu hay những cụm từ với mong muốn cụm từ đó là đại diện cho một đoạn văn, là đặc trưng cho cho một dãy các từ thì một cách trực quan ta nhận thấy rằng những cụm từ nào xuất hiện nhiều lần trong đoạn văn bản đó thì có thể là một mẫu tuần tự đáng quan tâm.

Xuất hiện nhiều lần nhưng cách xuất hiện liên tục hay rời rạc thì mỗi cách xuất hiện mang ý nghĩa khác nhau, xuất hiện liên tục ví như mẫu abcd nằm liền nhau trong dãy thì mẫu đó có thể là đại diện cho dãy còn ngược lại mẫu abcd nằm rời rạc nhau mỗi từ cách nhau khá nhiều khoảng trống (gap) như nằm trong dãy S=ca__b___c____d_____. thì mẫu đó thường là mẫu tối nghĩa, khó hiểu không đáng quan tâm.

Vậy ý tưởng nén dữ liệu là gom các mẫu xuất hiện nhiều lần trong dãy lại thành một từ ta gọi là từ không đơn, đưa vào trong từ điển và mã hóa thành một từ mã dùng nó nén dãy hay nén dữ liệu.

Những mẫu có các từ nằm rời rạc nhau thì phạt khoảng trống, với khoảng trống nhỏ tương ứng với từ mã có chiều dài ngắn và khỏang trống lớn thì tăng đột biến chiều dài từ mã

VÍ DỤ:

Khoảng cách 1 nghĩa là nằm liền nhau thì mã hóa là 1 vậy E(1)=1. Khoảng cách 8 nghĩa là nằm cách nhau 8 khoảng trống thì mã hóa là 0001000 vậy E(8)=7.

Tóm lại nén dữ liệu là tìm ra những mẫu nén tốt nhất. Nén tốt nhất căn cứ theo hiệu quả nén đó là hiệu số tính theo bit của mô tả chiều dài dữ liệu trước và sau khi nén.

CHƯƠNG 2
CƠ SỞ LÝ THUYẾT

2.1. MÃ HUFFMAN

Đối với bài toán tìm mẫu nén tốt cho cơ sở dữ liệu tuần tự đặc biệt đối với dữ liệu dạng văn bản tốt nhất là dùng phương pháp mã hóa Huffman để nén dữ liệu.

Mã hóa Huffman là một thuật toán mã hóa dùng để nén dữ liệu. Nó dựa trên bảng tần suất xuất hiện các kí tự cần mã hóa để xây dựng một bộ mã nhị phân cho các kí tự đó sao cho dung lượng (số bít) sau khi mã hóa là nhỏ nhất.

Để mã hóa dữ liệu văn bản ta dùng các chuỗi nhị phân để mã hóa các từ. Trong một đoạn văn bản hay còn gọi là dãy các từ ta tính xem trong dãy đó có bao nhiêu từ khác nhau và tính tần suất của các từ đi kèm. Phương pháp mã Huffman là phương pháp có các tính chất cơ bản sau:

2.1.1. Mã tiền tố

Là bộ các từ mã của một tập hợp các kí hiệu sao cho từ mã của mỗi ký hiệu không là tiền tố (phần đầu) của từ mã một ký hiệu khác trong bộ mã.

VD -Hai từ mã 101 và 1011 thì từ mã trước nằm ở phần đầu của từ mã sau do đó không có tính chất của mã tiền tố.

-Hai từ mã sau 10 và 110 là hai từ mã thỏa tính chất mã tiền tố vì không có từ mã nào là tiền tố của từ mã còn lại.

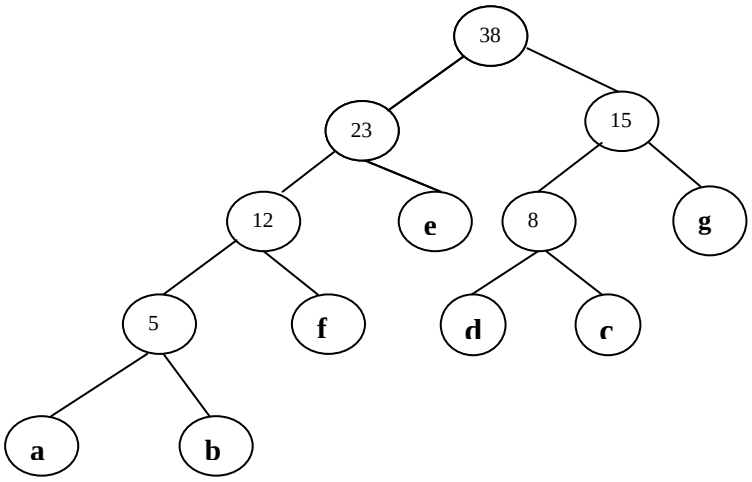
2.1.2. Cây nhị phân biểu diễn từ mã

Giả sử có n từ, ta ký hiệu w_1, w_2,…, w_n có các tần suất tương ứng t_1, t_2,…, t_n. Phương pháp mã hóa Huffman là phương pháp mã hóa sao cho chiều dài từ mã tỉ lệ nghịch với tần suất. Với phương pháp nầy xây dựng cây nhị phân n lá và tạo một bộ

mã tiền tố cho các từ bằng cách đặt mỗi từ mã vào một lá. Từ mã của mỗi kí hiệu được tạo ra khi đi từ gốc tới lá chứa ký hiệu đó, nếu đi qua cạnh trái thì ta thêm số 0, đi qua cạnh phải thì thêm số 1.

Cách xây dựng cây như sau: chọn hai nút lá có tần suất nhỏ nhất, vun lên thành một nút có trọng số là tổng trọng số của hai nút con, quá trình trên tiếp tục cho đến khi chỉ còn nút là gốc của cây.

VÍ DỤ: Cho các từ (a,2),(b,3),(c,5),(d,3),(e,9),(f,7),(g,7),(e,11) đi kèm theo là tần suất của các từ. Xây dựng cây nhị phân để mã hóa các từ trên.



Hình 2.1 Cây nhị phân mã Huffman

Kết luận mã hóa các từ theo phương pháp Huffman như sau:

Từ	a	b	c	d	e	f	g
Mã hóa	0000	0001	101	100	01	001	11

2.2. KHAI THÁC MẪU TUẦN TỰ NÉN

2.2.1. Khai thác mẫu tuần tự đóng

Cho S = $(e_1,t(e_1))$, $(e_2,t(e_2))$, …, $(e_n,t(e_n))$ biểu thị một dãy của những sự kiện, trong đó $e_i \in \Sigma$ là một sự kiện từ bảng chữ cái alphabet Σ và $t(e_i)$ là một nhãn thời gian của sự kiện e_i. Cho một dãy tuần tự P, ta nói S so khớp với P nếu P là một dãy con của S.

Coi б=$\{S_1, S_2,…, S_n\}$ là cơ sở dữ liệu của những dãy tuần tự. Số những dãy trong cơ sở dữ liệu so khớp với P được gọi là độ hỗ trợ (support) f_P của P. Bài toán khai thác những mẫu tuần tự phổ biến được định nghĩa như sau:

ĐỊNH NGHĨA *(Khai thác mẫu phổ biến)*

Cho một cơ sở dữ liệu б, giá trị độ hỗ trợ nhỏ nhất gọi là minsup, tìm tất cả những sự kiện P sao cho $f_P \geq$ minsup.

Một mẫu P được gọi là đóng nếu nó phổ biến ($\geq$ minsup) và không tồn tại mẫu Q phổ biến sao cho $f_P = f_Q$, P là con thực sự của Q. Bài toán khai thác tất cả những mẫu phổ biến đóng được định nghĩa như sau:

ĐỊNH NGHĨA *(Khai thác mẫu đóng)*

Cho một cơ sở dữ liệu tuần tự б và giá trị minsup, tìm tất cả những mẫu P sao cho $f_P \geq$ minsup và P đóng.

2.2.2. Nguyên lý nén dữ liệu

Từ tài liệu [4] các tác giả đã trình bày ngắn gọn nguyên lý MDL và cách tiếp cận khai thác mẫu dựa trên cơ sở MDL. Một mô hình M là tập những mẫu M=$\{P_1, P_2, …, P_n\}$ dùng để nén CSDL б.

Đặt $L^C(M)$ là chiều dài mô tả của mô hình M và $L^C(б |M)$ là chiều dài mô tả của CSDL б khi nó được mã hóa với sự trợ giúp của mô hình M và mã hóa C.

Vì thế tổng chiều dài mô tả của dữ liệu là $L^C_M(б)= L^C(M) + L^C(б |M)$. Sự khác nhau của các mô hình và cách mã hóa dẫn tới sự khác nhau của chiều dài mô tả của cơ sở dữ liệu. Từ đó nguyên lý MDL là tìm cách chọn ra mô hình M và mã hóa C sao cho $L^C_M(б) = L^C(M) + L^C(б |M)$ nhỏ nhất.

2.3. PHƯƠNG PHÁP MÃ HÓA DỮ LIỆU

2.3.1. Mã hóa và giải mã số tự nhiên

Trong việc mã hóa của dãy, ta cần phải biểu diễn nhị phân của những số tự nhiên dùng để chỉ ra số khoảng trống giữa những ký tự trong mã hóa một từ. Ta dùng mã hóa Elias được như sau: với số tự nhiên n biểu diễn nhị phân E(n) bắt đầu đúng cận dưới của $\log_2(n)$ bit 0 và theo sau bởi biểu diễn nhị phân của nó. Với cách nầy, chiều dài mã hóa Elias là $2[\log_2(n)]+1$ bits. Vậy chiều dài mã hóa Alias dài gấp đôi mã hóa chuẩn của n vì thế mã hóa khoảng trống ngắn hiệu quả hơn khoảng trống dài, đây là cách phạt những mẫu nằm rời ra trong dãy vì những mẫu nằm rời xa thì thường là những mẫu rời rạc và tối nghĩa.

VÍ DỤ *(mã hóa Elias)*

Mã hóa Elias được miêu tả trong bảng 2.1 cho tám số tự nhiên đầu tiên, số 8 được mã hóa Elias là 0001000 bắt đầu bởi phần nguyên cận dưới $\log_2 8=3$ số 0 sau đó biểu diễn nhị phân 1000 của số 8.

Bảng 2.1 Mã Alias [4]

Số	Mã Elias E(n)	Số	Mã Elias E(n)
1	**1**	**5**	**00101**
2	**010**	**6**	**00110**

3	**011**	**7**	**00111**
4	**00100**	**8**	**0001000**

Để giải mã chuỗi nhị phân chứa nhiều mã Elias rất đơn giản. Thật vậy việc giải mã đầu tiên đọc số bit 0 dẫn đầu cho tới khi gặp bit 1, đến lúc nầy ta biết cần phải đọc bao nhiêu bit nữa sẽ chấm dứt mã Elias hiên tại. Tiến trình này được lập lại cho mỗi khối mã Elias để giải mã hoàn tất một chuỗi nhị phân.

VÍ DỤ *(giải mã Elias)*

Chuỗi nhị phân 000100000100 đầu tiên đọc 3 số 0, gặp bit 1 ta biết sẽ đọc 3+1 bit tiếp theo 1000 là số 8 xong một khối, sau đó đọc 2 số 0 ta biết sẽ đọc 2+1 số tiếp theo là 100 là số 4 kết thúc, kết quả nhận được 84.

2.3.2. Phương pháp nén và hiệu quả nén

Cho $\Sigma = \{ a_1, a_2, \ldots, a_n\}$ là bảng chữ cái chứa tập hợp các ký tự a_i. Một tự điển D là một bảng với hai cột: cột thứ nhất chứa danh sách các từ $w_1, w_2, \ldots, w_n$ bao gồm tất cả những ký tự trong bảng chữ cái Σ, trong khi cột thứ hai chứa danh sách của những từ mã cho mỗi từ w_i trong từ điển và được ký hiệu là $C(w_i)$.Biểu diễn nhị phân của một từ điển được cho như sau: nó bắt đầu với n từ mã của tất cả những ký tự trong bảng chữ cái theo sau bởi biểu diễn nhị phân của các từ không đơn. Với mỗi từ không đơn w, biểu diễn nhị phân của nó chứa một dãy các từ mã của các ký tự của nó theo sau bởi biểu diễn của C(w). Ví dụ từ w=abc được biểu diễn trong tự điển như C(a)C(b)C(c)C(w). Biểu diễn nhị phân như trên cho phép ta đặt bất kỳ một từ nào vào trong từ điển cùng với từ mã của nó.

VÍ DỤ *(Từ điển nén dữ liệu và hiệu quả nén)*

Trong bảng 2.2, hai từ điển khác nhau D_1 và D_2 được đưa ra. Từ điển thứ nhất chứa cả từ đơn và không đơn trong khi từ điển thứ hai chỉ chứa các từ đơn

Bảng 2.2 Cách mã hóa và hiệu quả nén

Từ điển D1			Từ điển D2		
Từ	**Từ mã C_1**	**Tần suất**	**Từ**	**Từ mã C_2**	**Tần suất**
a	**000**	**2**	**a**	**01**	**4**
b	**011**	**2**	**b**	**10**	**4**
c	**010**	**2**	**c**	**11**	**4**
d	**011**	**2**	**d**	**000**	**2**
e	**11**	**2**	**e**	**001**	**2**
abc	**10**	**4**			
S=abcabdcaebc					
$C_1(abc)E(1)E(1)C_1(abc)E(1)E(2)C_1(d)C_1(abc)\ E(2)E(1)C_1(e)$ $L^{C1}(S)$= 46 bits			$C_1(a)C_1(b)C_1(c)C_1(a)C_1(b)C_1(d)\ C_1(c)C_1(a)C_1(e)C_1(b)\ C_1(c)$ $L^{C1}(S)$= 57 bits		

Hiệu quả nén khi thêm mẫu abc vào từ điển để nén dãy S, benefit(abc)= 57-46= 11 bits.

Vậy vấn đề của ta là tìm mẫu nào nén dữ liệu hiệu quả nhất đưa vào từ điển.

2.4. MÃ HÓA VÀ GIẢI MÃ MỘT DÃY CÁC TỪ

Cho tự điển D, một dãy S được mã hóa bởi việc thay thế các vùng (instance) của những từ trong tự điển trong dãy bởi những con trỏ. Một con trỏ p thay thế một vùng của một từ w trong dãy S là một dãy của những bit bắt đầu bởi từ mã C(w) theo sau là danh sách các mã Elias chỉ ra khoảng trống của các ký tự liền kề trong vùng của từ trong S. Trong trường hợp là từ đơn con trỏ chỉ chứa từ mã tương ứng.

VÍ DỤ *(con trỏ)*

Xét dãy S = abcabd_caebc có ba vùng chứa từ w=abc tại các vị trí (1,2,3), (4,5,8) và (8,11,12) được gạch dưới. Nếu từ abc đã có trong từ điển với từ mã là C(w) thì ba lần xuất hiện trên có thể thay thế bởi ba con trỏ p_1=C(w)E(1)E(1), p2=C(w)E(1)E(3), p3=C(w)E(2)E(1).

Mã hóa một dãy có thể được định nghĩa như sau:

ĐỊNH NGHĨA *(mã hóa một dãy) [4]*

Cho một từ điển D, mã hóa một dãy của S là sự thay thế những vùng của những từ trong tự điển bởi con trỏ.

Mã hóa trong định nghĩa trên được hoàn tất nếu tất cả những ký tự trong dãy S được mã hóa. Trong việc nầy, chúng ta xem xét chỉ những mã hóa đã hoàn tất. Trong mã hóa C của một dãy S, tần suất của một từ w ký hiệu là $f_C(w)$ được định nghĩa như là số lần từ w được thay thế bởi con trỏ cộng với số lần từ được biểu diễn trong biểu diễn nhị phân của từ điển.

VÍ DỤ *(Mã hóa dãy)*

Trong bảng 1.1, tự điển D_1 và D_2 được tạo cơ sở dựa trên mã hóa C_1 của dãy S=abcabdcaebc. Mã hóa C_1 thay thế ba lần từ abc trong dãy S bởi con trỏ. Vì thế tần suất của từ abc được đếm như là số con trỏ thay abc cộng với số lần xuất hiện trong tự điển vậy $f_C(abc)=4$. Trong khi mặc dù a không thay thế bất kỳ một con trỏ nào nhưng nó hiện diện hai lần trong biểu diễn nhị phân của từ điển vì thế $f_C(a)=2$. Tương tự như thế cho tần suất của những từ khác.

Với mỗi từ w, biểu diễn nhị phân của từ mã C(w) phụ thuộc vào tần suất của nó trong việc mã hóa. Gọi $F_C= \sum_{w \in D} f_C(w)$ là tổng của những tần suất trong tất cả những từ trong từ điển của mã hóa C. Tần suất liên quan của mỗi từ w xác định như là $\frac{f_C(w)}{F_C}$ có thể xem như phân phối xác suất định nghĩa trong không gian của tất cả các từ trong tự điển bởi vì $\sum_{w \in D} \frac{f_C(w)}{F_C}=1$.

Vậy tồn tại một mã hóa tiền tố C(w) sao cho chiều dài mã hóa cân xứng vật lý với tần suất của từ, nghĩa là $|C(w)| \sim -\log\frac{f_C(w)}{F_C}$, nghĩa là những từ mã ngắn được gán đến những từ có tần suất lớn. Như vậy mã hóa được tối ưu trong tất cả những kết quả mã hóa trong phân phối tần suất của các từ trong từ điển.

Tiếp tục nói về giải thuật giải mã một dãy đã được mã hóa. Đầu tiên nói về cách làm thế nào để đọc một từ điển đã được biểu diễn nhị phân. Một biểu diễn nhị phân của một từ điển có thể được giải mã như sau:

- Đọc từ mã của tất cả những từ đơn cho đến khi xãy ra sự lập lại của bất kỳ một từ đơn nào.
- Từng bước đọc từ mã của những từ không đơn w bằng cách đọc nột dung của từ w, đọc một dãy các từ đơn đã biết cho đến khi đọc đến từ mã chưa thấy bao giờ đó là C(w) từ mã từ không đơn trong từ điển.

VÍ DỤ *(Giải mã từ điển)*

Từ điển D_1 trong bảng 2.1 có biểu diễn nhị phân $C_1(a)C_1(b)C_1(c)C_1(d)C_1(e)C_1(a)$ $C_1(b)C_1(c)C_1(abc)$. Đầu tiên đọc tất cả từ mã của tất cả các từ đơn a,b,c,d,e. Dừng lại khi lập lại một từ mã của từ đơn xãy ra, trong trường hợp nầy khi ta thấy lặp lại từ mã của từ đơn a là $C_1(a)$. Trình giải mã biết rằng đó là bắt đầu của một từ không đơn vì thế nó tiếp tục đọc những từ mã theo sau cho đến khi gặp một từ mã chưa thấy trước đó bao giờ $C_1(abc)$ đó là từ mã sau cùng của từ không đơn abc trong từ điển.

Cho trước một từ điển, một dãy được giải mã bởi đọc mỗi khối của chuỗi nhị phân theo từ bởi con trỏ. Mỗi khối được đọc như sau:

- Đọc từ mã C(w) và tham chiếu đến từ điển để nhận thông tin về từ w.
- Nếu từ w là một từ đơn thì tiếp tục đọc khối kế tiếp. Ngược lại dùng mã hóa Elias nhận |w|-1 khoảng trống trước khi tiếp tục đọc khối kế tiếp.

Ví dụ sau đây nói lên cách làm thế nào giải mã một dãy C_1(abc)E(1)E(1)C_1(abc)E(1)E(2)C_1(d)C1(abc)E(2)E(1)C_1(e) với sự hỗ trợ tự điển D_1.

VÍ DỤ *(Giải mã một dãy)*

Trình giải mã đầu tiên đọc C_1(abc) rồi tham chiếu đến từ điển biết chiều dài là 3 vì thế đọc hai số dùng mã Elias. Tiếp tục đọc khối kế tiếp C_1(abc)E(1)E(2) cũng với cách tương tự để mã hóa vùng của abc. Sau khi gặp từ mã C_1(d) tham chiếu đến từ điển nói rằng không có khoảng trống theo sau vì thế trình giải mã tiếp tục đọc khối kế tiếp với cách tương tự như thế để giải mã vùng cuối cùng của abc và từ đơn e.

2.5. BÀI TOÁN TÌM MẪU NÉN

2.5.1. Định nghĩa (Bài toán nén dãy) [4]

Ký hiệu $g_C(w)$ là tổng chi phí của mã hóa các khoảng trống bởi mã Elias của từ w trong mã hóa C. Vậy thì chi phí khoảng trống cho tất cả những từ đơn là 0. Chiều dài mô tả của CSDL б mã hóa bởi mã C có thể được tính toán như sau:

$$L^C(б) = \sum_{w \in D}(|C(w)| * f_c(w) + g_C(w)) \quad (1)$$

$$= \sum_{w \in D}(\log \frac{F_C}{f_C(w)} * f_c(w) + g_C(w)) \quad (2)$$

Ta ký hiệu $L_D^{C_D^*}(б)$ là chiều dài của CSDL б trong mã hóa tối ưu C_D^* khi đã cho từ điển D. Bài toán tìm những mẫu nén được đề ra như sau:

Cho CSDL tuần tự б, một từ điển tối ưu D^* và mã hóa tối ưu C_D^* để dùng những từ trong tự điển D^* mã hóa CSDL б sao cho

$$D^* = argmin_D L_D^{C_D^*}(б).$$

Để giải quyết vấn đề nén một dãy chúng ta cần phải tìm cùng một lúc từ điển tối ưu D^* và mã hóa tối ưu C^*_D và dùng từ điển D^* để mã hóa CSDL б.

2.5.2. Kết luận

Vậy qua nghiên cứu các thuật toán ta thấy rằng độ phức tạp cao cho nên khó thể dùng phương pháp duyệt hết mọi trường hợp để tìm mẫu, vì trong mỗi lần duyệt như vậy, khi tổ hợp bùng nổ trong mỗi mẫu ta phải tính toán hiệu quả nén rất phức tạp do đó phải tìm ra một phương pháp tốt hơn để tránh sự bùng nổ của tổ hợp.

CHƯƠNG 3
THUẬT TOÁN KHAI THÁC MẪU NÉN

Như đã trình bày ở trên việc khai thác mẫu tuần tự đóng có nhiều mặt hạn chế như gặp phải mẫu trùng lắp, tối nghĩa. Để khắc phục vấn đề trên trong đoạn nầy trình bày hai thuật toán Heuristic lấy ý tưởng của thuật toán Krimp, bài toán khai thác những mẫu dùng để nén cơ sở dữ liệu. Ta nghiên cứu hai thuật toán SeqKrimp và GoKrimp, Seqkrimp là bước đệm tìm mẫu nén nhưng thuật toán chính là thuật toán GoKrimp hiệu quả hơn, tìm ra những mẫu khả dụng và dễ hiểu hơn.

3.1. THUẬT TOÁN SEQKRIMP

Trước khi triển khai thuật toán SeqKrimp ta nghiên cứu các thuật các thuật toán thành phần cần có để SeqKrimp cần gọi.

3.1.1. Thuật toán lấy mẫu tuần tự đóng GetCandidate()

GetCandidate() là thuật toán lấy các mẫu tuần tự đóng theo một độ hỗ trợ tối thiểu cho trước (minsup) rồi từ mẫu những mẫu nầy SeqKrimp sẽ chọn lại những mẫu nén tốt nhất và trích ra.

VÍ DỤ: *(Khai thác mẫu tuần tự đóng)*

Cho cơ sở dữ liệu tuần tự như sau:

S1= (a,1)(b,2)(a,3)(b,4)(c,5)

S2= (a,1)(b,2)(a,3)(b,4)(c,5)

S3= (a,1)(b,2)(a,3)(b,4)

S4= (a,1)(b,2)(a,3)(b,4)

S5= (a,1)(b,2)(a,3)(b,4)(c,5)

S6= (a,1)(b,2)(a,3)(b,4)(c,5)

Dùng hàm GetCandidate() để lấy tập mẫu tuần tự đóng cho trước minsup=4/6.

Bảng tổ hợp các mẫu và độ hỗ trợ

Mẫu	a	ab	aa	ac	b	ba	bb	bc
Sup	6/6	6/6	6/6	4/6	6/6	6/6	6/6	4/6
Mẫu	c	aba	abb	aab	aac	bab	bac	abc
Sup	6/6	6/6	6/6	6/6	4/6	6/6	4/6	4/6

Các mẫu được chọn là

C={a, ab, aa, b, ba, bb, c, aba, abb, aab, aac, bab, bac} do chỉ chọn các mẫu đóng nên tập ứng viên mẫu tuần tự đóng chỉ còn lại.*(Gạch dưới là con thực sự mà cùng supp)*

C={aba, abb, aac, bab, bac} là tập mẫu tuần tự đóng thỏa support.

3.1.2. Thuật toán so khớp với chi phí tối thiểu MinGapMatch [4]

Vấn đề chính trong thủ tục nầy là cho trước một dãy S và một mẫu P, tìm vị trí của P so khớp trong S có chi phí khoảng trống nhỏ nhất.

Bài toán đặt ra là tìm những vị trí so khớp của mẫu P = a_1, a_2, …, a_k có khoảng trống (gap) nhỏ nhất. Đặt $l_{a_1}, l_{a_2}, \dots, l_{a_k}$ là danh sách kết hợp với những từ của mẫu P, phần tử thứ j của danh sách l_{a_i} chứa hai giá trị ký hiệu là $l_{a_i}^1[j]$ và $l_{a_i}^2[j]$. Giá trị thứ nhất $l_{a_i}^1[j]$ chứa vị trí của a_i trong dãy S, giá trị thứ hai $l_{a_i}^2[j]$ chứa chiều dài mã hóa Alias,so khớp chấm dứt tại vị trí $l_{a_i}^1[j]$.

Thuật toán tìm so khớp với từ P với chi phí khoảng trống nhỏ nhất bằng cách duyệt qua danh sách l_{a_i} với i=1,2,…,k và tính $l_{a_i}^2[j]$ bởi công thức sau:

$$l_{a_i}^2[j] = min_p\{l_{a_{i-1}}^2[p] + E(l_{a_i}^1[j] - l_{a_{i-1}}^1[p])\} \quad (3)$$

Trong đó:

$l_{a_1}^2[j] = 0$ với j = 1, 2, …, $|l_{a_1}|$

E là hàm đo chiều dài của mã hóa Alias ví dụ số 3 biểu diễn 011 vậy E(3)=3.

- Đoạn mã giả [4]:

Thuật Toán minGapMatch(S_t,P)

1: **Input**: Dãy S_t, từ P = a_1, a_2, …, a_k và danh sách các vị trí $l_{a_1}, l_{a_2}, ..., l_{a_k}$

2: **Output**: So khớp với chi phí khoảng trống nhỏ nhất

3: **for** i=1 to k **do**

4: **for** j=1 to $|l_{a_i}|$ **do**

5: $l^2_{a_i}[j] = min_p \{l^2_{a_{i-1}}[p] + E(l^1_{a_i}[j] - l^1_{a_{i-1}}[p]) \}$

6: **endfor**

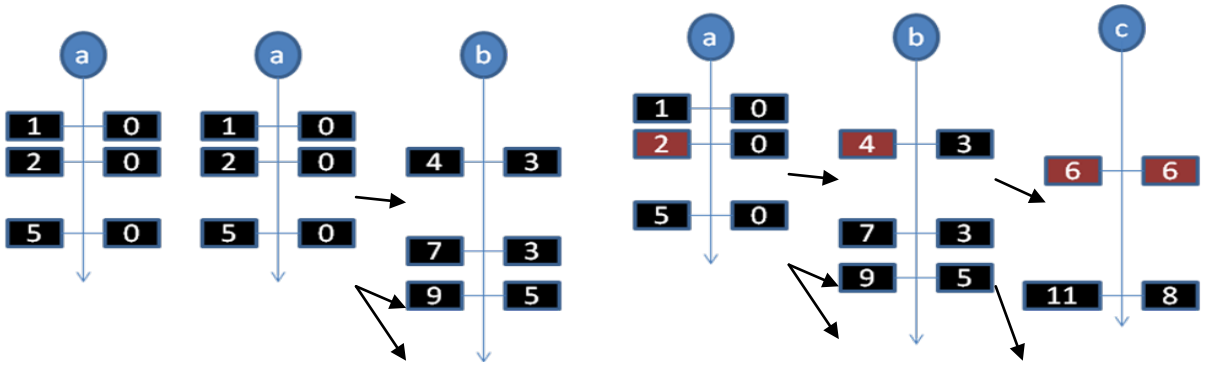
7: **endfor**

8: **Trả về** so khớp với chi phí nhỏ nhất

- VÍ DỤ MINH HỌA:

Cho dãy S=(b,0),(a,1),(a,2),(b,4),(a,5),(c,6),(b,7),(b,9),(c,11).

S= baa_bacb_b_c và mẫu P=abc. Tìm mẫu P tại vị trí nào có chi phí khoảng trống nhỏ nhất.



Hình 3.1 So khớp khoảng cách tối thiểu[4]

Các bước cụ thể của thuật toán

Bước 1: l_a chứa ba phần tử $l_a^1 = 1$, $l_a^1 = 2$, $l_a^1 = 5$ chỉ ra vị trí của a trong dãy S. Đầu tiên khởi tạo giá trị thứ hai của danh sách l_a^2 [p]=0 (p=1,2,3).

Bước 2: $l_b$chứa bốn phần tử l_b^1=0,$l_b^1 = 4$, $l_b^1 = 7$, $l_b^1 = 9$ chỉ ra vị trí của b trong dãy S. Theo công thức (3) chúng ta tính giá trị thứ hai của danh sách l_b như sau, với trường hợp l_b^2 .

l_b^2 = $\min_p$ { l_a^2 [p] + E($l_b^1 - l_a^1$ [p]) }

= l_a^2 + E($l_b^1 - l_a^1$)= 3

Vẽ một mũi tên nối l_a và l_b.Tương tự như trên tính l_a^2 = 3 và l_a^2 =5.

Bước 3:l_c chứa hai phần tử $l_c^1 = 6$, $l_c^1 = 11$ chỉ ra vị trí của c trong dãy S.

Những giá trị $l_c^2 = 6$, $l_c^1 = 8$ được tính tương tự như trên. Trong số chúng thì l_c^2= 6 bits là giá trị nhỏ nhất vì thế so khớp của abc trong S với chi phí khoảng trống nhỏ nhất theo vùng của abc trong dãy S tại vị trí (2,4,6).

Kết luận: Tìm được so khớp abc trong S có chi phí khoảng trống nhỏ nhất tại (2,4,6).

3.1.3. Thuật toán tính hiệu quả nén Compress [4]

Cho trước một dãy S và một mẫu P. Vấn đề là nén dãy S bởi mẫu P và tính hiệu quả nén của nó.

- Đoạn mã giả [4]:

Thuật toán Compress (Ϭ/P)

1: **Input**: CSDL tuần tự ϭ ={S_1,S_2, …, S_n } và một mẫu P

2: **Output**: Hiệu quả nén của việc thêm P vào từ điển D

3: $L^{C_0}(\sigma) \leftarrow$ chiều dài nguyên thủy của dữ liệu

4: **for** $S_i \in$ ϭ **do**

5: **while** S_i có chứa vùng P **do**

6: s ← minGapMatch(S_i,P)

7: thay thế s bởi con trỏ đến P trong tự điển

8: **endwhile**

9: **endfor**

10: L(ϭ | D∪{P}) ← chiều dài của dữ liệu sau khi thêm P

11: Trả về $L^{C_0} - L(\sigma \mid D \cup \{P\})$

- VÍ DỤ MINH HỌA:

Cho dãy S=bacdbcacbadc và mẫu P=bac. Tìm hiệu quả nén.

Giải: Ta có bộ từ điển ban đầu D={a, b, c, d}và thêm vào mẫu bac được bộ từ điển D={a,b,c,d,abc} sau đó tính tần suất của từng từ:

Từ	a	b	c	d	bac
Tần suất	1	1	2	3	4

Vậy mã Huffman:

Từ	a	b	c	d	bac
Từ mã	000	001	01	10	11

Qua đó biết chiều dài từng từ mã và tính hiệu quả nén: Vì có 3 lần so khớp gặp mẫu bac, lần đầu hiệu quả 4=8-(2+1+1), lần hai là 2, lần ba là 2.

Vậy hiệu quả nén của mẫu bac, benefit(bac)=8.

3.1.4. Thuật toán khai thác mẫu nén SeqKrimp [4]

Trong phần nầy, giới thiệu thuật toán khai thác những mẫu nén từ một CSDL tuần tự tương tự thuật toán Krimp cho dữ liệu Itemset. SeqKrimp mô tả trong thuật toán bao gồm hai pha. Trong pha thứ nhất, một tập hợp các mẫu ứng viên được phát sinh bởi dùng thuật toán khai thác mẫu tuần tự đóng mẫu tuần tự đóng GetCandidate().

Trong pha thứ nhì, thuật toán SeqKrimp chọn một tập những mẫu tốt từ tập các ứng viên cơ sở dựa trên một thủ tục tham lam ComPress.

- Đoạn mã giả [4]:

Thuật Toán SeqKrimp(б)

1: **Input**: CSDL б
2: **Output**: Những mẫu nén
3: C ← **GetCandidate**()
4: D ← Σ
5: **while** C ≠ 0 Benefit (P^*) > 0 **do**
6: **for** P ∈ C **do**
7: Benefit(P) ← **Compress** (б | P)
8: **end for**

9: $P^* \leftarrow \text{argmax}_P \text{Benefit}(P)$

10: **if** $\text{Benefit}(P^*) \leq 0$ **then**

11: break

12: **endif**

13: $D \leftarrow D \cup \{P^*\}$

14: $C \leftarrow C \setminus \{P^*\}$

15: Dùng thuật toán 1 để thay thế tất cả những vùng của P^* bởi những con trỏ

16: **endwhile**

17: Trả về D

- VÍ DỤ MINH HỌA:

Giả sử có CSDL như trong [4], ta có các bước thực hiện như sau:

S_1=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

S_2=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

S_3=(a,1)(b,2)(c,3)(a,4)(b,5)

S_4=(a,1)(b,2)(c,3)(a,4)(b,5)

S_5=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

S_6=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

Khai thác mẫu nén với thuật toán SeqKrimp

-Trước tiên ở pha thứ nhất thuật toán GetCandidate() cho ta tập ứng viên các mẫu tuần tự đóng C={abc,cb,ba}.

-Ở pha thứ hai, tính hiệu quả nén khi đưa từng mẫu vào từ điển D= {a,b,c} sau đó tính hiệu quả nén ta được benefit(abc)= 20, benefit(cb)= -5, benefit(ba)= -5.

Mẫu abc có hiệu quả nén dương cao nhất đưa vào từ điển.

D={a,b,c,abc}

Xóa abc khỏi cơ sở dữ liệu vậy dữ liệu còn lại là:

S_1=…….

S_2=…….

S_3=……(a,4)(b,5)

S_4=……(a,4)(b,5)

S_5=……..

S_6=…….

Lặp lại quá trình trên với cơ sở dữ liệu mới cập nhật và tính hiệu quả nén của mẫu ba và cb ta thấy không có hiệu quả nén dương nào, thuật toán dừng.

Kết luận: Trích được mẫu nén abc.

Tốc độ thực thi nhanh hay chậm của SeqKrimp phụ thuộc hoàn toàn vào tốc độ của Getcandidate.

Để khắc phục tình trạng trên đưa ra thuật toán GoKrimp, tự phát sinh mẫu mà không cần nhờ lấy mẫu của GetCandidate().

3.2. THUẬT TOÁN GOKRIMP

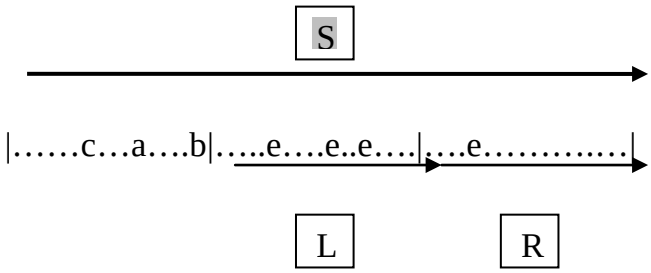
Thuật toán GoKrimp khởi đầu là những từ đơn F các sự kiện phổ biến, sự kiện nào sẽ được chọn để tránh phải sự bùng nổ tổ hợp. Đưa ra một kỹ thuật kiểm tra sự phụ thuộc của một sự kiện e vào một mẫu P, sự kiện e gọi là phụ thuộc vào mẫu P nếu thỏa kiểm tra dấu

3.2.1. Kiểm tra sự kiện có liên quan

Tính hiệu quả nén của lần mở rộng mẫu tiêu tốn nhiều thời gian bởi vì nhiều lần tìm kiếm cho so khớp khoảng trống nhỏ nhất của phần nới rộng trong cơ sở dữ

liệu. Vì thế, tập hợp các sự kiện chọn để nới rộng một mẫu được giới hạn đến tập hợp của những sự kiện có liên quan đến những việc xãy ra của mẫu đã cho. Sau đó, sự kiện khi thêm vào mẫu đã cho có hiệu quả nén tốt nhất được chọn để nới rộng mẫu này.

Để kiểm tra sự liên quan giữa một mẫu P và một sự kiện e dùng phương pháp kiểm tra dấu (signtest). Cho m cặp số nguyên (X_1, Y_1), (X_2, Y_2) … (X_m, Y_m), ký hiệu N^+ là số những cặp sao cho $X_i > Y_i$ với i=1, 2,…,m.



Hình 3.2 Kiểm tra sự kiện liên quan

Kiểm tra dấu được áp dụng để kiểm tra sự phụ thuộc giữa một mẫu P và một sự kiện e như sau. Với mỗi dãy S ∈ б và một sự kiện c ∈ P ký hiệu S(c) phía trái nhất vùng của c trong S.

Xét khoảng cách bên phải sau vị trí cuối cùng của S(c) như minh họa trong hình3.1, khoảng cách nầy được chia thành hai phần bằng nhau là L và R, ký hiệu tần suất của sự kiện e trong hai khoảng con là L và R tương ứng. Ta ký hiệu tần suất của sự kiện e trong hai khoảng con là L_e và R_e . Nếu sự kiện e là độc lập với sự xuất hiện của S(c), chúng ta kỳ vọng rằng e xuất hiện ngẫu nhiên bên trái L và phải R là như nhau. Vì thế số của những dãy trong đó ta thấy $L_e > R_e$ có thể được dùng như là một kiểm tra thống kê trong kiểm tra dấu cho việc kiểm tra sự liên quan của sự kiện e và mẫu c. Kiểm tra được thực hiện với mỗi sự kiện c ∈ P và sự kiện e được xem như liên quan đến với mẫu P nếu nó vượt qua tất cả những kiểm tra phụ thuộc

đối với tất cả những sự kiện thuộc về P. Khi một kiểm tra được thực hiện chúng ta giữ lại đoạn mà kết quả phụ thuộc dùng lại cho giai đoạn sau.

- VÍ DỤ MINH HỌA:

Giả sử có CSDL như trong [4], ta có các bước thực hiện như sau:

S_1=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

S_2=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

S_3=(a,1)(b,2)(c,3)(a,4)(b,5)

S_4=(a,1)(b,2)(c,3)(a,4)(b,5)

S_5=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

S_6=(a,1)(b,2)(c,3)(a,4)(b,5)(c,6)

Cho trước số dãy thỏa kiểm tra N=4, nếu L_e=R_e thì vượt qua kiểm tra dấu. Với mẫu P={a} thì sự kiện nào có liên quan với P?

Dãy S_1 sau từ a cuối cùng ta chia làm hai khoảng L={b} và R={c} có b.

Tương tự cho các dãy sau, sự kiện b thỏa kiểm tra dấu cả 6 dãy >N=4.

Kết luận: sự kiện b có liên quan với mẫu P={a}. Tiếp tục như thế ta tìm những sự kiện có liên quan đến mẫu P={ab}…

3.2.2. Thuật toán nới rộng mẫu GetNextPattern [4]

Thuật toán dùng để tự khai thác trực tiếp mẫu nén, thay thế cho thuật toán GetCandidate() của SeqKrimp().

- Đoạn mã giả [4]:

Thuật Toán GetNextPattern(б)

1: **Input:** CSDL tuần tự б = {S_1,S_2, …, S_n }

2: **Output:** P xấp xỉ của mẫu nén tốt nhất

3: P $\leftarrow$ {$\emptyset$}

4: F $\leftarrow$ những sự kiện phổ biến

5: **while** (True) **do**

6: **for** e $\in$ F **do**

7: Benefit(e) $\leftarrow$ **Compress**(D | P,e)

8: **endfor**

9: e^* $\leftarrow$ $argmax_e$ Benefit(e)

10: **if** Benefit(e^*) < 0 **then**

11: break

12: **endif**

13: P $\leftarrow$ P.e^*

14: б $\leftarrow$ б ước lượng đến e^*

15: F $\leftarrow$ những sự kiện phổ biến liên quan

16: **endwhile**

17: Trả về P

3.2.3. Thuật toán khai thác trực tiếp mẫu nén GoKrimp [4]

Đây là thuật toán trọng tâm của đề tài, sử dụng những ý tưởng, các thuật toán thành phần đã trình bày phần trên.

- Đoạn mã giả [4]:

Thuật Toán Khai thác trực tiếp mẫu nén GoKrimp()

1: **Input**: Cơ Sở Dữ Liệu tuần tự ϭ = {S_1,S_2, …, S_n }

2: **Output:** Tập những mẫu nén

3: D ← Σ

4: **while** Benefit(P^*) > 0 **do**

5: P^* ← **GetNextPattern**(ϭ)

6: **if** Benefit(P^*) ≤ 0 **then**

7: break

8: **endif**

9: D ← D ∪ {P^*}

10: Dùng thuật toán 1 để thay thế tất cả những vùng của P^* bởi những con trỏ

11: **endwhile**

12: Trả về D

- VÍ DỤ MINH HỌA:

Cho Cơ sở dữ liệu tuần tự chứa các dãy tuần tự như sau:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
S1=	**a**	**c**	**a**	**e**	**b**	**a**	**d**	**e**	**a**	**d**	**b**	**c**	**d**	**e**
S2=	**e**	**a**	**e**	**c**	**b**	**d**	**a**	**b**	**c**	**d**	**e**			
S3=	**a**	**b**	**c**	**e**	**d**	**a**	**c**	**b**	**d**	**c**	**b**	**c**	**e**	
S4=	**d**	**a**	**e**	**c**	**b**	**a**	**d**	**b**	**c**	**b**	**d**	**e**		

(a,b,c,d,e là các từ đơn)

Xác lập bảng chữ cái đưa vào từ điển D={a,b,c,d,e}

Bảng phân phối xác suất để tính chiều dài từ mã |C(w)| $\sim -\log\frac{f_C(w)}{F_C}$

	a	b	c	d	e
Tần suất	11	11	11	11	11
Chiều dài	2∼log(55/11)	2∼log(55/11)	2∼log(55/11)	3∼log(55/11)	3∼log(55/11)

- Mẫu P= $\emptyset$
- Những sự kiện phổ biến đưa vào F

  F= {a,b,c,d,e}

Khai thác trực tiếp mẫu

Nới rộng mẫu

Với mỗi sự kiện x $\in F$ ta nới rộng mẫu dùng thuật toán GetNextPattern(б)

* Nới rộng với x= a: Đặt độ sai khác kiểm tra dấu là 0.01 và số cặp vượt qua kiểm tra dấu là 4 cặp.

- Sự kiện b có liên quan với a do kiểm tra dấu

  Sự kiện c có liên quan với ab

Vậy ta được mẫu nới rộng là abc.

Tính hiệu quả nén

	a	b	c	d	e	abc
Tần suất	8	8	8	11	11	6
Chiều dài	3	3	3	2	2	3

Hiệu quả nén của abc, benefit(abc)=3+6+6+0+3=18.

- Sự kiện e không có liên quan với a do kiểm tra dấu

Tương tự như thế ta nới rộng mẫu và tính hiệu quả nén

*Nới rộng với x= b: không có sự kiện liên quan

*Nới rộng với x= c: không có sự kiện liên quan

*Nới rộng với x= d: không có sự kiện liên quan

*Nới rộng với x= e: không có sự kiện liên quan

Vậy ta có một mẫu có hiệu quả nén dương là abc , mẫu có hiệu quả nén tốt nhất benefit(abc)=20 đưa abc vào bộ từ điển D.

D={a,b,c,d,e,abc}

Hiệu chỉnh lại CSDL ổ theo mẫu abc

Xóa vùng của abc trong Cơ sở dữ liệu bằng cách dùng thuật toán qui hoạch động mingapMatch, tương tự như thế tính hiệu quả nén của ac theo từ điển mới cập nhật D={a,b,c,d,e,abc}

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
S1=	**a**	**c**	**a**	**e**	**b**	**a**	**d**	**e**	**_**	**d**	**_**	**_**	**d**	**e**
S2=	**e**	**a**	**e**	**c**	**b**	**d**	**_**	**_**	**_**	**d**	**e**			
S3=	**_**	**_**	**_**	**e**	**d**	**_**	**_**	**b**	**_**	**c**	**b**	**c**	**e**	
S4=	**d**	**a**	**e**	**c**	**b**	**_**	**d**	**_**	**_**	**b**	**d**	**e**		

Tương tự như trên tính hiệu quả nén benefit(ac)= 2 duy nhất đưa ac vào bộ từ điển

D={a,b,c,d,e,abc,ac} thuật toán kết thúc.

Kết luận

Mẫu được trích ra là **abc, ac.**

CHƯƠNG 4
THỰC NGHIỆM VÀ KẾT LUẬN

Trong phần trình bày lại kết quả thực nghiệm của tài liệu tham khảo [4]. Các tác giả sẽ so sánh tập những mẫu được đưa ra bởi SeqKrimp và GoKrimp đối với những thuật toán cơ bản sau:

- BIDE: BIDE được chọn bởi vì nó là một trạng thái của tiếp cận khéo léo của khai thác những mẫu tuần tự đóng. BIDE cũng được dùng để phát sinh những tập ứng viên cho SeqKrimp, hàm **GetCandidate()**.
- SQS: được đề xuất gần đây bởi Tatti và Vreeken[6] cho việc khai thác những mẫu nén trong CSDL tuần tự.
- pGOKRIMP: là thế hệ trước của thuật toán GoKrimp (ký hiệu là pGOKRIMP) đã được công bố trước đây. Ta dùng cả pGOKRIMP trong việc so sánh để minh chứng hiệu quả của sự duyệt lại mã hóa được làm theo bởi thuật toán GoKrimp.

Bảng 4.1 Mẫu tuần tự đóng[4]

Phương pháp	**Mẫu**			
SEQKRIMP	Support vector machin	State art	Compon alnalysi	Solv problem
	Real word	Hight demension	Sampl size	Kernel kernel kernel
	Machin learn	Larg scale	Supervis learn	Model select
	Data set	Futur select	Support vector	Train set
	Bayesian netword	Experiment result	Loss function	Loss loss
GOKRIMP	Support vector machin	State art	Neural netword	Well know
	Real word	Hight demension	Expirement result	Special case
	Machin learn	Reproduct hilbert space	Sampl size	Solv problem
	Data set	Larg scale	Supervis learn	Signific improv
	Bayesian netword	Independ compon alnalysi	Support vector	Object function
SEARCH_SQS	Support vector machin	Larg scale	Futur select	Sampl size

	Real word	Nearest neighbor	Graphic model	Learn algorithm
	State art	Decis tree	Real word	Princip compon
	Data set	Neural network	Hight demension	Logist regress
	Bayesian netword	Cross valid	Mutual inform	Model select
pGOKRIMP	Machin learn learn learn	Result show	Result result	Present algorithm
	Algorithm Algorithm Algorithm	Paper data	Machin learn	Such data
	Data data data	Problem problem	Perform perform	Learn learn
	Data set data set	Set set	Paper propose	Show show
	Method method method	Model model gener	Machine kernel	Function function

Những mẫu được khai thác bởi SeqKrimp,GoKrimp,SQS và pGOKRIMP.

Mỗi bộ dữ liệu là một cơ sở dữ liệu tượng trưng những dãy các khoảng với những nhãn lớp.

Bảng 4.2 Bộ dữ liệu[4]

Tập dữ liệu	Sự kiện	Dãy	Lớp
Jmlr	787	75646	NA
Parallel	1000000	10000	NA
aslbu	36500	441	7
Aslgt	178494	3493	40
Auslan2	1800	200	10
Pioneer	9766	160	3
Context	25832	240	5
Skating	37186	530	7
unix	295008	11133	10

Tóm tắt của những bộ dữ liệu

Hơn nữa hai tập dữ liệu cũng được dùng để đánh giá những tiếp cận được đề xuất trong giới hạn của việc làm sáng tỏ các mẫu. Tập dữ liệu đầu tiên JMLR chứa 787 bản tóm tắt của *Tạp chí nghiên cứu máy học* . JMLR được chọn bởi vì nó tiềm tàng những mẫu quan trọng và dễ dàng làm dễ hiểu. Tập dữ liệu thứ nhì là dữ liệu tự tạo với những mẫu đã biết. Với tập dữ liệu nầy các tác giả đánh giá những thuật

toán được đề xuất cơ sở dựa trên độ chính xác của tập những mẫu trả về bởi mỗi thuật toán. Đánh giá được thực hiện với 4 x 2.4 GHz, 4GB Ram, Fedora 10/64-bit.

Tóm tắt, những cách tiếp cận đã đề xuất được đánh giá theo những tiêu chuẩn sau:

Tính dễ hiểu để định giá một cách không chính thức tính có nghĩa và dư thừa của những mẫu.
Thời gian thực thi đo lường tính hiệu quả của những tiếp cận.
Độ nén đo lường mức độ nén dữ liệu.
Phân lớp chính xác đo lường tính hữu ích của tập các mẫu.

4.1. BỘ DỮ LIỆU THỬ NGHIỆM

4.1.1. Bộ dữ liệu JMLR

Khai thác những miêu tả của mẫu thì không giám sát được, rất khó so sánh những tập khác nhau của những mẫu trong trường hợp tổng quát. Hơn nữa với dữ liệu văn bản nó có thể làm rõ nghĩa mẫu trích ra. Trong việc này các tác giả so sánh những thuật toán khác nhau trên tập dữ liệu JMLR.

Với thuật toán GoKrimp, mức độ đáng quan tâm dùng trong kiểm tra dấu được đặt đến 0.01 và số cặp ít nhất để hoàn thành kiểm tra dấu là 25. Với thuật toán SeqKrimp, độ hỗ trợ nhỏ nhất (minsup) là 0.1 tại điểm mà 20 mẫu trên cùng lấy được của mỗi thuật toán không thay đổi khi độ hỗ trợ nhỏ nhất được định nhỏ hơn. Bảng 4.1 biểu thị 20 mẫu trên cùng có từ tập dữ liệu JMLR được trích ra bởi thuật toán SeqKrimp, GoKrimp,SQS và pGOKRIMP.

Thuật toán pGoKrimp, là thế hệ trước của GoKrimp trả về nhiều mẫu không đáng quan tâm là do sự phối hợp của những sự kiện phổ biến. Có sự khác nhau trên có thể là do pGOKRIMP dùng cách mã hóa không trừng phạt khoảng trống và nó không xét đến tần số một mẫu khi gán từ mã đến mẫu.

4.1.2. Bộ dữ liệu Parallel

Parallel là bộ dữ liệu tự tạo bắt chước những trạng thái tiêu biểu trong thực tế nơi dòng dữ liệu được sinh ra bởi năm tiến trình song song độc lập. Mỗi tiến trình P_i phát sinh một sự kiện từ tập các sự kiện { A_i, B_i, C_i, D_i, E_i } theo thứ tự đó. Trong mỗi bước, việc phát sinh chọn một trong năm tiến trình giống nhau một cách ngẫu nhiên và phát sinh một sự kiện bởi dùng các tiến trình nầy cho tới khi chiều dài dòng dữ liệu là 1.000.000.

Với tập dữ liệu nầy, chúng ta có một nền thực sự từ đó tất cả những pha trộn các sự kiện từ những tiến trình song song khác nhau thì không là một mẫu tốt.

Ta nhận được 10 mẫu đầu tiên trích ra từ mỗi thuật toán và tính toán độ chính xác và truy hồi tại K. Độ chính xác tại K được tính là tỉ số của số mẫu thích hợp trong K mẫu đầu tiên được chọn bởi mỗi thuật toán. Trong khi độ truy hồi được đo lường là tỉ số những kiểu của mẫu thực trong K mẫu đầu tiên được chọn bởi mỗi thuật toán. Ví dụ, nếu tập hợp của 10 mẫu đầu tiên chỉ chứa những sự kiện từ tập {A_i, B_i, C_i, D_i, E_i} với một i cho trước thì độ chính xác tại K=10 là 100% trong khi độ truy hồi tại K=10 là 20%. Độ chính xác đo lường tính chính xác của tập những mẫu và độ truy hồi đo lường tính đa dạng của tập những mẫu.

Với bộ dữ liệu nầy thì thuật toán BIDE không thể chấm dứt chạy sau một tuần ngay cả nếu minsup = 1.0. Nguyên do là tất cả những tổ hợp có thể có của 25 sự kiện là những mẫu phổ biến. Vì thế kết quả của thuật toán BIDE và SeqKrimp bị lỗi. Hình 4.1 chỉ ra độ chính xác và độ truy hồi của ba thuật toán SQS, SeqKrimp và GoKrimp khi K thay đổi.

Trong điều kiện của độ chính xác thì tất cả những thuật toán đều tốt bởi vì những mẫu trên cùng được chọn tất cả thỏa độ chính xác. Tuy thế, theo điều kiện của truy hồi thì thuật toán SQS xấu hơn hai thuật toán kia. Có thể giải thích là thuật toán SQS dùng cách mã hóa mà không chấp nhận mã hóa những mẫu đan xen. Với

bộ dữ liệu đặc biệt nầy nơi mà những mẫu đan xen được quan sát thường xuyên thì SQS lại thiếu vắng.

4.2. THỜI GIAN THỰC THI

Thực hiện những thí nghiệm để so sánh thời gian thực thi của những thuật toán khác nhau. Với thuật toán SeqKrimp và BIDE, ta trước tiên định tham số độ hỗ trợ nhỏ nhất đến giá trị nhỏ nhất được dùng trong thí nghiệm nơi mà những mẫu được dùng như là đặc trưng của sự phân lớp.

Thuật toán SQS thì tham số tự do trong khi GoKrimp dùng tham số chuẩn như đã giới thiệu với kiểm tra dấu vì thế thời gian chạy của những thuật toán nầy chỉ phụ thuộc trên kích cỡ dữ liệu. Kết quả của thí nghiệm được minh họa trong bảng 4.3.

Bảng 4.3 Thời gian thực thi và số mẫu trích[4]

	Thời gian thực thi (giây)					**Số mẫu**			
Bộ dữ liệu	Bide	Seqkrimp	SQS	Gokrimp	Bộ dữ liệu	Bide	Seqkrimp	SQS	Gokrimp
Auslan2	0.85	1.0	1.0	0.40	Auslan2	128	4	13	4
Aslbu	74.3	972	277	28	Aslbu	14620	52	195	67
Aslgt	73.7	1344	58501	1842	Aslgt	3472	56	1095	68
Poineer	11.4	65	15	9	Poineer	5475	21	143	49
Skating	67.3	183	123	85	Skating	3767	24	140	49
Context	309	402	86	44	Context	6760	15	138	33
Unix	1055	47111	84869	1824	Unix	28477	75	1070	165
Jmlr	10	232	890	93	Jmlr	4240	23	580	30
parallel	U/N	U/N	2066	342	parallel	U/N	U/N	17	23

Qua kết quả trên thấy rằng SeqKrimp luôn luôn chậm hơn BIDE bởi vì nó cần một thủ tục phụ để chọn các mẫu nén từ tập các ứng viên được trả về bởi thuật toán BIDE.

Thuật toán GoKrimp xếp thứ 1 hoặc 2 theo độ nhanh hơn SeqKrimp hay BIDE. Thuật toán SQS thì rất nhanh trên tập dữ liệu nhỏ mặc dù vẫn còn chậm hơn GoKrimp, tuy thế vẫn nhiều lần chậm hơn những thuật toán khác trên tập dữ liệu lớn như Unix, aslgt. Bảng 4.3 cũng báo số các mẫu được trả về bởi mỗi thuật toán. Thuật toán BIDE luôn trả về nhiều mẫu phụ thuộc vào tham số độ hỗ trợ nhỏ nhất. Khi tham số được đặt thấp thì số mẫu trả về có khi lớn hơn kích cỡ của dữ liệu. Nói một cách khác, SeqKrimp, GoKrimp và SQS trả về một vài mẫu thích đáng. Và tổng số những mẫu trả về dường như chỉ phụ thuộc trên kích cỡ dữ liệu.

4.3. ĐỘ CHÍNH XÁC PHÂN LỚP

Phân lớp là một trong những áp dụng quan trọng nhất của những thuật toán khai thác mẫu. Trong phần nầy, các tác giả thảo luận kết quả của việc dùng những mẫu trích ra cùng với tất cả những từ đơn, như những thuộc tính nhị phân cho thao tác phân lớp. Đưa ra một cách tiếp cận dùng những từ đơn như là đặc trưng của những từ đơn. Thuật toán nầy cùng với BIDE được xem như là một vạch biên ngang trong việc so sánh.

Tất cả tham số được đặt ở giá trị mặc định. Những kết quả của sự phân lớp thu được bởi trung bình cộng độ chính xác phân lớp trên 10 phân đoạn xác nhận chéo. Trong thực nghiệm, có hai tham số quan trọng: giá trị độ hỗ trợ nhỏ nhất cho BIDE và SeqKrimp và thuật toán phân lớp dùng để xây dựng những sự phân loại.

Vì thế, ta thực hiện hai thực nghiệm khác nhau để lượng giá tiếp cận đề xuất khi những tham số nầy thay đổi. Trong thực nghiệm thứ nhất, những độ hỗ trợ nhỏ nhất được định tới những giá trị nhỏ nhất và được trình bày trong bảng 4.4. Trước tiên, tham số K được cho là không giới hạn nhận nhiều mẫu khi có thể. Trong cách làm đó, chúng ta nhận được tập những mẫu với kích cỡ khác nhau và những mẫu

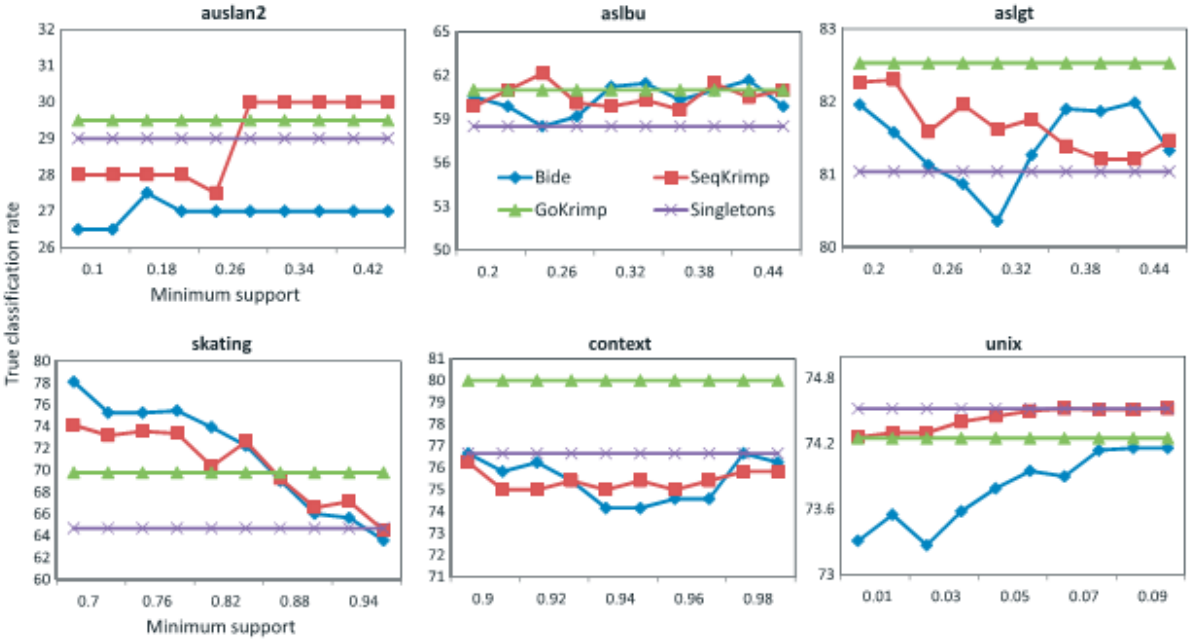
nầy được xếp thứ tự giảm theo thứ hạng định nghĩa bởi mỗi thuật toán. Để tạo sự so sánh cho đủ công bằng, những mẫu tại điểm cuối của tập mẫu được xóa để cùng số lượng mẫu và bằng số mẫu nhỏ nhất có được của các thuật toán. Hơn thế nữa những lớp khác nhau được dùng để lượng giá độ chính xác khi phân lớp. Điều nầy giúp ta chọn phân loại tốt nhất cho thực nghiệm tiếp theo. Bảng 4.4 biểu thị những kết quả của thực nghiệm thứ nhất. Tám cách phân lớp phổ biến thường dùng được chọn cho sự phân lớp. Giá trị số trong mỗi ô chỉ ra phần trăm độ chính xác phân lớp cho mỗi trường hợp. Cột cuối cùng của bảng nầy tóm tắt kết quả tốt nhất, nghĩa là số lớn nhất của mỗi dòng. Hơn nữa, với mỗi ô của cột nầy, giá trị cao nhất theo kết quả phân lớp tốt nhất trong tập dữ liệu cũng được tô sáng lên. Những số sáng màu trong cột cuối cùng là số những mẫu trên cùng được trả về bởi SeqKrimp và GoKrimp thì dễ dự đoán hơn những mẫu trả về bởi BIDE. Trên mỗi bộ dữ liệu thì SeqKrimp và GoKrimp đạt được kết quả tốt nhất. Ngoài ra, những số sáng trên mỗi dòng chỉ ra rằng *phân lớp linear support vector machine (SVM)* là tiếp cận phân lớp tốt nhất cho kiểu dữ liệu nầy nó cho kết quả tốt nhất trong mọi trường hợp.

Bảng 4.4 Phân lớp[4]

Dữ liệu	Algorithm	Naïve Bayes	Random Forest	I48	VFI	Linear SVM	RBF SVM	Kstar	IB1	Best
Auslan2	BIDE	22.50	29.00	25.50	22.50	26.50	23.50	25.50	25.50	29.00
	SEQKRIMP	22.00	30.50	26.50	24.50	28.50	22.50	24.00	27.00	30.50
	GOKRIMP	20.50	29.00	26.00	24.00	29.50	23.50	26.00	26.00	29.50
	SINGLETONS	22.00	29.00	27.00	23.50	29.00	22.00	25.00	26.00	29.00
Aslbu	BIDE	48.07	58.27	50.56	31.06	59.18	50.34	59.41	59.18	81.82
	SEQKRIMP	52.15	60.31	51.02	26.98	59.86	52.07	59.18	57.59	82.27
	GOKRIMP	52.38	54.87	50.34	24.26	59.86	53.28	59.18	58.27	81.90
	SINGLETONS	51.24	54.89	50.79	25.17	58.50	51.24	59.64	57.14	81.04
Pioneer	BIDE	96.87	95.62	94.37	93.75	99.37	95.62	98.12	98.75	99.37
	SEQKRIMP	100.0	98.75	99.37	93.12	100.0	100.0	90.37	93.37	100.0
	GOKRIMP	100.0	99.37	99.39	95.12	100.0	100.0	99.37	99.37	100.0
	SINGLETONS	100.0	96.67	99.37	95.12	100.0	100.0	98.75	99.37	100.0

Skating	BIDE	60.75	57.73	54.33	50.37	63.77	57.33	48.49	47.6	63.77
	SEQKRIMP	73.58	73.58	72.45	66.03	74.15	74.33	64.52	61.69	74.33
	GOKRIMP	67.54	59.81	62.45	57.92	67.54	66.98	53.58	52.64	67.54
	SINGLETONS	61.88	58.67	55.09	51.69	64.71	58.67	49.24	61.25	64.71
Context	BIDE	77.50	70.83	75.00	71.25	74.56	70.41	70.41	61.66	77.50
	SEQKRIMP	79.58	72.91	77.91	74.58	76.25	73.75	72.08	65.00	79.58
	GOKRIMP	80.83	75.41	80.00	77.91	82.08	78.75	74.58	72.18	82.08
	SINGLETONS	78.75	68.33	71.40	74.16	76.66	74.16	67.50	61.25	78.75
Unix	BIDE	42.15	72.15	71.25	29.08	74.05	44.43	67.71	63.36	74.05
	SEQKRIMP	54.80	73.81	72.09	37.48	74.26	45.90	70.25	65.85	74.26
	GOKRIMP	54.09	73.88	72.05	37.70	74.33	45.87	70.39	65.87	74.52
	SINGLETONS	57.77	73.90	72.05	38.06	74.52	44.43	70.77	66.35	74.52

Những kết quả phân lớp với những mẫu được dùng như những thuộc tính nhị phân. Số những mẫu dùng trong mỗi thuật toán thì cân bằng.



Hình 4.1 Kết quả phân lớp[4]

Kết quả phân lớp với tuyến SVM khi dùng đủ những mẫu và minsup thay đổi

Trong thực nghiệm kế tiếp, tham số minsup được biến đổi để thấy kết quả của sự phân lớp biến đổi như thế nào. Bởi vì tuyến phân lớp SVM cho những kết

quả tốt nhất trong hầu hết những bộ dữ liệu, chúng ta chọn sự phân lớp nầy cho thực nghiệm. Hình 4.1 chỉ ra những kết quả. Vì nét đặc biệt của GoKrimp và Singletons không phụ thuộc vào cài đặt minsup, kết quả của những thuật toán nầy không thay đổi dù minsup khác nhau do đó biểu diễn là đường thẳng.

Kết quả chỉ ra rằng, trong hầu hết các bộ dữ liệu, thêm mẫu vào tập các từ đơn thì tốt cho kết quả của phân lớp. Tuy thế, hiệu quả của việc thêm nhiều mẫu thì rất nhạy cảm với minsup. Đặc biệt nhất là nó thay đổi đáng kể từ tập dữ liệu nầy với tập dữ liệu khác. Cách chạy của BIDE có đặc thù là không ổn định, Ví dụ trong bộ dữ liệu aslgt và skating thêm mẫu, nghĩa là thấp minsup, thực tế chứng minh kết quả phân lớp của BIDE. Hơn nữa trong bộ dữ liệu auslan2, aslbu, context và Unix thì hiệu quả của thêm mẫu tối nghĩa. Cách chạy của thuật toán SeqKrimp cũng rất không ổn định khi nó dùng những mẫu trích ra bởi BIDE như là các mẫu ứng viên. Vì thế cần phải điều chỉnh tham số. Mặc khác, kết quả phân lớp của GoKrimp không phụ thuộc vào minsup, nó tốt hơn tiếp cận Singleton trong hầu hết các trường hợp. Nó cũng tốt hơn BIDE trong những bộ dữ liệu dày đặc như là context, aslgt và unix.

4.4. TÍNH NÉN

Việc tính toán hiệu quả nén của tập những mẫu trả về bởi mỗi thuật toán. Để so sánh công bằng, tập tất cả những mẫu phải cùng kích cỡ và bằng số những mẫu nhỏ nhất được trả về của tất cả các thuật toán. Với thuật toán SeqKrimp và GoKrimp những hiệu quả nén được tính toán như là tổng hiệu quả nén trả về sau mỗi bước tham lam. Với những mẫu đóng, hiệu quả nén được tính toán theo thủ tục mã hóa tham lam dùng trong thuật toán SeqKrimp. Với SeqKrimp và BIDE, độ hỗ trợ nhỏ nhất được đặt tại giá trị nhỏ nhất theo thực nghiệm. Hiệu quả nén đo lường như là số các bits được lưu khi mã hóa dữ liệu nguyên thủy dùng tập những mẫu trong từ điển. Vì thuật toán SQS dùng mã hóa khác cho tập dữ liệu trước khi nén nên không thể so sánh tính nén được của thuật toán nầy. Hình 4.3 chỉ ra những kết quả nhận được trong tám bộ dữ liệu khác nhau (kết quả của thuật toán trong bộ dữ

liệu parallel được bỏ qua bởi vì cả SeqKrimp và BIDE cấp độ với kích cỡ của tập dữ liệu nầy). Như kỳ vọng, trong hầu hết những bộ dữ liệu, SeqKrimp và GoKrimp có thể tìm những mẫu nén tốt hơn BIDE. Đặc biệt, trong hầu hết những bộ dữ liệu lớn như là aslgt, aslbu, unix, context và skating thì sự khác nhau giữa SeqKrimp, GoKrimp và BIDE rất đáng kể. Thuật toán GoKrimp có thể tìm những mẫu nén với chất lượng tương đương như là SeqKrimp trong hầu hết các bộ dữ liệu thậm chí tốt hơn trong vài trường hợp như là pioneer, skating và context.

Hình 4.2 Hiệu quả nén[4]

Hiệu quả nén (số bit) khi những mẫu trên cùng được chọn bởi mỗi thuật toán để nén dữ liệu.

Sau cùng ta thực hiện một thực nghiệm khác để so sánh GoKrimp với SQS dựa trên tỉ lệ nén tính toán bởi sự phân dữ liệu trước khi nén và sau khi nén. Điều nầy quan trọng và nói lên rằng tỉ lệ nén phụ thuộc như thế nào vào kích cỡ của dữ liệu chưa nén và cách chọn mã hóa khoảng trống. Vì thế để có sự so sánh công bằng của tỉ lệ nén ta tính khi dùng cùng kích cỡ của dữ liệu chưa nén đại diện. Tuy nhiên có những vấn đề thực tế khác để so sánh như sau.

Cài đặt hiện thời của SQS dùng ý tưởng mã hóa chiều dài các khoảng trống. Nó tính tần suất của một khoảng trống và một khoảng không trống rồi gán chiều dài mã đến khoảng trống và khoảng không trống bởi tính vật lý của khoảng trống và khoảng không trống. Khi số những khoảng không trống trội hơn, trường hợp thực tế trong thực nghiệm với bộ dữ liệu của ta, một khoảng không trống được gán chiều dài mã là 0. Đây là một ý tưởng bởi vì trong thực tế người ta không thể gán chiều dài một từ mã là 0. Ngược lại, GoKrimp thực tế dùng mã hóa Alias cho những khoảng trống. Vì thế có đưa ra thực hành so sánh hai thuật toán một đưa ra ý tưởng chiều dài mã hóa và cái khác thực tế mã hóa chiều dài cho khoảng trống. Vì thế với GoKrimp, ta tính toán cho ý tưởng chiều dài mã hóa một khoảng trống n là log n, kết quả của trường hợp nầy sẽ được đưa ra là GoKrimp* trong thực nghiệm.

Bảng 4.5 chỉ ra tỉ lệ nén của ba thuật toán trên chín tập dữ liệu. Thuật toán SQS cho thấy tỉ lệ nén tốt hơn cho hầu hết trường hợp ngoại trừ bộ dữ liệu Parallel khi không khoảng trống không phổ biến. Đối với tập dữ liệu nầy ảnh hưởng của ý tưởng chiều dài mã hóa thì không hiệu lực. Bằng cách nào mà thế hệ của GoKrimp với ý tưởng chiều dài mã hóa cho khoảng trống với tỉ lệ nén tốt hơn SQS trong hầu hết các trường hợp. Những kết quả nầy chỉ ra rằng sự biến đổi của việc tính chiều dài từ mã ảnh hưởng đáng kể tỉ lệ nén. Vì thế, sự rõ ràng của những kết quả đối với tỉ lệ nén thì rất khó đối với những trường hợp này.

Bảng 4.5 So sánh tỉ lệ nén[4]

	SQS	GoKrimp	GoKrimp*
Auslan2	1.571	1.428	1.907
Aslbu	1.555	1.123	1.284
Aslgt	1.308	1.156	1.450
Poineer	1.302	1.171	1.243
Skating	1.880	1.629	2.095
Context	2.700	1.706	2.698
Unix	2.230	1.638	1.880
Jmlr	1.039	1.008	1.008
parallel	1.070	1.135	2.042

So sánh tỉ lệ nén của những thuật toán khác nhau

4.5. HIỆU LỰC CỦA SỰ KIỆN LIÊN QUAN

Trong phần này, các tác giả hoàn thành một thực nghiệm để minh chứng kiểm tra phụ thuộc được đề xuất với thuật toán GoKrimp. Kiểm tra phụ thuộc đã được đề xuất tránh hoàn toàn sự lượng giá tất cả những nới rộng có thể cuả một mẫu. Khi kiểm tra thực hiện, kết quả được giữ cho lần sau như thế trong trường hợp xấu nhất số tối đa của các kiểm tra thì hầu như bằng kích cỡ của bảng chữ cái alphabet. Hơn nữa tập hợp những sự kiện có liên quan đến một sự kiện cho trước thì hoàn toàn nhỏ hơn kích cỡ của bảng chữ cái vì thế kiểm tra phụ thuộc cũng giúp cho thu nhỏ số lần lượng giá phần nới rộng.

Bảng 4.6 Tỉ lệ nén với kiểm tra dấu[4]

	GoKrimp với Sign test			GoKrimp không Sign test		
	Thời Gian	Tỉ lệ nén	Số mẫu	Thời Gian	Tỉ lệ nén	Số mẫu
Auslan2	0.40	1.428	4	1	1.420	3
Aslbu	28	1.123	67	7414	1.169	117
Aslgt	1842	1.156	68	10293	1.158	79
Poineer	9	1.171	49	822	1.214	88
Skating	85	1.629	49	384	1.662	59
Context	44	1.706	33	251	1.802	33
Unix	1824	1.638	165	U/N	U/N	U/N
Jmlr	93	1.008	30	537895	1.018	182
parallel	342	1.135	23	2296	1.135	23

Tỉ lệ nén của những mẫu với thuật toán GoKrimp với có và không có kiểm tra dấu thì hầu như tương đương nhưng có sign test thì GoKrimp hiệu quả hơn nhiều.

Bảng 4.6 chỉ ra thời gian chạy của thuật toán GoKrimp với có và không có kiểm tra phụ thuộc. Quan sát thấy rằng thuật toán GoKrimp có nhiều hiệu quả hơn khi kiểm tra phụ thuộc được dùng. Quan trọng hơn, tỉ lệ nén hầu như tương đương trong cả hai trường hợp. Bởi vậy kiểm tra phụ thuộc giúp đẩy nhanh tốc độ GoKrimp một cách đáng kể trong khi vẫn bảo quản chất lượng của tập mẫu trong tất cả những bộ dữ liệu. Kết quả nầy phù hợp với trực giác rằng dùng mẫu với những sự kiện không liên quan cho việc nén thì sẽ không cho tỉ lệ nén tốt.

4.6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Các mô hình khai thác chuỗi tuần tự nén thường được thực hiện trên các cơ sở dữ liệu khác nhau trên thế. Tuy nhiên, hầu hết các cơ sở dữ liệu thế giới thực cập nhật với tiến bộ của thời gian. Qua nghiên cứu một số thuật toán khai thác mẫu tuần tự nén SeqKrimp và GoKrimp ta nhận thấy, thuật toán GoKrimp đã đạt được nhiều cải tiến về mặt thời gian thực hiện các mẫu nén và các mẫu nén được trích ra tỏ ra hữu dụng hơn, giảm thiểu khả năng trùng lắp của các mẫu kết quả. Tác giả luận văn

cũng đã hiểu và xây dựng được các ví dụ cụ thể cho từng thuật toán nhằm giúp người đọc dễ tiếp cận các thuật toán hơn.

Từ những nghiên cứu bước đầu này, trong thời gian tới tác giả luận án sẽ tiếp tục nghiên cứu các thuật toán khai thác mẫu tuần tự nén mới hơn để từ đó có thể cải tiến và xây dựng cho mình một thuật toán khai thác mẫu tuần tự tối ưu về mặt thời gian và kết quả nén.

TÀI LIỆU THAM KHẢO

1. D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos, "*Fully automatic cross-associations"*, KDD, 2004, 79–88. Statistical Analysis and Data Mining DOI:10.1002/sam 52 Statistical Analysis and Data Mining, Vol. 7 (2014)
2. D. Fradkin and F. Moerchen, Margin-Closed "*Frequent Sequential Pattern Mining"*, Workshop on Mining Useful Patterns, KDD, 2010.
3. D. Huffman, "*A method for the construction of minimumredundancy codes"*, Proc IRE 40(9) (1952), 1098–1102.
4. H. T. Lam, F. Moerchen, D. Fradkin, and T. Calders , "*Mining Compressing Sequential Patterns*", Statistical Analysis and Data Mining: The ASA Data Science Journal, Volume 7, Issue 1, pages 34–52, February 2014.
5. H. T. Lam, F. Moerchen, D. Fradkin, and T. Calders, "*Mining Compressing Sequential Patterns* ", SDM, SIAM, Philadelphia, PA, USA, 2012.
6. J. Vreeken and N. Tatti, "*The Long and the Short of It: Summarizing Event Sequences with Serial Episodes"*, SIGKDD, ACM, 2012, 462–470.
7. J. Vreeken, M. van Leeuwen, and A. Siebes, "*A. Krimp mining itemsets that compress"*, Data Mining Knowl Discov 23(1) (2011), 169–214.
8. K. Smets and J. V. Slim, "*Directly Mining Descriptive Patterns"*, SIAM SDM, 2012, 236–247.
9. N. Castro and P. Azevedo, "*Time Series Motifs Statistical Significance"*, SDM, 2011, 687–698
10. N. Tatti and J. Vreeken, "*Finding good itemsets by packing data"*, ICDM (2008), 588–597.
11. J. Vreeken and N. Tatti, "*The Long and the Short of It: Summarizing Event Sequences with Serial Episodes*", SIGKDD, ACM, 2012, 462–470.
12. H. T. Lam, F. Moerchen, D. Fradkin, and T. Calders, "*Mining Compressing Sequential Patterns"*, SDM, SIAM, Philadelphia, PA, USA, 2012.

13. C. Faloutsos and V. Megalooikonomou, *"On data mining, compression and Kolmogorov complexity"*, Data Mining Knowl Discov 15(1) (2007), 3–20.

14. C. Ambuhl, M. Mastrolilli, and O. Svensson, "*Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut*", SIAM J Comput 40(2) (2011), 567–596.