|  |  |
| --- | --- |
| ĐẠI HỌC QUỐC GIA TP. HCM  **TRƯỜNG ĐẠI HỌC  CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập - Tự do - Hạnh phúc** |

**ĐỀ CƯƠNG ĐỀ TÀI LUẬN VĂN THẠC SĨ (PTII): 15TC**

**1**. **Tên đề tài hoặc hướng NC** (*gồm cả tiếng Việt và tiếng Anh*):

Tiếng Việt: Ứng dụng N-list trong khai thác mẫu tuần tự phổ biến

Tiếng Anh: Sequential pattern mining using N-lists.

**2**. **Ngành và mã ngành**: Ngành khoa học máy tính, mã ngành 06.48.01.01

**3**. **Họ tên học viên**: BÙI VĂN BẰNG, khóa 10 - đợt 1

**Địa chỉ email, điện thoại liên lạc**:[vanbang0208@gmail.com](mailto:vanbang0208@gmail.com) - 0984219010

**Người hướng dẫn**: PGS.TS VÕ ĐÌNH BẢY

**Địa chỉ email, điện thoại liên lạc**:bayvodinh@gmail.com - 0937306858

**4**. **Tổng quan tình hình NC**

Khai thác mẫu tuần tự phổ biến có vai trò quan trọng trong khai thác dữ liệu, bởi vì nó được ứng dụng trong nhiều lĩnh vực như phần tích thông tin giao dịch của khách hàng, web access log, chuỗi DNA, … Xem xét một cơ sở dữ liệu giao dịch của một cửa hàng bán sách, với mỗi đối tượng đại diện cho một khách hàng, những thuộc tính đại diện cho tác tác giả hay tên sách. Cơ sở dữ liệu ghi nhận những quyển sách được mỗi khách hàng mua trong một khoảng thời gian nhất định. Câu hỏi đặt ra là chuỗi những quyển sách nào được khách hàng mua cùng nhau nhiều nhất trong một khoảng thời gian trên.

Một trang web nổi tiếng được nhiều người truy cập. Mỗi đối tượng là một người dùng, thuộc tính là một trang của web. Bài toán là tìm ra chuỗi các trang của web được truy cập phổ biến.

Bài toán khai thác mẫu tuần tự phổ biến được phát biểu như sau:

Với I = { , , …, } là một tập m phần tử khác nhau, được gọi là những mẫu. Một sư kiện là một tập hợp không có thứ tự và khác rỗng của những mẫu (Xem sự kiên chỉ có một phần tử). Một chuỗi là một danh sách có thứ tự của các sự kiện. Một chuỗi S được ký hiệu ( → → ··· → ), với mỗi phần tử là một sự kiện. Số sự kiện trong chuỗi được gọi là kích thước của chuỗi và tổng số mẫu của chuỗi đươc gọi là độ dài của chuỗi. Một chuỗi có độ dài k được ký hiệu là k-sequence.

Một chuỗi S = ( → → ··· → ) được gọi là chuỗi con của chuỗi R = ( → → ··· → ), được ký hiệu S ⊆ R, nếu tồn tại một số nguyên → < ··· < sao cho ⊆ với tất cả .

Cho một cơ sở dữ liệu tuần tự D, với mỗi chuỗi có một id duy nhất, cho một chuỗi S = ( → → ··· → ), với độ phổ biến của S trong D được tính bằng số lần chuỗi D xuất hiên S. Với một ngưỡng phổ biến cho trước (ký hiệu là minsup), một chuỗi được gọi là tập phổ biến nếu nó có độ phổ biến thỏa mãn ngưỡng minsup.

Cho một cơ sở dữ liệu tuần tự D, và ngưỡng minsup, bài toán là tìm tất cả những chuỗi tuần tự phổ biến trong D.

Bài toán khai thác mẫu tuần tự phổ biến được giới thiệu lần đầu bởi Agrawal and Srinkant vào năm 1995 [2]. Và từ đó đã có nhiều cách tiếp cận và thuật toán được đưa ra để giải quyết.

Các thuật toán để khai thác mẫu tuần tự phổ biến có thể đưa vào hai lớp: lớp Apriopri [2] và lớp FP-growth [4]. Các thuật toán ApropriAll [2], GSP [3], SPADE [5] đều thuộc nhóm Apriopri, cách tiếp cận này sinh ra tập ứng viên và kiểm tra độ phổ biến, yếu điểm của cách tiếp cận này là độ phức tạp của nó tăng theo thời gian và không gian lưu trữ. Cách tiếp cận thuộc lớp FP-growth như: PrefixSpan [6], FUSP [7], “Tree Based Sequential Pattern Mining” [8] không sinh ra tập ứng viên, nhưng cấu trúc cây phức tạp.

Cùng với việc đưa ra bài toán, Agrawal and Srinkant cũng đề xuất ba thuật toán để giả quyết vấn đề trên. Trong đó thuật toán AprioriAll [2] thực thi nhanh hơn hai thuật toán còn lại. AprioriAll gồm ba phần, đầu tiên là tìm tất cả các mẫu tuần tự phổ biến, loại các mẫu tuần tự không phổ biến, rồi khai thác mẫu tuần tự phổ biến. Vấn đề với cách tiếp cận này, tốn quá nhiều chi phí cho việc chuyển dữ liệu trong quá trình khai thác tập phổ biến, và yêu cầu ổ đĩa gần như gấp đôi với kích thước cơ sở dữ liệu, điều này không thực tế với cơ sở dữ liệu lớn.

Thuật toán GSP của Agrawal and Srinkant 1996 [3], sau khi các mẫu đơn được đếm, từ những mẫu phổ biến đơn sinh ra mẫu ứng viên hai phần tử, sau đó tính độ phổ biến của những ứng viên để tìm ra mẫu phổ biến hai phần tử, từ đó sinh ra mẫu ứng viên ba phần tử. Qúa trình này được lập lại cho đến khi không tìm ra mẫu phổ biến nào nữa. Thuật toán này cần yêu cầu quá nhiều lần quét toàn bộ cơ sở dữ liệu, mà việc này tốn rất nhiều thời gian khi cơ sở dữ liệu lớn dần.

Thuật toán SPADE [3] gồm việc xác định chuỗi phổ biến một phần tử, chuỗi phổ biến hai phần tử, sau đó chuyển vào dạng dữ liệu theo chiều dọc và xác định những chuỗi phổ biến khác bằng BFS hay DFS trong mỗi lớp. Thuật toán SPADE chỉ cần ba lần quét cơ sở dữ liệu để tìm được tất cả mẫu phổ biến. Qua thử nghiệm và thực tế cho thấy ưu việt hơn nhiều so với thuật toán GSP. Nhưng SPADE yêu cầu phải chuyển đổi cơ sở dữ liệu theo chiều ngang thành cơ sở dữ liệu theo chiều dọc điều này làm không gian lưu trữ lớn hơn không gian lưu trữ ban đầu.

PrefixSpan [6] thuật toán này mở rộng từ FP-growth, trong cách tiếp cận PrefixSpan cơ sở dữ liệu tuần tự được chiếu bằng cách sau; (1) quét qua cơ sở dữ liệu tìm tất cả các mẫu phổ biến trong chuỗi. Mỗi mẫu phổ biến là một mẫu tuần tự phổ biến một phần tử. (2) Sau đó chia không gian tìm kiếm, tập hợp tất các chuỗi tuần tự phổ biến có thể chia theo các mẫu phổ biến được tìm thấy ở bước 1. (3) Tìm những chuỗi con bằng cách xây dựng các cơ sở dữ liệu được chiếu tương ứng, quá trình này được đệ quy.

Vào năm 2008, Lin đã đưa ra FUSP-tree [7] và thuật toán để cập nhật cây một cách hiệu quả. Cây FUSP-tree gồm một nút gốc (root) và một tập các cây con như là con của nút gốc. Mỗi nút trong cây con gồm **tên mẫu** thể hiện cho nút chứa mẫu, **count** số chuỗi có đường đi qua nút này, **node-links** liên kết tới nút kế tiếp của cùng mẫu của nhánh tiếp theo. FUSP-tree chứa Header-Table đây là bảng chứa những mẫu phổ biến, độ phổ biến của mẫu, và liên kết tới nút đầu tiên của một mẫu.Việc tạo ra cấu trúc cây này từ cơ sở dữ liệu ban đầu cũng tương tự như việc tao ra cây FP-tree. Nhưng khác nhau là liên kết giữ hai nút trên cây phần làm hai loại là ‘s’ và ‘i’. Liên kết ‘s’ là liên kết giữa hai sự kiện khác trong chuỗi. Liên kết ‘i’ là liên kết giữ nhưng mẫu trong cùng một sự kiện.

Bithi, Ashin Ara, Manira Akhter, and Abu Ahmed Ferdaus (2012) đưa ra “Tree Based Sequential Pattern Mining” [8] phát triển từ FUSP-tree. Sau khi cây FUSP-tree được xây dựng từ cơ sở dữ liệu ban đầu, việc khai thác mẫu phổ biến diễn ra từ cây này. Cây FUSP-tree điều kiện được sinh ra tương ứng với mỗi mẫu tuần tự trong Header-table. Quá trình này đệ quy đến khi tất cả các chuỗi tuần tự phổ biến được sinh ra. Thuật toán này khai thác được tất cả các mẫu tuần tự phô biến không sinh ra mẫu ứng viên, và giảm số lần phải quét toàn bộ cơ sở dữ liệu. Và qua thực nghiệm cho thấy kết quả tốt hơn các thuật toán trước đó như GSP, PrefixSpan.

Để giải quyết bài toán khai mẫu phổ biến từ cơ sở dữ liệu giao dịch [1] có nhiều phương pháp hiệu quả được đề xuất như: FP-growth [4], N-list [9]

Trong những năm gần đây, Deng đề xuất N-list [9], một cấu trúc dữ liệu mới để giải quyết. Thuật toán dựa trên cấu trúc cây PPC-tree. Cấu trúc cây PPC-tree gồm hai đặc điểm chính. Thứ nhất là một nút gốc với một tập nhưng nút con. Thứ hai mỗi nút con gồm năm thành phần: item-name, count, children-list, pre-order, post-order.

Cây PPC-tree khác cây FP-tree ở ba điểm chính là:

* PPC-tree không có header table để duy trì sự kết nối giữ các nút.
* Mỗi nút của cây PPC-tree có thêm thuộc tính là pre-order, post-order.
* FP-tree đươc sử dụng trong toàn bộ quá trình của thuật toán FP-growth trong khi đó PPC-tree chỉ để sinh ra Pre-Post của mỗi nút.

N-list là một tập có thứ tự của những nút sinh ra từ cây PPC-tree. N-list có hai thuộc tính quan trong. Một là N-list của itemset có độ dài k+1 được tạo thành từ việc join N-list của nhưng tập con có độ dài k. Hai là độ phổ biến của một itemset là count của nhưng nút trên N-list. PrePost cho thấy hiệu quả hơn FP-growth nhờ N-list, nhưng cần đánh chỉ số mỗi nút của PPC-tree với các chỉ số pre-order và post-order dẫn đến tiêu hao bộ nhớ.

Qua việc tìm hiểu các thuật toán cho việc giải quyết bài toán khai thác mẫu tuần tự phổ biến và bài toán khai thác mẫu phổ biến. Nhận thấy N-list là cấu trúc dữ liệu giúp nâng cao tốc độ giải quyết bài toán khai thác mẫu phổ biến, nhưng chưa thấy việc kết hợp cấu trúc N-list vào việc giải quyết bài toán khai thác mẫu tuần tự phổ biến. Đây là hướng tiếp cận mới có thể đem lại hiệu quả tốt.

**5**. **Tính khoa học** **và tính mới** **của đề tài**

Ỹ nghĩa khoa học: Bài toán quan trọng, yêu cầu thời gian thực thi.

Tính mới:

Tính ưu việt của N-list

Do N-list khó áp dụng: nêu các khó khăn khi chuyển đổi.

Bài toán khai thác mẫu phổ biến tuần tự là một bài toán quan trọng của việc khai thác dữ liệu tuần tự, nên đã có nhiều thuật toán được đưa ra để giải quyết bài này trong đó thuật toán PrePost dựa trên cây PPC-tree và cấu trúc N-list đã cho thấy hiệu quả cao trong việc tìm mẫu phổ biến trên cơ sở dữ liệu giao dịch. Việc ứng dụng N-list trong thác mẫu phổ biến tuần tự có thể đem lại hiệu quả tốt. Đây cũng là một cách tiếp cận mới cho bài toán tìm mẫu tuần tự phổ biến.

**6.** **Mục tiêu, đối tượng và phạm vi** **NC**

6.1 Mục tiêu nghiên cứu

Mục tiêu tổng quan: Đưa cấu trúc N-list vào phương pháp khai thác mẫu phổ biến tuần tự dựa trên cây FUSP.

Mục tiêu chi tiết:

* Xây dựng nền tảng lý thuyết cho việc kết hợp N-list để khai thác tập phổ biến tuần tự trên cây FUSP.
* Hiện thưc lý thuyết đã chứng minh.
* Kiểm tra kết quả, so sách với các phương pháp khác SPADE, PrefixSpan, FUSP.
* Đưa ra nhận xét, hướng phát triển.

6.2 Đối tượng và pham vi nghiên cứu

ĐỐi tượng dạng chuỗi

Các itemset là item đợn

**7**. **Nội dung, phương pháp** **dự định NC**

+ Tìm hiểu về bài toán khai thác tập tuần tự phổ biến và các thuật toán giải được đề xuất bằng cách đọc các tài liệu liên quan trong và ngoài nước.

+ Cách ứng dụng cấu trúc N-list vào khai thác mẫu phổ biến, cây FUSP với việc khai thác tập tuần tự phổ biến bằng tài liệu, chương trình, mã code, và kiểm thử chương trình trên máy tính cá nhân.

+ Kết hợp cấu trúc N-list và cây FUSP để khai thác tập phổ biến bằng cách làm trên giấy và hiện thực bằng chương trình, kiểm trả tính đúng, thời gian, hiệu suất của chương trình trên cơ sở dữ liệu mẫu.  
+ Hoàn thiện cải tiến chương trình dựa trên kết quả thực nghiệm.

+ Dựa mã nguồn mở [10] cùng với các cơ sở dữ liệu mẫu để lấy các kết quả thực nghiệm của các thuật toán được đề xuất trước như SPADE, PrefixSpan, FUSP.

+ So sánh kết quả thực nghiệm của phương pháp đề xuất với kết quả của các phương pháp trước đây

+ Hướng phát triển tiếp theo.

**8**. **Kế hoạch bố trí thời gian NC**:

Kế hoạch viết bản thảo luận văn

|  |  |
| --- | --- |
| Thời gian | Công việc |
| Tháng 1 | Đọc tài liệu tham khảo, đưa ra nhận xét mặt còn tồn tại và hướng giải quyết. |
| Tháng 2 | Nghiên cứu lý thuyết và đưa ra giải pháp làm nền tảng cho việc hiên thực. |
| Tháng 3 | Hiện thực lý thuyết đã nghiên cứu và chỉnh sửa cho phù hợp thực tế. |
| Tháng 4 | Viết bài báo về kết quả thu được, chỉnh sửa theo nhận xét của thầy hướng dẫn. |
| Tháng 5 | Cải tiến phần hiện thực và lý thuyết sau khi nhận được phản hồi từ tạp chí cho hoàn thiện |
| Tháng 6 | Viết luận văn, thực hiện báo cáo luận văn trước hội đồng |
| Tháng 7 | Thực hiện báo cáo luận văn trước hội đồng |

**9**.**Tài liệu tham khảo**

[1] Agrawal, R., Imielinski, T., & Swami, A. (1993). “Mining association rules between sets of items in large databases”. In SIGMOD’93.

[2] Agrawal R and Srikant R (1995). “Mining Sequential Patterns”. In 11th Intl. Conf. Data Engineering.

[3] Srikant R, Agrawal R (1996). “Mining Sequential Patterns: Generalizations and Performance Improvements”, International Conference on Extending Database Technology.

[4] J. Han, J. Pei and Y. Yin (2000), “Mining Frequent Patterns without Candidate Generation”. The 2000 ACM SIGMOD International Conference on Management of Data.

[5] M. Zaki (2001). “SPADE: An Efficient Algorithm for Mining Frequent Sequences”. Machine Learning.

[6] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu (2001). “PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth”. Proc. 2001 Int’l Conf. Data Eng.

[7] C. W. Lin, T. P. Hong, Wen-Hsiang Lu and Wen-Yang Lin (2008). “An Incremental FUSP-Tree Maintenance Algorithm,” The Eighth International Conference on Intelligent System Design and Application.

[8] Bithi, Ashin Ara, Manira Akhter, and Abu Ahmed Ferdaus (2012). "Tree Based Sequential Pattern Mining." IRACST-International Journal of Computer Science and Information Technology & Security (IJCSITS)

[9] Deng, ZhiHong, ZhongHui Wang, and JiaJian Jiang (2012). "A new algorithm for fast mining frequent itemsets using N-lists." Science China Information Sciences55.9

[10] Web link: An Open-Source Data Mining Library

http://www.philippe-fournierviger.com/spmf/index.php?link=algorithms.php

*TP. HCM, ngày tháng năm 201..*

**NGƯỜI HƯỚNG DẪN HỌC VIÊN KÝ TÊN**

(*Họ tên và chữ ký*) (*Họ tên và chữ ký*)

…………………………………. ………………………………….