BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH



NGUYỄN ĐẮC DZỰ TRÌNH

KHAI THÁC TẬP MỤC HỮU ÍCH CAO TRÊN MÔI TRƯỜNG TÍNH TOÁN SONG SONG

LUẬN ÁN TIẾN SĨ

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

NGUYỄN ĐẮC DZỰ TRÌNH

KHAI THÁC TẬP MỤC HỮU ÍCH CAO TRÊN MÔI TRƯỜNG TÍNH TOÁN SONG SONG

Chuyên ngành: Khoa học Máy tính
Mã số chuyên ngành: 9480101

Phản biện độc lập 1: .....
Phản biện độc lập 2: .....

Phản biện 1: .....
Phản biện 2: .....
Phản biện 3: .....

NGƯỜI HƯỚNG DẪN:
1. PGS. TS. Nguyễn Thị Thuý Loan
2. TS. Phạm Thị Thiết

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ LUẬN ÁN TIẾN SĨ

Họ tên nghiên cứu sinh: Nguyễn Đắc Dzự Trình — MSHV: 20126291

Ngày, tháng, năm sinh: 13/3/1979 — Nơi sinh: TP. Hồ Chí Minh

Ngành: Khoa học máy tính — Mã ngành: 9480101

I. TÊN LUẬN ÁN:

Khai thác tập mục hữu ích cao trên môi trường tính toán song song

NHIỆM VỤ VÀ NỘI DUNG:

- Tìm hiểu định nghĩa, mục tiêu và ứng dụng của các tập mục hữu ích cao.
- Tìm hiểu định dạng cơ sở dữ liệu có sự phân cấp các hạng mục.
- Khảo sát các tiếp cận khai thác tập mục hữu ích cao trên cơ sở dữ liệu có phân cấp hạng mục có áp dụng kỹ thuật xử lý song song.
- Đề xuất phương pháp giải quyết bài toán khai thác tập mục hữu ích cao, tập mục hữu ích đa mức từ các cơ sở dữ liệu có sự phân cấp các hạng mục, áp dụng các chiến lược song song hoá.
- Thực nghiệm, đánh giá hiệu năng của phương pháp được đề xuất dựa trên các tiêu chí như thời gian thực hiện, mức độ tiêu thụ bộ nhớ, tính khả mở.

II. NGÀY GIAO NHIỆM VỤ: 13/8/2021

III. NGÀY HOÀN THÀNH NHIỆM VỤ: 01/06/2024

IV. NGƯỜI HƯỚNG DẪN KHOA HỌC:

- PGS. TS. Nguyễn Thị Thuý Loan (Trường Đại học Quốc tế, ĐHQG TP.HCM)
- TS. Phạm Thị Thiết (Trường Đại học Công nghiệp TP.HCM)

Tp. Hồ Chí Minh, ngày ... tháng ... năm 20 ...

NGƯỜI HƯỚNG DẪN	**CHỦ NHIỆM BỘ MÔN ĐÀO TẠO**
(Họ tên và chữ ký)	*(Họ tên và chữ ký)*

TRƯỞNG KHOA/VIỆN……

(Họ tên và chữ ký)

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của bản thân tôi. Các kết quả nghiên cứu và các kết luận trong luận án là trung thực, không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

Tác giả luận án

NCS. Nguyễn Đắc Dzự Trình

TÓM TẮT LUẬN ÁN TIẾN SĨ

Khai thác tập mục hữu ích cao (HUIM) là bài toán tìm kiếm các tập mục mang lại độ hữu ích thỏa một ngưỡng cho trước. Mỗi hạng mục trong giao dịch có một giá trị biểu thị số lượng. Chúng còn có một giá trị biểu thị đơn giá trong toàn bộ CSDL. Tích của hai đại lượng này biểu thị tầm quan trọng, trọng số, hay nói chung là độ hữu ích của hạng mục tương ứng. Nhiều kỹ thuật và giải thuật đã được giới thiệu để tối ưu hoá không gian tìm kiếm dựa trên các chiến lược tỉa ứng viên, giảm chi phí quét CSDL, tăng tốc quá trình tính độ hữu ích, v.v… Ngày nay, các vi xử lý đa nhân dần phổ biến với chi phí phù hợp với người dùng. Các tác vụ tính toán cũng vì thế được mở rộng để tận dụng sức mạnh tính toán này nhằm cải thiện hiệu năng. Tuy nhiên, đối với bài toán khai thác dữ liệu, các tiếp cận kết hợp mô hình xử lý song song đa nhân hiện này vẫn còn rất hạn chế. Vì vậy, luận án nghiên cứu và đề xuất các đóng góp chính sau:

Đóng góp thứ nhất: đề xuất mô hình cho phép khai thác hiệu quả các tập mục hữu ích cao đóng từ CSDL có chứa các hạng mục với độ hữu ích động trong quá trình khai thác nhằm phản ánh sát hơn nữa các CSDL trong thực tế. Đóng góp này áp dụng mô hình độ hữu ích động kết hợp với phương pháp để giảm chi phí quét CSDL nhằm cải thiện hiệu năng của quá trình khai thác tập mục hữu ích cao đóng. Chương 2 trình bày nghiên cứu và mô hình đề xuất thông qua giải thuật iEFIM-Closed.

Đóng góp thứ hai: mở rộng bài toán để áp dụng với dạng CSDL có sự phân cấp các hạng mục dựa trên các nghiên cứu từ *Đóng góp thứ nhất*, và đề xuất mô hình xử lý song song đa nhân để giải quyết bài toán này thông qua việc tận dụng năng lực xử lý của các CPU đa nhân để giảm chi phí về mặt thời gian. Nội dung của đóng góp được trình bày tại Chương 3. Thực nghiệm cho thấy mô hình đề xuất có sự cải thiện rõ rệt về thời gian khai thác.

Đóng góp thứ ba: dựa trên các kết quả nghiên cứu từ *Đóng góp thứ 2 và Đóng góp thứ nhất*, đóng góp này triển khai việc áp dụng mô hình xử lý song song trên nhiều giai đoạn khác nhau của quá trình khai thác, triển khai sâu hơn nữa mô hình song song trong quá trình khai thác. Ngoài ra, một chiến lược điều phối cũng được đề xuất

để giảm thời gian chờ giữa các tác vụ song song. Toàn bộ đóng góp thứ ba nhắm đến việc tận dụng triệt để hơn nữa sức mạnh xử lý của các CPU đa nhân với chi phí phù hợp vào bài toán khai thác tập mục hữu ích cao từ định dạng CSDL phân cấp, có độ hữu ích động. Thực nghiệm cho thấy với mô hình được đề xuất trong Đóng góp thứ ba, hiệu năng khai thác của bài toán được nâng lên đáng kể so với tiếp cận không song song, đặc biệt trên các CSDL có kích thước lớn.

ABSTRACT

High Utility Itemset Mining (HUIM) aims to extract itemsets that have utility to satisfy a user-specified threshold. Each item in the transaction is linked with a value denoted by its quantity. In addition, each item has another value denoted by its unit profit in the whole transaction database. The product of the values of an item yields its weight, importance, or utility. Several strategies and algorithms were proposed to enhance the mining performance of this task, focusing on optimizing search space, reducing the cost of database scans, speeding up utility calculations, etc. Nowadays, multi-core processors are widely available at a reasonable price to users. Many computing tasks are also extended using multi-core processors to boost performance. However, approaches that adopt parallelism into the HUIM task still need to be improved. Thus, the main contributions of this dissertation are as follows.

Contribution #1: proposed a model to efficiently extract closed high-utility itemset from databases containing items with dynamic utility values, to reflect real-world databases better. The contribution adopts the dynamic utility framework and a method to significantly reduce the cost of database scans while mining closed high-utility itemsets. Chapter 2 presents the studies of this contribution as well as the proposed model, resulting in the iEFIM-Closed algorithm.

Contribution #2: extend the mining task studied in *Contribution #1* to work with hierarchical databases. A parallel computing model is also proposed to utilize the multicore processor in the mining task, reducing mining time. Chapter 3 presents the studies and the proposed method to solve the task efficiently. Evaluations clearly show that the proposed model boosted mining performance regarding mining time.

Contribution #3: In addition to the results from Contribution #1 and Contribution #2, this contribution extends the parallel computing model into other phases of the mining algorithm, not just the mining phase. A deeper level of parallelism is also applied to the mining phase. A scheduling task is also proposed to minimize the average waiting time among threads. The overall focus of this contribution is to extensively utilize the

computing power provided by the multicore processors in the mining task on dynamic profit hierarchical databases. Experiments on several databases also show that the proposed model and the algorithms perform better than the serial, non-parallel approach. The performance boost is clearly demonstrated on large databases.

LỜI CẢM ƠN

Tôi bắt đầu con đường nghiên cứu và học thuật khi đã là một người trưởng thành, có công việc và một gia đình ổn định. Vì vậy, việc theo đuổi công việc nghiên cứu, đặc biệt là thực hiện luận án tiến sĩ là một thử thách rất lớn. Công việc nghiên cứu lại đòi hỏi sự tập trung và kiên trì trong một quãng thời gian dài. Với các kết quả đạt được trong đề tài nghiên cứu của mình, có phần hỗ trợ rất lớn từ tập thể cán bộ hướng dẫn, cơ sở đào tạo, đồng nghiệp và từ phía gia đình. Là những người đã luôn đồng hành, sát cánh cùng tôi vượt qua các khó khăn, thử thách và trở ngại trong quá trình nghiên cứu, thực hiện luận án để thu được những kết quả nghiên cứu như hiện nay. Qua luận án này, tôi muốn bày tỏ lòng biết ơn đến những người hỗ trợ cho tôi.

Trước hết, tôi xin bày tỏ lòng biết ơn đến PGS. TS. Nguyễn Thị Thúy Loan, cán bộ hướng dẫn chính của luận án. Cô đã luôn theo sát, động viên, truyền đạt các kiến thức chuyên môn trong quá trình thực hiện luận án, cũng như trong quá trình thực hiện các nghiên cứu của luận án này. Tôi cũng xin gửi lời cảm ơn chân thành đến TS. Phạm Thị Thiết, cán bộ đồng hướng dẫn luận án. Cô luôn đôn đốc, nhắc nhở và hướng dẫn các kiến thức hữu ích trong quá trình nghiên cứu cũng như trong quá trình thực hiện luận án. Tôi cũng muốn bày tỏ lòng biết ơn đến Nhóm Seminar Bảy-Loan, đã cố vấn và đưa ra nhiều góp ý quan trọng liên quan đến định hướng học thuật cũng như quá trình thực hiện luận án.

Kế đến, tôi cũng xin gửi lời cảm ơn đến tập thể cán bộ, giảng viên khoa Công nghệ thông tin, Trường Đại học Công nghiệp TP.HCM nói riêng và Trường Đại học Công nghiệp TP.HCM nói chung, đặc biệt là TS. Lê Nhật Duy, đã luôn tận tâm hỗ trợ, đóng góp các ý kiến thiết thực và quan trọng, tạo mọi điều kiện thuận lợi để tôi có thể hoàn thành luận án này.

Bên cạnh đó, tôi xin cảm ơn tập thể cán bộ Viện đào tào quốc tế và Sau đại học đã hỗ trợ và tạo mọi điều kiện để tôi có thể hoàn thành được quá trình học tập và nghiên cứu tại Trường Đại học Công nghiệp TP.HCM.

Ngoài ra, tôi cũng muốn gửi lời cảm ơn đến các đồng nghiệp trong nhóm nghiên cứu, cũng như tại Khoa Công nghệ thông tin thuộc Trường đại học HUTECH đã chia sẻ các buồn vui, kinh nghiệm và hỗ trợ tôi trong suốt hành trình tham gia nghiên cứu khoa học và thực hiện luận án này.

Cuối cùng, tôi cũng muốn gửi đến gia đình yêu quý của mình lời cảm ơn sâu sắc. Mặc dù công việc nghiên cứu và thực hiện luận án của tôi chiếm phần lớn thời gian mỗi ngày, gia đình vẫn luôn là chỗ dựa vững chắc để tôi có thêm động lực để hoàn thành tốt công việc mà mình đam mê.

NCS. Nguyễn Đắc Dzự Trình

MỤC LỤC

MỞ ĐẦU 1
1. Đặt vấn đề 1
2. Mục tiêu nghiên cứu 7
3. Phạm vi nghiên cứu 8
4. Đối tượng nghiên cứu 9
5. Cách tiếp cận và phương pháp nghiên cứu 10
6. Các đóng góp của luận án 10
7. Ý nghĩa thực tiễn của luận án 10
8. Cấu trúc luận án 11
CHƯƠNG 1 TỔNG QUAN VỀ LĨNH VỰC NGHIÊN CỨU 13
1.1 Giới thiệu 13
1.2 Các nghiên cứu liên quan 14
1.2.1 Bài toán khai thác tập mục hữu ích cao 15
1.2.2 Bài toán khai thác tập mục hữu ích cao với biểu diễn đóng 22
1.2.3 Bài toán khai thác tập mục hữu ích cao trên CSDL phân cấp 26
1.2.4 Bài toán khai thác song song các tập mục hữu ích cao 29
1.2.5 Các bài toán khai thác tập mục hữu ích cao khác 31
CHƯƠNG 2 KHAI THÁC HIỆU QUẢ TẬP MỤC HỮU ÍCH CAO ĐÓNG TỪ CƠ SỞ DỮ LIỆU CÓ ĐỘ HỮU ÍCH BIẾN ĐỘNG 33
2.1 Giới thiệu bài toán 33
2.2 Một số định nghĩa cơ sở 36
2.3 Cải thiện hiệu năng quét CSDL 45
2.4 Giải thuật iEFIM-Closed 46
2.5 Đánh giá mức độ hiệu quả của P-set 48
2.6 Đánh giá độ phức tạp của giải thuật iEFIM-Closed 48
2.7 Thực nghiệm 49
2.7.1 Môi trường thực nghiệm 49
2.7.2 Cơ sở dữ liệu thực nghiệm 49
2.7.3 Phương pháp đánh giá 50
2.7.4 Thời gian thực hiện 51
2.7.5 Mức độ sử dụng bộ nhớ 54
2.7.6 Chi phí quét CSDL 56

2.7.7 Khả năng thích nghi với việc mở rộng CSDL 58
2.8 Kết chương 60
CHƯƠNG 3 MÔ HÌNH SONG SONG HÓA KHAI THÁC TẬP MỤC HỮU ÍCH CAO ĐA MỨC 62
3.1 Giới thiệu 62
3.2 Một số định nghĩa 65
3.3 Mô hình song song hoá quá trình khai thác tập mục hữu ích cao đa mức trên CSDL phân cấp 72
3.4 Khai thác song song các tập mục hữu ích cao đa mức từ CSDL phân cấp 73
3.4.1 Giải thuật MCML-Miner 73
3.4.2 Độ phức tạp của giải thuật MCML-Miner 76
3.4.3 Đánh giá hiệu năng của giải thuật MCML-Miner 76
3.5 Khai thác tập mục hữu ích cao đa mức - đóng từ CSDL phân cấp 80
3.5.1 Kiểm tra nhanh tính đóng của một tập mục 81
3.5.2 Giải thuật MLC-Miner 83
3.5.3 Độ phức tạp của giải thuật MLC-Miner 87
3.5.4 Thực nghiệm 87
3.6 Khai thác song song các tập mục hữu ích cao đa mức - đóng 97
3.6.1 Giải thuật PMLC-Miner 97
3.6.2 Độ phức tạp của giải thuật PMLC-Miner 98
3.6.3 Thực nghiệm 99
3.7 Kết chương 105
CHƯƠNG 4 CẢI THIỆN HIỆU QUẢ MÔ HÌNH KHAI THÁC SONG SONG CÁC TẬP MỤC HỮU ÍCH CAO ĐA MỨC 108
4.1 Giới thiệu 108
4.2 Cải thiện hiệu quả của mô hình khai thác song song tập mục hữu ích cao đa mức 109
4.2.1 Giải thuật MCML+ 109
4.2.2 Giải thuật MCML++ 112
4.3 Thực nghiệm 116
4.3.1 Môi trường và CSDL thực nghiệm 116
4.3.2 Đánh giá về thời gian thực hiện 118
4.3.3 Đánh giá về mức độ sử dụng bộ nhớ 120
4.3.4 Đánh giá về khả năng thích nghi với việc mở rộng CSDL 122

4.3.5 So sánh giữa các chiến lược điều phối 124
4.4 Kết chương 125
CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 127
5.1 Thảo luận 127
5.1.1 So sánh với các nghiên cứu trước đây 127
5.1.2 Phân tích hiệu quả của các giải thuật đề xuất 128
5.1.3 Hạn chế của nghiên cứu 129
5.1.4 Vấn đề phát sinh và cách giải quyết 129
5.1.5 Ý nghĩa của nghiên cứu 130
5.2 Kết luận 130
5.3 Hướng phát triển trong tương lai 132
5.3.1 Nghiên cứu và phát triển chiến lược cân bằng tải 132
5.3.2 Mở rộng mô hình xử lý song song trên môi trường phân tán 133
5.3.3 Tích hợp các phương pháp học sâu vào khai thác dữ liệu 133
5.3.4 Ứng dụng trong các lĩnh vực thực tiễn 133
5.3.5 Tăng cường tính khả mở của giải thuật 134
5.3.6 Phân tích và cải thiện các tiêu chí đánh giá 134
DANH MỤC CÔNG TRÌNH ĐÃ CÔNG BỐ CỦA HỌC VIÊN 135
1. Các công trình là tác giả chính 135
2. Các công trình đồng tác giả 135
TÀI LIỆU THAM KHẢO 136

DANH MỤC HÌNH ẢNH

Hình 1.1. Minh hoạ một cấu trúc taxonomy 26
Hình 2.1. Không gian tìm kiếm cho tập $\mathbb{I} = a, b, c, d, e$ 40
Hình 2.2. So sánh thời gian thực hiện trên các CSDL [CT4] 53
Hình 2.3. So sánh mức độ sử dụng bộ nhớ trên các CSDL [CT4].....55
Hình 2.4. So sánh số lượng giao dịch được quét trên các CSDL [CT4].....57
Hình 2.5. So sánh khả năng mở rộng theo tiêu chí thời gian thực hiện. [CT4] 59
Hình 2.6. So sánh khả năng mở rộng theo tiêu chí bộ nhớ tiêu thụ. [CT4] 60
Hình 3.1. Cấu trúc phân cấp gồm 2 mức trong CSDL tại Bảng 3.1 63
Hình 3.2. Utility-list của các hạng mục tiềm năng tại $minutil = 30$ 67
Hình 3.3. Cấu trúc taxonomy của CSDL $\mathbb{D}$ cho tại Bảng 1.1.....69
Hình 3.4. Không gian tìm kiếm được song song hoá của mô hình khai thác [CT2] 74
Hình 3.5. Thời gian khai thác của MCML-Miner và MLHUI-Miner [CT3].....78
Hình 3.6. Lượng bộ nhớ sử dụng của MCML-Miner và MLHUI-Miner [CT3] 79
Hình 3.7. Hiệu năng của MCML-Miner và MLHUI-Miner trên các CSDL [CT3] .80
Hình 3.8. So sánh thời gian thực hiện trên các CSDL thực nghiệm. [CT1] 93
Hình 3.9. So sánh mức độ sử dụng bộ nhớ trên các CSDL thực nghiệm. [CT1] 94
Hình 3.10. So sánh hiệu quả của EUCP trên các CSDL thực nghiệm. [CT1].....96
Hình 3.11. So sánh thời gian thực hiện của PMLC-Miner. 101
Hình 3.12. So sánh mức độ sử dụng bộ nhớ của PMLC-Miner.....102
Hình 3.13. Hiệu quả của PMLC-Miner so các giải thuật baseline. 104
Hình 4.1. Mô hình song song hoá quá trình tỉa ứng viên [CT2].....111
Hình 4.2. Thời gian thực hiện của các giải thuật [CT2] 118
Hình 4.3. Lượng bộ nhớ sử dụng của các giải thuật [CT2] 121
Hình 4.4. Thời gian thực hiện khi mở rộng CSDL [CT2] 122
Hình 4.5. Lượng bộ nhớ sử dụng khi mở rộng CSDL [CT2] 123
Hình 4.6. Các chiến lược điều phối được sử dụng [CT2].....124

DANH MỤC BẢNG BIỂU

Bảng 1.1. Ví dụ về CSDL giao dịch 14
Bảng 1.2. Độ hữu ích ngoại của các hạng mục cho tại Bảng 1.1.....14
Bảng 1.3. Các giải thuật khai thác tập mục hữu ích cao 20
Bảng 1.4. Các giải thuật khai thác tập mục hữu ích cao đóng 24
Bảng 1.5. Các giải thuật khai thác tập mục hữu ích cao tổng quát 28
Bảng 1.6. Các giải thuật khai thác tập mục hữu ích cao song song, phân tán 30
Bảng 2.1. Ví dụ về CSDL giao dịch có độ hữu ích tĩnh 33
Bảng 2.2. Độ hữu ích ngoại của các hạng mục cho tại Bảng 2.1.....33
Bảng 2.3. CSDL mở rộng để biểu diễn sự thay đổi độ hữu ích của các hạng mục ..35
Bảng 2.4. CSDL độ hữu ích biến đổi thu gọn.....39
Bảng 2.5. Các CSDL thực nghiệm.....50
Bảng 2.6. So sánh số lượng giao dịch cần quét trong quá trình khai thác 55
Bảng 3.1. Ví dụ về CSDL giao dịch giao dịch.....62
Bảng 3.2. Độ hữu ích đơn ngoại của các hạng mục cho tại Bảng 3.1 62
Bảng 3.3. MLHUI tìm được trong CSDL cho ở Bảng 3.1, $minutil = 60$. 64
Bảng 3.4. TWU của các hạng mục đơn từ CSDL ở Bảng 3.1 và Bảng 3.2 66
Bảng 3.5. CSDL sau lần quét thứ 2, với $minutil = 30$. 66
Bảng 3.6. Các CSDL thực nghiệm giải thuật MCML-Miner 77
Bảng 3.7. Các CSDL thực nghiệm giải thuật MLC-Miner.....89
Bảng 4.1. Mức độ song song hoá giữa các giải thuật 115
Bảng 4.2. CSDL thực nghiệm giải thuật MCML+/MCML++.....116
Bảng 5.1. Các giải thuật khai thác đã được đề xuất.....131

DANH MỤC TỪ VIẾT TẮT

BCC	Kỹ thuật kiểm tra bao đóng lùi (Backward Closure Checker)
CHUI	Tập mục hữu ích cao đóng (Closed High Utility Itemset)
CHUIM	Bài toán khai thác tập mục hữu ích cao đóng (CHUI Mining)
CJU	Kỹ thuật kiểm tra nhanh bao đóng (Closure Jumping)
CSDL	Cơ sở dữ liệu
DCP	Tính chất bao đóng giảm (Downward Closure Property)
EUCP	Chiến lược cắt tỉa ước lượng hữu ích đồng xuất hiện (Estimate Utility Co-occurrence Pruning)
EUCS	Cấu trúc ước lượng hữu ích đồng xuất hiện (Estimate Utility Co-occurrence Structure)
FCC	Kỹ thuật kiểm tra bao đóng tiến (Forward Closure Checker)
FCFS	Chiến lược điều phối đến trước, phục vụ trước (First Come, First Served)
FI	Tập mục phổ biến (Frequent Itemset)
FIM	Frequent Itemset Mining (khai thác tập phổ biến)
GWU	Trọng số hữu ích giao dịch tổng quát (Generalized Weighted Utilization)
HARM	Khai thác luật kết hợp hữu ích cao (High Utility Association Rule Mining)
HDP	Kỹ thuật chiếu CSDL hữu ích cao (High utility Database Projection)
HTM	Kỹ thuật ghép giao dịch hữu ích cao (High utility Transaction Merging)
HUI	Tập mục hữu ích cao (High Utility Itemset)

HUIM	Bài toán khai thác tập mục hữu ích cao (High Utility Itemset Mining)
MCML-Miner	Giải thuật khai thác tập mục hữu ích cao đa mức song song (Multi-Core Multi-Level HUI-Miner)
MCML+	Giải thuật MCML-Miner cải tiến mức độ 1.
MCML++	Giải thuật MCML-Miner cải tiến mức độ 2.
MHUI	Tập mục hữu ích cao tối đại (Maximal High Utility Itemset)
minsup	Ngưỡng độ hỗ trợ tối thiểu (Minimum Support Threshold)
minutil	Ngưỡng độ hữu ích tối thiểu (Minimum Utility Threshold)
MLCHUI	Tập mục hữu ích cao đa mức, đóng (Multi-Leve Closed High Utility Itemset)
MLHUI	Tập mục hữu ích cao đa mức (Multi-Level HUI)
PMLC-Miner	Giải thuật MLC-Miner song song (Parallelized MLC-Miner)
RSJF	Chiến lược điều phối SJF nghịch (Reversed SJF)
RUCK	Chiến lược kiểm tra nhanh bao đóng hữu ích (Rapid Utility Closure Checker)
SJF	Chiến lược điều phối Ưu tiên Tác vụ Ngắn nhất trước (Shortest Jobs First)
TU	Độ hữu ích của giao dịch (Transaction Utility)
TWU	Trọng số hữu ích giao dịch (Transaction Weighted Utilization)

MỞ ĐẦU

1. Đặt vấn đề

Bài toán khai thác tập phổ biến (FIM) được Agrawal và đồng sự đề xuất vào năm 1993 [1] nhằm phân tích hành vi khách hàng, phân tích giỏ hàng trong các cửa hàng bán lẻ, các sàn giao dịch thương mại điện tử, v.v… Mỗi khi khách hàng mua sắm tại cửa hàng và thanh toán tại quầy thu ngân, một giao dịch giữa khách hàng và cửa hàng sẽ được phát sinh. Các giao dịch này thường được quản lý bên trong một CSDL giao dịch. Qua thời gian, lượng giao dịch tích lũy trong CSDL sẽ ngày càng lớn.

Mục tiêu của bài toán FIM là thực hiện phân tích các giao dịch giữa khách hàng và cửa hàng bán lẻ trong CSDL để có thể tìm ra các tri thức ẩn chứa bên trong nguồn dữ liệu này. Cụ thể, FIM có thể xác định được những mặt hàng hoặc những kết hợp giữa các mặt hàng nào thường hay được người dùng chọn mua. Các kết hợp này nếu có tần suất xuất hiện thỏa mãn một ngưỡng tối thiểu cho trước, thì được gọi là "phổ biến". Ngưỡng tối thiểu cho trước này được gọi là ngưỡng độ hỗ trợ tối thiểu, và thường được ký hiệu là $minsup$ hoặc σ.

Để không làm mất tính tổng quát của bài toán, các mặt hàng được thống nhất sử dụng thông qua thuật ngữu hạng mục và kết hợp giữa các mặt hàng là tập mục. Kết quả thu được từ bài toán FIM được sử dụng để sinh ra tập các luật kết hợp, giúp cho nhà quản lý có thể nắm được các quy luật kết hợp nào giữa các hạng mục hoặc tập mục có độ tin cậy cao để thúc đẩy việc kinh doanh hiệu quả hơn [1].

Kể từ khi được đề xuất vào năm 1993 [1], FIM dần thu hút được nhiều sự quan tâm từ cộng đồng nghiên cứu về khai thác dữ liệu nói chung và khai thác mẫu nói riêng. Bài toán này nhận hai tham số đầu vào. Tham số đầu vào thứ nhất của bài toán là một CSDL giao dịch như đã nêu, tham số đầu vào thứ hai là ngưỡng $minsup$. Kết quả nhận được của bài toán FIM là tập đầy đủ các tập mục có tần suất xuất hiện không bé hơn ngưỡng $minsup$ đã cho, và được gọi là các tập mục phổ biến. Nhiều tiếp cận,

mô hình, giải thuật đã được đề xuất nhằm hướng đến việc giải quyết hiệu quả bài toán này [2–12].

Nhược điểm lớn nhất của bài toán FIM là chỉ quan tâm đến tần suất xuất hiện của các hạng mục và tập mục trong CSDL đang xét. Các hạng mục này được FIM xem như có tầm quan trọng như nhau. Đối với bài toán FIM, tại mỗi giao dịch, các hạng mục chỉ có thể thuộc một trong hai trạng thái đó là tồn tại hoặc không tồn tại trong giao dịch đó. Do đó, các thông tin hữu ích khác trong CSDL giao dịch không được xem xét. Trong thực tế, các thông tin này cũng mang lại nhiều tri thức quan trọng và hữu ích cho quá trình khai thác dữ liệu.

Một loại thông tin không kém phần quan trọng đối với quá trình khai thác dữ liệu từ CSDL giao dịch, đó là mức độ quan trọng của các hạng mục trong mỗi giao dịch. Xét ví dụ, với một giao dịch được phát sinh khi khách hàng tiến hành mua sắm tại một cửa hàng bán lẻ, giao dịch này thường chứa các thông tin như số lượng mỗi mặt hàng đã được chọn mua, đơn giá, mức thuế tương ứng của chúng, v.v… Các tiếp cận đã đề xuất trong bài toán khai thác tập phổ biến đều bỏ qua các thông tin này. Kết quả là các giải thuật FIM thường tạo ra các tập mục có tần suất xuất hiện lớn (mức độ phổ biến cao). Tuy nhiên, doanh thu mà chúng mang lại là không được xem xét đến.

Một ví dụ cụ thể, tại một sàn giao dịch điện tử có kinh doanh nhiều mặt hàng liên quan đến lĩnh vực Tin học. Trong số các mặt hàng kinh doanh, các sản phẩm mang lại giá trị doanh thu thấp như bàn phím, chuột, thẻ nhớ hoặc các linh kiện khác thường được người dùng chọn mua nhiều; các sản phẩm có giá trị doanh thu cao như hệ thống máy trạm, máy chủ, v.v… lại thường ít được người dùng thông thường chọn mua. Khi ta đưa CSDL giao dịch của sàn giao dịch điện tử này làm đầu vào của vào bài toán FIM, kết quả thu được sẽ không chứa các mặt hàng hoặc sản phẩm có giá trị doanh thu cao mà chỉ chứa các sản phẩm thường được khách hàng lựa chọn mua nhiều. Các mặt hàng này mặc dù thường hay xuất hiện trong các giao dịch của CSDL, nhưng doanh thu của chúng mang lại cho đơn vị kinh doanh là không nhiều.

Có thể nhận thấy, khi áp dụng bài toán FIM vào trong một số ứng dụng thực tế, hạn chế của bài toán này được bộc lộ rõ. Từ đó, một bài toán khác mở rộng dựa trên bài toán FIM đã được giới thiệu. Bài toán này có tên gọi là bài toán khai thác tập mục hữu ích cao (HUIM) [13]. Tương tự như FIM, bài toán HUIM cũng nhận được nhiều sự quan tâm, khảo sát từ các nhà nghiên cứu [14–21]. Nhược điểm quan trọng đã nêu của bài toán FIM được xác định là mục tiêu chính của bài toán HUIM.

Thay vì chỉ là liệt kê danh sách các hạng mục được chọn mua dưới dạng giao dịch, bài toán HUIM mở rộng CSDL giao dịch để chứa thêm các dữ liệu thực tế mà bài toán quan tâm. Khi đó, số lượng của mỗi hạng mục và đơn giá của chúng sẽ được HUIM xem xét. Tích của hai đại lượng này cung cấp một giá trị biểu thị trọng số, doanh thu hoặc tầm quan trọng mà hạng mục đó mang lại. Và để không làm mất tính tổng quát của bài toán HUIM, các khái niệm này ta gọi chung là mức độ hữu ích (utility) của các hạng mục/tập mục trong giao dịch.

Tương tự bài toán FIM, bài toán HUIM cũng xem xét hai tham số đầu vào. Tham số đầu tiên một CSDL giao dịch với đầy đủ các thông tin cần thiết của mỗi hạng mục như số lượng và đơn giá của hạng mục đó. Tham số thứ hai là một ngưỡng biểu thị độ hữu ích tối thiểu ($minutil$) mà các tập mục cần phải thoả để trở thành tập mục hữu ích cao (HUI). Ngoài mục đích chính là áp dụng trong bài toán phân tích hành vi khách hàng, hiện nay HUIM cũng được đưa vào các ứng dụng khác như phục vụ khảo sát thị trường, sinh tin học, phát hiện xâm nhập, v.v…

Bài toán FIM tồn tại một tính chất quan trọng mà có thể tận dụng để giảm chi phí thực hiện giải thuật, đó là tính chất Apriori hay còn gọi là tính chất bao đóng giảm DCP. Tính chất này có thể phát biểu ngắn gọn như sau: "*nếu một tập mục là phổ biến, các tập con của nó cũng là các tập mục phổ biến*" [22]. Các tiếp cận giải quyết bài toán FIM đều áp dụng một cách triệt để tính chất quan trọng này lược bỏ bớt các tập mục không có tiềm năng trở thành tập phổ biến khỏi không gian tìm kiếm của bài toán; giảm chi phí tính toán không cần thiết để xét các tập ứng viên này. Từ đó, cải thiện được hiệu năng của giải thuật khai thác [2–12].

Không như độ hỗ trợ của bài toán FIM, độ đo mức độ hữu ích của các hạng mục/tập mục trong bài toán HUIM không thoả tính chất DCP [22]. Do đó, các chiến lược tỉa ứng viên dựa trên độ đo độ hỗ trợ mà trước đó có thể hoạt động hiệu quả với các tiếp cận FIM thì lại hoàn toàn không khả dụng đối với bài toán HUIM. Điều này làm tăng số lượng ứng và không gian tìm kiếm của bài toán HUIM so với FIM. Đồng thời, làm tăng độ phức tạp của bài toán HUIM so với FIM. Các tiếp cận, giải thuật khai thác HUI vì vậy mà có thời gian khai thác dài, yêu cầu không gian bộ nhớ lớn.

Vì vậy, các nghiên cứu tập trung tìm kiếm một độ đo thỏa tính chất DCP để giúp loại bỏ các tập ứng viên không tiềm năng trong quá trình khai thác, cải thiện thời gian khai thác của bài toán và giảm thiểu mức độ sử dụng bộ nhớ. Trong thực tế, hàng loạt các nghiên cứu đã giới thiệu nhiều giải thuật để giải quyết hiệu quả bài toán này [14–21].

Tương tự bài toán FIM, HUIM cũng tồn tại một đặc điểm là phát sinh một lượng lớn các tập mục kết quả. Do đó, cần nhiều thời gian để phân tích đầy đủ tập kết quả thu được. Ngoài ra, việc phải lưu trữ một lượng lớn các tập mục kết quả sẽ làm tăng không gian bộ nhớ và chi phí tính toán. Điều này dẫn đến việc cần có một biểu diễn thu gọn của các tập mục HUI nhằm giảm thiểu số lượng tập mục, chi phí tính toán, cũng như lượng bộ nhớ cần để lưu trữ chúng.

Để thu gọn tập kết quả của bài toán, có hai dạng biểu diễn chính thường được sử dụng trong các bài toán khai thác mẫu nói chung và bài toán HUIM nói riêng: biểu diễn tập mục tối đại [23–26] và biểu diễn tập mục đóng [18,25,27–31]. Khái niệm các tập mục hữu ích cao đóng (CHUI), tập mục hữu ích cao tối đại (MHUI) cũng được mở rộng từ khái niệm tập mục phổ biến đóng [32] và tối đại [33] trong bài toán FIM.

Ngoài ra, trong thực tế các cửa hàng bán lẻ thường tổ chức các mặt hàng dưới dạng các nhóm mặt hàng hoặc chủng loại mặt hàng. Ví dụ, một cửa hàng bán thiết bị Tin học có kinh doanh nhiều mặt hàng như đĩa cứng, bộ nhớ RAM, card đồ họa, máy in, máy scan, máy tính cá nhân, máy chủ, laptop, v.v… Các mặt hàng này được cửa hàng phân thành các nhóm ở mức tổng quát cao hơn như thiết bị ngoại vi bao gồm máy in,

máy scan; linh kiện, bao gồm đĩa cứng, bộ nhớ RAM, card đồ họa; máy điện toán bao gồm máy tính cá nhân, máy chủ, laptop thuộc nhiều nhãn hiệu khác nhau, v.v… Các nhóm mặt hàng tổng quát này lại có thể được tổng quát hóa ở mức cao hơn nữa.

Các tiếp cận phục vụ cho bài toán HUIM trước đó đều chỉ quan tâm đến các mặt hàng chuyên biệt trong quá trình khai thác mà chưa quan tâm đến nhóm mặt hàng ở các mức tổng quát cao hơn.

Năm 2017, kế thừa từ bài toán khai thác FIM tổng quát của các nghiên cứu trước đó [34–40], bài toán khai thác tập mục hữu ích cao tổng quát từ CSDL có sự phân cấp các hạng mục được đề xuất [41]. Bài toán này mở rộng không gian tìm kiếm thông qua cấu trúc phân cấp các hạng mục để xem xét các mức tổng quát cao hơn, và kết xuất ra các tập mục hữu ích cao có mức độ tổng quát cao hơn nhằm phục vụ tốt hơn cho người sử dụng. Tuy nhiên, với không gian tìm kiếm được mở rộng thông qua cấu trúc phân cấp các hạng mục, thời gian khai thác của bài toán cũng gia tăng và số lượng tập mục kết quả cũng gia tăng theo. Điều này đưa đến việc cần phải có nhiều thời gian để phân tích các kết quả nhận được từ bài toán.

Cho đến thời điểm này, số lượng các nghiên cứu về hướng nghiên cứu này vẫn còn hạn chế, đặc biệt là khai thác tập mục hữu ích cao trên CSDL phân cấp. Lý do chính là vì không gian tìm kiếm của bài toán mở rộng một cách đáng kể, gia tăng độ phức tạp của bài toán. Dẫn đến chi phí tính toán và thời gian khai thác lớn, và mức độ sử dụng bộ nhớ cũng gia tăng theo.

Với sự tiến bộ của công nghệ chế tạo vi xử lý, ngày nay hầu như tất cả máy tính điện tử đều được trang bị các CPU có nhiều hơn một nhân tính toán – còn được gọi là vi xử lý đa nhân. Các CPU này phổ biến ở mức từ 2 đến 12 nhân xử lý vật lý (hoặc từ 4 đến 24 nhân luận lý) với chi phí phù hợp cho nhiều đối tượng người sử dụng. Việc tích hợp nhiều nhân tính toán vào một đế CPU giúp giảm đáng kể thời gian chờ của các tác vụ có thời gian xử lý dài và giúp các ứng dụng thực thi được hiệu quả hơn.

Sự ra đời của các vi xử lý đa nhân cũng là động lực thúc đẩy sự phát triển của các kỹ thuật tính toán, xử lý song song [42–44]. Ngày càng có nhiều các ứng dụng, giải thuật,

mô hình được triển khai để tận dụng khả năng xử lý song song của vi xử lý. Thế nhưng đối với bài toán khai thác dữ liệu, việc áp dụng các mô hình xử lý song song vào quá trình khai thác vẫn còn hạn chế, đặc biệt là bài toán HUIM.

Đối với các kỹ thuật sử dụng mô hình tính toán phân tán để phục vụ bài toán HUIM, các phương pháp này có ưu điểm là khả năng xử lý được các CSDL cực lớn dựa trên các nền tảng như Hadoop, Spark [45–47]. Tuy nhiên, các tiếp cận này thông thường chỉ là các giải thuật xấp xỉ, hoặc chỉ thực hiện công việc phân hoạch CSDL lớn thành các CSDL có kích thước nhỏ hơn và được khai thác thông qua các tiếp cận tuần tự trên mỗi nút tính toán. Ngoài ra, việc sử dụng các hệ thống phân tán cũng đưa đến tốn kém về mặt kinh tế khi phải đầu tư chi phí cho các hệ thống tính toán lớn hoặc chi phí thuê mướn thiết bị trong thời gian dài.

Trong thời gian gần đây, một số các nghiên cứu áp dụng mô hình xử lý song song đa nhân vào bài toán HUIM cũng đã được đề xuất [57], [58]. Tuy nhiên các tiếp cận này vẫn còn các nhược điểm cần phải khắc phục, ví dụ như chiến lược điều phối, khả năng tận dụng tối đa sức tính toán của các vi xử lý đa nhân, v.v…

Với các vấn đề nêu trên, các nội dung nghiên cứu chính của luận án được đặt ra cho quá trình nghiên cứu nhằm tìm ra giải pháp khắc phục, bao gồm:

- **Nội dung 1:** Làm thế nào để tăng hiệu quả cho quá trình khai thác các tập mục hữu ích cao đóng, phục vụ cho bài toán tìm luật kết hợp hữu ích cao mà vẫn bảo toàn lượng thông tin đã khai thác. Việc thu gọn kích thước tập kết quả của quá trình khai thác nhằm giảm chi phí về bộ nhớ cũng như thời gian phân tích các kết quả nhận được nhưng không làm mất mát lượng tri thức đã khai thác. Nội dung nghiên cứu này được triển khai và hiện thực hóa tại Chương 2.

- **Nội dung 2:** Tìm hiểu các kỹ thuật, chiến lược nhằm kết hợp bài toán khai thác tập mục hữu ích cao, tập mục hữu ích cao đóng với định dạng CSDL có phân cấp các hạng mục và cải thiện hiệu năng khai thác của bài toán thực tiễn này. Định dạng CSDL phân cấp tiềm ẩn một lượng lớn tri thức có thể hỗ trợ người khai thác

trong việc ra quyết định. Tuy nhiên, bài toán có không gian tìm kiếm rất lớn đồng thời với số lượng tập mục kết quả gia tăng đáng kể. Chương 3 triển khai, giải quyết bài toán này thông qua một loạt các tiếp cận từ tuần tự đến song song hóa.

- **Nội dung 3:** Tối ưu hóa quá trình khai thác HUI trên định dạng CSDL phân cấp với không gian tìm kiếm lớn, tận dụng hiệu quả hơn nữa sức mạnh tính toán của các vi xử lý đa nhân phổ biến hiện nay để tăng hiệu quả của bài toán. Chương 4 đề xuất một số kỹ thuật, chiến lược xử lý để cải thiện hiệu năng khai thác song song, tận dụng tốt hơn khả năng xử lý song song của các vi xử lý đa nhân. Từ đó, Chương 4 cũng đề xuất hai giải thuật để hiện thực hóa nội dung nghiên cứu này.

2. Mục tiêu nghiên cứu

Kỹ thuật tính toán đa nhân được đề xuất để tận dụng tối đa năng lực xử lý của các bộ vi xử lý hiện đại ngày nay, vốn được thiết kế để có thể tích hợp nhiều nhân vi xử lý nhỏ hơn vào trong một đế, có giá thành ngày càng hợp lý và ngày càng phổ biến [42–44]. Thông thường, các vi xử lý đa nhân thông dụng có thể chứa từ hai, bốn, tám hoặc nhiều hơn nữa các nhân xử lý vật lý. Các vi xử lý này cùng với các kỹ thuật xử lý đa nhân hiện đang được triển khai rộng rãi trên nhiều ứng dụng để cải thiện hiệu năng làm việc với chi phí thấp. Một số ứng dụng thực tế hiện nay sử dụng các kỹ thuật xử lý đa nhân này có thể kể đến như raytracing, phân tích hình ảnh, máy chủ dịch vụ web, v.v… Tuy nhiên, việc áp dụng mô hình xử lý đa nhân vào bài toán HUIM, đặc biệt là áp dụng trên mô hình CSDL có phân cấp hạng mục, còn rất hạn chế [50]. Vì vậy, tập trung vào việc nghiên cứu và đề xuất ra giải pháp cho phép đưa các vi xử lý đa nhân vào các giải thuật HUIM để cải thiện hiệu năng của quá trình khai thác là cấp thiết. Các nghiên cứu trước đây về hướng tiếp cận này vẫn còn một số nhược điểm mà luận án có thể tập trung nghiên cứu để cải thiện [50] như:

- Các nghiên cứu chưa tận dụng tối đa năng lực tính toán của các vi xử lý đa nhân [50]. Ngay cả khi được triển khai trên các vi xử lý có nhiều nhân, các tiếp cận đã được đề xuất này vẫn có thể không tận dụng tối đa số nhân tính toán được cấp phát. Hơn nữa, quá trình song song chỉ được triển khai trong quá trình khai thác.

- Chiến lược điều phối hiện đang được sử dụng trong các tiếp cận hiện có chỉ đơn thuần là "đến trước, phục vụ trước" khi quét không gian tìm kiếm đã được song song hóa [50]. Do áp dụng chiến lược điều phối đơn giản, vì vậy mức độ hiệu quả của cơ chế song song hoá cũng không cao do thời gian chờ trung bình của các tác vụ xử lý song song vẫn còn cao. Các tác vụ có thời gian thực hiện ngắn đôi khi thực thi sau các tác vụ có thời gian khai thác dài, làm tăng thời gian khai thác.

Vì vậy, luận án tập trung vào việc tìm hiểu các kỹ thuật song song hoá cũng như các chiến lược điều phối hiệu quả để áp dụng chúng vào bài toán HUIM trên CSDL có sự phân cấp các hạng mục. Luận án đi sâu vào việc tìm hiểu các mặt tồn tại của những phương pháp khai thác tập mục hữu ích cao hiện có, từ đó đề xuất các kỹ thuật cải tiến phù hợp cũng như áp dụng mô hình tính toán song song vào bài toán HUIM này. Qua đó, giúp tận dụng tối đa năng lực tính toán của các bộ vi xử lý đa nhân hiện có với chi phí thấp và từ đó cải thiện hiệu năng của quá trình khai thác HUI.

Lý do lựa chọn áp dụng mô hình song song hóa đa nhân vào các giải thuật HUIM là do mô hình xử lý này được sự hỗ trợ trực tiếp từ các hệ điều hành và các thư viện lập trình hệ thống của các ngôn ngữ lập trình hiện đại ngày nay như C/C++/Java. Việc áp dụng mô hình song song hóa này vào các giải thuật không đòi hỏi phải sử dụng thêm các môi trường lập trình chuyên biệt hoặc các thư viện có độ phức tạp cao như Hadoop, Spark.

Qua khảo sát, cho đến nay chưa có nghiên cứu nào trong lĩnh vực khai thác tập mục hữu ích cao tập trung vào bài toán khai thác tập mục hữu ích cao trên CSDL phân cấp kết hợp với phương pháp xử lý song song để tăng hiệu quả khai thác của bài toán. Vì vậy, hướng nghiên cứu của luận án là nghiên cứu mới có tính cấp thiết.

3. Phạm vi nghiên cứu

Luận án tìm hiểu khái niệm, nguyên lý và đặc trưng của các tập mục hữu ích cao, hữu ích cao đóng, độ hữu ích của các tập mục hữu ích cao này.

- Các phương pháp xử lý song song tận dụng mô hình xử lý đa nhân.

- Các giải thuật khai thác tập mục hữu ích cao hiện có cùng với các bài toán mở rộng, bao gồm:
  - Nghiên cứu bài toán khai thác các tập mục hữu ích cao, tập mục hữu ích cao đóng trong cơ sở dữ liệu.
  - Nghiên cứu bài toán khai thác tập mục hữu ích cao tổng quát từ CSDL có sự phân cấp các hạng mục.
  - Nghiên cứu bài toán khai thác tập mục hữu ích cao sử dụng mô hình tính toán song song.

- Các CSDL thực nghiệm hiện đang được sử dụng rộng rãi trong các báo cáo, nghiên cứu liên quan để nhằm đánh giá một cách khách quan hiệu năng các giải thuật khai thác tập mục hữu ích cao.

4. Đối tượng nghiên cứu

Luận án tập trung nghiên cứu các chủ thể sau:

- Các giải thuật khai thác tập mục hữu ích cao truyền thống hiện có cùng các chiến lược hiệu quả đi kèm như Two-Phase, FHM, EFIM, iMEFIM.

- Các giải thuật khai thác biểu diễn cô đọng của tập mục hữu ích cao như CHUI-Miner, CLS-Miner, EFIM-Closed cùng các kỹ thuật liên quan.

- Các giải thuật khai thác tập mục hữu ích cao dựa trên cấu trúc phân cấp các hạng mục như MLHUI-Miner, CLH-Miner.

- Các giải thuật khai thác tập mục hữu ích cao sử dụng mô hình xử lý song song đa nhân như pEFIM, MCH-Miner.

- Định dạng CSDL có độ hữu ích tĩnh và định dạng CSDL có độ hữu ích biến đổi.

- Định dạng CSDL có sự phân cấp các hạng mục cùng với cấu trúc phân cấp.

- Các CSDL thực nghiệm chuẩn, công khai, thường dùng trong báo cáo khoa học.

5. Cách tiếp cận và phương pháp nghiên cứu

- Về lý thuyết:
  - Trang bị cơ sở lý thuyết về các dạng tập mục hữu ích cao như tập mục hữu ích truyền thống, các dạng biểu diễn tập mục hữu ích cao cô đọng, tập mục hữu ích cao trên CSDL phân cấp.
  - Trang bị cơ sở lý thuyết về các mô hình xử lý song song đa nhân.
  - Tìm hiểu về định dạng CSDL giao dịch và các định dạng mở rộng.
  - Khảo sát các phương pháp hiện có để khai thác các tập mục hữu ích cao trên CSDL giao dịch.
  - Tìm kiếm, đề xuất các kỹ thuật, mô hình để cải thiện hiệu quả của quá trình khai thác tập mục hữu ích cao.
  - Đề xuất phương pháp khai thác các tập hữu ích cao sử dụng mô hình xử lý song song dựa trên các cơ sở lý thuyết đã tìm hiểu.
- Về thực nghiệm:
  - Hiện thực hoá các giải thuật khai thác các tập mục hữu ích cao đã đề xuất.
  - Thực nghiệm, đánh giá khách quan mức độ hiệu quả của các giải thuật đã đề xuất trên một số bộ dữ liệu mẫu.

6. Các đóng góp của luận án

Luận án nghiên cứu và đề xuất các tiếp cận sau, tập trung vào việc cải thiện thời gian khai thác của bài toán khai thác tập mục hữu ích cao:

- Các tiếp cận cho phép khai thác hiệu quả các tập mục hữu ích cao sử dụng biểu diễn thu gọn và bảo toàn.

- Các mô hình song song hóa sử dụng kiến trúc xử lý đa nhân để rút ngắn thời gian thực hiện của các giải thuật khai thác tập mục hữu ích cao.

7. Ý nghĩa thực tiễn của luận án

Kết quả nghiên cứu của luận án có thể được áp dụng vào các bài toán sau đây:

- Giảm thiểu kích thước tập kết quả của bài toán khai thác luật kết hợp hữu ích cao (HARM) trên CSDL thông thường và cả trên các CSDL có phân cấp các hạng mục mà vẫn bảo toàn độ chính xác. Từ đó, có thể đưa ra được các hạng mục có liên quan với nhau và mang lại mức độ hữu ích cao trong quá trình vận hành các cửa hàng bán lẻ, siêu thị, v.v…

- Mở rộng các phương pháp khai thác tập mục hữu ích cao hiện có để tận dụng tối đa sức mạnh của các vi xử lý đa nhân đang được sử dụng rộng rãi hiện nay nhằm nâng cao hiệu quả của bài toán với chi phí thấp.

8. Cấu trúc luận án

Luận án được cấu trúc thành năm chương và các phần như sau:

- **Chương 1. (Tổng quan về lĩnh vực nghiên cứu)**: trình bày các nội dung tổng quan về bài toán khai thác mẫu nói chung và bài toán khai thác tập mục hữu ích cao nói riêng. Khảo sát các công trình có liên quan để nhằm tìm ra hướng phát triển, các hạn chế còn tồn tại trong hướng nghiên cứu và đề xuất hướng nghiên cứu để đáp ứng.

- **Chương 2. (Khai thác hiệu quả tập mục hữu ích cao đóng từ cơ sở dữ liệu có độ hữu ích biến động)**: nghiên cứu và đề xuất phương pháp để khai thác một cách hiệu quả các tập mục hữu ích cao đóng từ CSDL có độ hữu ích thay đổi.

- **Chương 3. (Khai thác song song các tập mục hữu ích cao đa mức từ cơ sở dữ liệu có sự phân cấp các hạng mục)**: nghiên cứu mô hình xử lý song song hoá dựa trên kiến trúc tính toán đa nhân để áp dụng vào bài toán khai thác tập mục hữu ích cao đa mức từ CSDL có sự phân cấp các hạng mục, phục vụ cho các ứng dụng trong thực tế.

- **Chương 4. (Mô hình song song hóa khai thác hiệu quả tập mục hữu ích cao đa mức)**: nghiên cứu và cải tiến sâu hơn nữa mô hình song song hoá dựa trên kiến trúc tính toán đa nhân đã đề xuất ở Chương 3, thông qua các kỹ thuật, chiến lược hiệu quả. Từ đó tận dụng triệt để hơn nữa sức mạnh tính toán của các vi xử lý đa

nhân. Áp dụng mô hình vào hai giải thuật phục vụ cho bài toán khai thác tập mục hữu ích cao từ CSDL phân cấp.

- **Chương 5. (Kết luận và Hướng phát triển)**: tổng kết, đưa ra nhận định về các đóng góp của luận án và đề xuất hướng phát triển tiếp theo cho luận án.
- **Danh mục các công trình đã công bố của học viên**: liệt kê các công trình nghiên cứu đã được công bố trên các tạp chí khoa học chuyên ngành có uy tín và các hội thảo quốc tế chuyên ngành.
- **Danh mục tài liệu tham khảo**: liệt kê tất cả các tài liệu tham khảo đã được trích dẫn trong luận án.

CHƯƠNG 1 TỔNG QUAN VỀ LĨNH VỰC NGHIÊN CỨU

1.1 Giới thiệu

Là một hướng nghiên cứu về lĩnh vực khai thác mẫu, bài toán khai thác tập phổ biến thu hút nhiều sự quan tâm nghiên cứu từ cộng đồng khai thác dữ liệu. Mục tiêu của bài toán này là xác định ra các tập mục phổ biến từ CSDL giao dịch [1]. Nhiều kỹ thuật khai thác hiệu quả đã được đề xuất như Apriori [22], FP-Growth [51], Eclat [52], HMine [53], LCM [54], v.v… Như đã nêu, FIM là chỉ đơn thuần xem xét sự xuất hiện của mỗi hạng mục trong giao dịch mà không quan tâm đến số lượng của chúng. Khi một mặt hàng được chọn mua với số lượng là 1 hay 1000 hoặc nhiều hơn thì tầm quan trọng của mặt hàng này cũng đều như nhau. Kết quả là người dùng khi triển khai các giải thuật FIM sẽ bỏ qua các thông tin mà có thể mang lại các tri thức hữu ích khác. Vì vậy, bài toán khai thác tập mục hữu ích cao ra đời để nhằm tận dụng triệt để các thông tin sẵn có trong những CSDL giao dịch để cung cấp cho người dùng các tri thức hữu ích hơn trong thực tế [13,14,55]. Cụ thể, các thông tin này bao gồm giá trị biểu thị số lượng của các mặt hàng khi được khách hàng chọn mua. Ngoài ra, mỗi mặt hàng khi được kinh doanh sẽ hiển nhiên kèm theo đơn giá của mặt hàng đó. Độ hữu ích nhận được khi mặt hàng đó được bán đơn thuần chỉ là tích của hai đại lượng này. Các mặt hàng có độ hữu ích thoả một ngưỡng độ hữu ích tối thiểu $minutil$ cho trước thì được gọi là tập mục hữu ích cao.

Xét một ví dụ về CSDL giao dịch cho tại Bảng 1.1 với 8 giao dịch. Trong đó, mỗi giao dịch mang một định danh giao dịch duy nhất (TID) trong toàn bộ CSDL. Trong mỗi giao dịch, ta có các thông tin sau: các hạng mục (item) cùng với số lượng đã được giao dịch (độ hữu ích nội). Lưu ý, cột TU biểu thị giá trị tổng độ hữu ích của các hạng mục trong từng giao dịch, và được thể hiện để giúp dễ theo dõi giá trị độ hữu ích của từng giao dịch. Đơn giá (external utility, độ hữu ích ngoại/độ hữu ích đơn vị) của từng hạng mục được cho tại Bảng 1.2. Giả sử, doanh thu có được của hạng mục a trong giao dịch T_2 có thể được xác định bằng cách lấy số lượng giao dịch của a tại T_2 là 2 nhân với đơn giá của a là 5, doanh thu có được khi bán hạng mục a là 10. Tương

tự với các hạng mục còn lại trong CSDL. Đối với bài toán HUIM, ta gọi chung khái niệm về doanh thu, trọng số hoặc tầm quan trọng của một hạng mục là *độ hữu ích* của hạng mục đó. Định nghĩa hình thức về một CSDL giao dịch cũng như các phương pháp xác định độ hữu ích của các hạng mục, tập mục, v.v… được trình bày chi tiết trong các chương sau.

1.2 Các nghiên cứu liên quan

Phần này trình bày khảo sát về hướng nghiên cứu khai thác tập mục hữu ích cao tổng quát. Ngoài ra, các nhánh sâu hơn của lĩnh vực cũng được tìm hiểu để rút ra được các ưu khuyết điểm của các giải thuật đã được đề xuất. Từ đó, định ra các tiếp cận, mô hình để giải quyết các mặt còn tồn tại của những phương pháp này.

Bảng 1.1. Ví dụ về CSDL giao dịch

TID	**Các hạng mục**	**Độ hữu ích nội**	**TU**
T_1	a, c, d	$1, 1, 1$	8
T_2	a, b, c, e	$2, 2, 6, 2$	26
T_3	a, b, c, d, e, f	$1, 2, 1, 6, 1, 5$	30
T_4	a, b, e	$1, 4, 1$	16
T_5	b, c, e	$1, 1, 1$	9
T_6	a, c, d	$3, 3, 3$	24
T_7	a, b, c, d, f	$1, 1, 1, 2, 3$	15
T_8	a, c, d, e, f	$1, 2, 2, 1, 1$	15

Bảng 1.2. Độ hữu ích ngoại của các hạng mục cho tại Bảng 1.1

Item	a	b	c	d	e	f
Độ hữu ích ngoại	5	2	1	2	3	1

1.2.1 Bài toán khai thác tập mục hữu ích cao

Bài toán HUIM được đề xuất bởi Yao và đồng sự vào năm 2004 [13]. Bài toán này được xem như là một bài toán có mức độ tổng quát hơn so với FIM [1,13]. Tương tự FIM, bài toán HUIM cũng thu hút được nhiều sự quan tâm nghiên cứu để nhằm cải thiện hiệu quả của bài toán. Ban đầu, khi đề xuất bài toán HUIM, Yao và đồng sự cũng giới thiệu hai tiếp cận để giải quyết bài toán này là UMining và UMining-H [55].

Tương tự FIM, HUIM cũng là một bài toán dạng liệt kê tập hợp, vì vậy ứng với mỗi ngưỡng $minutil$, luôn chỉ có duy nhất một lời giải. Đầu ra của các giải thuật là tập đầy đủ các tập mục hữu ích cao. Ta có thể áp dụng phương pháp vét cạn, là phương pháp đơn giản nhất, để thực hiện công việc liệt kê các tập mục thoả ngưỡng $minutil$. Tuy nhiên, không gian tìm kiếm của phương pháp vét cạn trong trường hợp xấu nhất là $O(2^n)$ với n là số hạng mục trong CSDL giao dịch. Điều này dễ dàng đưa đến sự bùng nổ tổ hợp nếu n lớn. Và trong thực tế n luôn lớn vì đây chính là số lượng mặt hàng được kinh doanh trong một CSDL giao dịch.

Một vài giải thuật đã được đề xuất nhằm phục vụ bài toán HUIM như Two-Phase [14], TWU-Mining [56], dTWU-Mining [15], HUI-Miner [57], FHM [16], EFIM [58], HMiner [59], và iMEFIM [17]. Tất cả các giải thuật nói trên đều nhận đầu vào là một CSDL giao dịch cùng với một ngưỡng $minutil$. Khác biệt giữa các giải thuật này chính là các cấu trúc dữ liệu, chiến lược quét không gian tìm kiếm và các kỹ thuật hiệu quả để loại bỏ các tập hợp ứng viên không tiềm năng trong quá trình khai thác.

Mặc dù HUIM là một bài toán mở rộng từ FIM, bài toán này lại có độ phức tạp cao hơn so với bài toán FIM [14]. Nguyên nhân là do độ đo độ hữu ích các hạng mục không thoả tính chất DCP. Vì vậy, các giải thuật HUIM không thể sử dụng trực tiếp độ đo này để loại bỏ một cách an toàn các ứng viên không tiềm năng khỏi không gian tìm kiếm.

Hai giải thuật đầu tiên phục vụ bài toán HUIM là UMining và UMining-H [55]. Trong một số trường hợp, hai giải thuật này không trả về đầy đủ các tập mục hữu ích cao.

Vì vậy, hai tiếp cận này chỉ được xem là giải thuật xấp xỉ ban đầu để nhằm giải quyết bài toán đã đề xuất. Lý do chính là việc chưa thể có một ngưỡng giới hạn trên phù hợp để tỉa an toàn các ứng viên không tiềm năng giống như bài toán FIM, mãi cho đến năm 2005 khi xuất hiện giải thuật Two-Phase và ngưỡng giới hạn trên TWU [14].

Các giải thuật HUIM có thể được chia thành hai nhóm chính: nhóm thực hiện qua hai giai đoạn (hai pha) và nhóm chỉ cần một giai đoạn khai thác (một pha).

Two-Phase [14], IHUP [60], TWU-Mining [56], dTWU-Mining [15], UP-Growth [20], UP-Growth+ [61] là một vài ví dụ về các giải thuật hai pha. Ở pha đầu tiên, các giải thuật quét CSDL để phát sinh ra các tập ứng viên. Ở pha thứ hai, CSDL lại được quét qua một lần nữa để xác định chính xác các ứng viên tiềm năng và lọc ra các tập thoả mãn ngưỡng $minutil$.

Two-Phase mở rộng Apriori với chiến lược tìm kiếm theo chiều rộng. Các tác giả của giải thuật cũng lần đầu tiên đề xuất khái niệm về ngưỡng giới hạn trên (upper bound) để có thể lọc bớt các ứng viên không tiềm năng. Giới hạn trên được sử dụng trong giải thuật Two-Phase được gọi là Transaction Weighted Utilization (TWU) [14]. TWU được chứng minh bởi các tác giả rằng nó thoả tính chất DCP và có thể áp dụng để tỉa an toàn các tập ứng viên có $TWU < minutil$. Định nghĩa về giới hạn trên TWU được trình bày trong CHƯƠNG 2 tại **Định nghĩa 2.7**. TWU được sử dụng trong toàn bộ các tiếp cận khai thác tiếp theo nhằm kéo giảm số lượng ứng viên không tiềm năng, kết hợp với các chiến lược tỉa ứng viên khác chặt chẽ hơn. Chiến lược tỉa ứng viên dựa trên TWU được trình bày tại **Tính chất 2.1** và **Tính chất 2.2**.

Năm 2009, giải thuật IHUP được Ahmed và đồng sự giới thiệu [60]. Đây là phiên bản mở rộng dựa trên FP-Growth được đề xuất bởi Han và đồng sự [51]. Giải thuật này trình bày cấu trúc cây IHUP-Tree để khai thác trực tiếp các tập HUI. Tuy nhiên, tương tự FP-Growth, IHUP cũng gặp phải giới hạn khi triển khai trên các CSDL lớn khi phải đưa toàn bộ CSDL lưu trữ trên cây IHUP-Tree. TWU tiếp tục được sử dụng trong tất cả các giải thuật HUIM được đề xuất tiếp sau đó.

Giải thuật TWU-Mining được đề xuất dựa trên việc mở rộng cấu trúc dữ liệu IT-Tree [62] để trở thành cấu trúc WIT-Tree, qua đó sử dụng WIT-Tree để thu gọn không gian bài toán và giảm thời gian khai thác. dTWU-Mining với việc sử dụng diffset vào TWU-Mining làm giảm hơn nữa thời gian khai thác cũng như bộ nhớ cần sử dụng và tăng tốc quá trình tính TWU [63]. Cả TWU-Mining và dTWU-Mining đều áp dụng TWU để lược bỏ các ứng viên không tiềm năng.

UP-Growth/UP-Growth+ [61] là các phương pháp dựa trên kỹ thuật tăng trưởng mẫu (pattern growth) do Han và đồng sự đề xuất năm 2004 thông qua giải thuật FP-Growth [51]. Đặc điểm của các tiếp cận dạng này là chi phí quét CSDL được giảm đáng kể, cùng hàng loạt các kỹ thuật loại bỏ sớm các tập mục không tiềm năng trong quá trình khai thác. Tuy nhiên, việc đưa toàn bộ cấu trúc CSDL giao dịch lên bộ nhớ có thể gây ra vấn đề nếu như CSDL khai thác có kích thước vượt ngoài khả năng lưu trữ của bộ nhớ chính.

Các giải thuật này đều mất nhiều thời gian thực thi và có mức độ sử dụng bộ nhớ cao. Các tiếp cận sau đó được đề xuất đều chỉ cần một giai đoạn khai thác. Các giải thuật một pha tính trực tiếp độ hữu ích của các ứng viên mà không cần phải phát sinh và lưu trữ chúng trên bộ nhớ. Một số giải thuật HUIM một pha đầu tiên được đề xuất là HUI-Miner [57] và d^{2}HUP [64].

Cả hai giải thuật HUI-Miner [57] và d^{2}HUP [64] cũng đồng giới thiệu một ngưỡng giới hạn trên chặt chẽ hơn TWU gọi là *remaining utility* và sử dụng ngưỡng này để loại bỏ một lượng lớn các ứng viên không có khả năng trở thành HUI. Bên cạnh đó, HUI-Miner cũng giới thiệu một cấu trúc dữ liệu tiên tiến có tên gọi là utility-list [57]. Cấu trúc dữ liệu này vẫn còn đang được áp dụng rộng rãi trong các tiếp cận sau này. Cấu trúc này cho phép tăng tốc quá trình tính độ hữu ích cho các tập mục. Chi tiết về giới hạn trên remaining utility được trình bày trong CHƯƠNG 3, **Định nghĩa 3.1** và **Định nghĩa 3.2**. Cấu trúc utility-list được mô tả tại **Định nghĩa 3.3**.

Giải thuật FHM kế thừa các ưu điểm từ HUI-Miner và đề xuất thêm một cấu trúc dữ liệu mới, có tên gọi EUCS [16]. Cấu trúc này được xây dựng chỉ với một lần quét

CSDL và được sử dụng trong chiến lược tỉa ứng viên có tên gọi EUCP. Chiến lược này được áp dụng để loại bỏ các kết hợp tập mục không tiềm năng trước khi xây dựng hoặc kết các utility-list của chúng. Kỹ thuật này giúp giảm đáng kể chi phí của quá trình khai thác vì giải thuật không mất thời gian để thực hiện phép kết các utility-list của những kết hợp mà sau đó được xác định là không thể hình thành các tập mục hữu ích cao. Chi phí phép kết hai utility-list kích thước m và n xấp xỉ $O(m \log_2 n)$. Chi tiết về cấu trúc EUCS cũng như chiến lược tỉa ứng viên EUCP được trình bày trong CHƯƠNG 3, **Định nghĩa 3.14** và **Tính chất 3.4**.

Krishnamoorthy đề xuất giải thuật HUP-Miner [65] dựa trên việc mở rộng giải thuật HUI-Miner. HUP-Miner thực hiện việc tỉa ứng viên bằng cách phân hoạch CSDL và áp dụng chiến lược dò tìm trước để tăng tốc quá trình xây dựng utility-list.

Ryang và đồng sự đề xuất giải thuật SHU-Growth vào năm 2016 cùng với cấu trúc SIQ-Tree và hai chiến lược có tên gọi lần lượt là RPS và REPU để tăng tốc quá trình khai thác HUI [66]. Mục tiêu chính được nhắm đến của các cách tiếp cận đã trình bày là cải thiện hiệu năng của quá trình khai thác HUI thông qua việc tăng hiệu suất của các chiến lược tỉa ứng viên, càng nhiều các tập ứng viên không tiềm năng được loại bỏ, chi phí tính toán càng được giảm thiểu.

Giải thuật EFIM [58] đề xuất hai ngưỡng giới hạn trên chặt chẽ hơn cả TWU và remaining utility, hai giới hạn trên này được gọi là *local utility* và *subtree utility*. Giải thuật này kết hợp các ngưỡng giới hạn trên đã có với các ngưỡng vừa được đề xuất để thu gọn hơn nữa không gian của bài toán. Bên cạnh đó, các tác giả cũng đề xuất các kỹ thuật cho phép giảm chi phí quét CSDL và bộ nhớ sử dụng, gọi là các kỹ thuật ghép giao dịch (HTM) và chiếu giao dịch (HDP). Các định nghĩa liên quan đến *local utility* và *subtree utility* được giới thiệu tại CHƯƠNG 2, **Định nghĩa 2.11**, **Định nghĩa 2.12**, cùng các tính chất tương ứng là **Tính chất 2.3**, **Tính chất 2.4**. Các giới hạn trên này được áp dụng trực tiếp vào quá trình khai thác để thu gọn đáng kể không gian tìm kiếm của bài toán. Bên cạnh đó, kỹ thuật HTM và HDP cũng được trình bày tại **Định nghĩa 2.9** và **Định nghĩa 2.10** cùng trong CHƯƠNG 2.

Tận dụng các kỹ thuật tỉa ứng viên đã có trước đó như TWU-Prune [14], LA-Prune [65], EUCS-Prune [16], U-Prune [67], vào năm 2017, Krishnamoorthy giới thiệu thêm một kỹ thuật mới là C-Prune, kết hợp chúng lại để phát triển một giải thuật có tên gọi là HMiner [59] cho phép khai thác HUI hiệu quả hơn.

Duong và đồng sự năm 2018 đề xuất một giải thuật mở rộng dựa trên FHM có tên gọi là ULBMiner [68]. Giải thuật này có khả năng tái sử dụng lại bộ nhớ thông qua cấu trúc dữ liệu bộ đệm utility-list, cho phép giảm thiểu lượng bộ nhớ cần thiết để thực hiện giải thuật và cải thiện hiệu năng của quá trình khai thác HUI.

Giải thuật iMEFIM [17] mở rộng hơn nữa EFIM thông qua việc đề xuất cấu trúc P-set để giảm thiểu đáng kể chi phí quét CSDL, đặc biệt trên các CSDL có mật độ thấp khi mà kỹ thuật HTM và HDP không phát huy được hiệu quả. iMEFIM cũng giới thiệu giải pháp để xử lý các CSDL có độ hữu ích đơn vị của các hạng mục là không cố định nhằm thích nghi tốt hơn nữa với các ứng dụng trong thực tế. Cấu trúc dữ liệu P-set được mô tả và áp dụng tại CHƯƠNG 2, **Định nghĩa 2.20**, **Định nghĩa 2.21** để cải thiện một cách rõ rệt hiệu năng khai thác thông qua việc giảm chi phí quét CSDL.

Năm 2023, Qu và đồng sự đề xuất một giải thuật có tên gọi là HAMM. Đây là giải thuật một-pha, lấy cảm hứng từ giải thuật FP-Growth, kết hợp cấu trúc cây tiền tố và véc-tơ độ hữu ích để khai thác các tập mục hữu ích cao [69]. Ngoài ra, nhóm tác giả còn đề xuất một số kỹ thuật tối ưu hoá khác để nhằm tăng tốc quá trình khai thác.

Bên cạnh bài toán HUIM truyền thống, nhiều biến thể của bài toán này cũng được đề xuất để đáp ứng các ràng buộc phức tạp và áp dụng thực tế trong các ứng dụng.

Tóm tắt các giải thuật khai thác tập mục hữu ích cao cùng với các đặc điểm và năm công bố của các giải thuật này được cho tại Bảng 1.3.

Bảng 1.3. Các giải thuật khai thác tập mục hữu ích cao

Giải thuật	**Mô tả**	**Đặc điểm**	**Năm**
UMining, UMining-H	Mô hình tổng quát cùng với các đặc điểm toán học cho độ đo lợi ích và bài toán HUIM [13].	Sử dụng kỹ thuật heuristic để cắt tỉa các ứng viên	2004
Two-Phase	Đề xuất kỹ thuật dựa vào TWU để khai thác HUI trong hai giai đoạn [14].	Cắt tỉa đáng kể các ứng viên không tiềm năng.	2005
IHUP	Khai thác HUI sử dụng tiếp cận cấu trúc cây [60].	Cho phép nén tập dữ liệu thông qua cấu trúc cây và có hiệu năng tốt hơn đáng kể so với Two-Phase.	2009
TWU-Mining	Sử dụng cấu trúc WIT-Tree để khai thác HUI [62].	Giảm số lượng ứng viên bằng TWU và quét CSDL một lần duy nhất. Hiệu năng cải thiện đáng kể so với Two-Phase.	2009
UP-Growth	Sử dụng kỹ thuật tăng trưởng mẫu thông qua cấu trúc cây UP-Tree [61].	UP-tree có khả năng nén dữ liệu cao hơn đáng kể so với IHUP-Tree. Sử dụng các kỹ thuật cắt tỉa để thu nhỏ không gian tìm kiếm.	2010

Giải thuật	Mô tả	Đặc điểm	Năm
HUI-Miner	Giải thuật khai thác HUI một giai đoạn đầu tiên [57].	Đề xuất cấu trúc utility-list.	2012
UP-Growth+	Cải tiến UP-Growth với chiến lược cắt tỉa mới [61].	Giảm thiểu đáng kể các tập ứng viên được phát sinh.	2013
FHM	Phiên bản cải tiến của HUI-Miner với cấu trúc EUCS và chiến lược cắt tỉa EUCP [16].	Giảm thiểu đáng kể chi phí phép kết utility-list.	2014
HUP-Miner	Bản cải tiến của HUI-Miner với hai chiến lược cắt tỉa là LA-Prune và PU-Prune [65].	Giảm thiểu số lượng phép kết các utility-list.	2015
EFIM	Sử dụng các kỹ thuật ghép và chiếu giao dịch để giảm thiểu chi phí quét CSDL [58].	Tiêu thụ ít bộ nhớ và có độ phức tạp gần như tuyến tính với số hạng mục trong không gian tìm kiếm.	2015
SHU-Growth	Tăng tốc quá trình khai thác thông qua cấu trúc SIQ-Tree và hai chiến lược RPS và REPU [66].	Thuộc mô hình tăng trưởng mẫu nên vẫn gặp phải các vấn đề khi làm việc với CSDL lớn	2016

Giải thuật	Mô tả	Đặc điểm	Năm
HMiner	Kết hợp nhiều chiến lược cắt tỉa hiệu quả khác nhau để giảm số lượng ứng viên cần xét trong khi khai thác [59].	Số lượng các ứng viên cần xét được giảm đáng kể do việc kết hợp hàng loạt các kỹ thuật cắt tỉa ứng viên hiệu quả.	2017
ULBMiner	Giải quyết các vấn đề tồn đọng về việc quản lý và thực hiện phép kết trên các utility-list [68].	Cải tiến cấu trúc utility-list trở thành cấu trúc utility-list buffer để giảm việc tiêu thụ bộ nhớ và tăng tốc phép kết các utility-list.	2017
iMEFIM	Mở rộng EFFIM và đề xuất bài toán khai thác HUI từ các tập dữ liệu có lợi ích biến đổi [17].	Giảm thiểu đáng kể chi phí quét CSDL thông qua cấu trúc P-set. Hiệu năng trên tập dữ liệu mật độ thấp tăng đáng kể và số lượng giao dịch cần quétgiảm lên đến 99%.	2019
HAMM	Kết hợp cấu trúc cây tiền tố và cấu trúc vector độ hữu ích để tăng tốc quá trình khai thác [69].	Cho hiệu năng khai thác vượt trội các giải thuật state-of-the-art trước đó như HMiner.	2023

1.2.2 Bài toán khai thác tập mục hữu ích cao với biểu diễn đóng

Với các quan sát và nhận định rằng bài toán HUI thường kết xuất quá nhiều tập mục kết quả, dẫn đến thời gian khai thác của giải thuật dài và lượng bộ nhớ sử dụng lớn.

Ngoài ra, sau khi đã có tập mục kết quả, ta cần phải có nhiều thời gian để phân tích các thông tin này. Vì vậy, biểu diễn thu gọn và bảo toàn thông tin của các tập mục hữu ích cao được đề xuất [70]. Biểu diễn này có tên gọi là tập mục hữu ích cao đóng (CHUI). Bài toán nhằm liệt kê đầy đủ các tập mục hữu ích cao đóng thoả một ngưỡng *minutil* cho trước được gọi là bài toán khai thác tập mục hữu ích cao đóng (CHUIM).

Ba giải thuật đầu tiên phục vụ cho bài toán CHUIM có tên gọi lần lượt là AprioriHC, AprioriHC-D và CHUD đã được Tseng và đồng sự đề xuất [70] nhằm phục vụ cho việc khai thác CHUI. Hai giải thuật đầu là mở rộng của giải thuật Apriori để khai thác các CHUI thông qua chiến lược tìm kiếm theo chiều rộng. Mở rộng từ Eclat [62] và DCI-Closed [71], CHUD sử dụng định dạng CSDL dọc cùng với chiến lược tìm kiếm theo chiều sâu để khai thác CHUI. Tại thời điểm giới thiệu, CHUD được xem là giải thuật khai thác CHUI tốt nhất. Để trích xuất lại các tập mục HUI từ các tập CHUI đã tìm được, Tseng và đồng sự cùng đề xuất một giải thuật có tên gọi là DAHU để thực hiện công việc này.

Tuy nhiên, ba giải thuật trên đều là các giải thuật hai giai đoạn. Năm 2015, Wu và đồng sự đề xuất một giải thuật khai thác CHUI hiệu quả có tên gọi là CHUI-Miner [27]. Đây là giải thuật khai thác CHUI đầu tiên sử dụng mô hình một pha. CHUI-Miner tính toán trực tiếp độ hữu ích của các tập mục trong bộ nhớ mà không qua giai đoạn phát sinh các tập ứng viên. Giải thuật CHUI-Miner đạt được điều này thông qua việc sử dụng một cấu trúc có tên gọi là EU-List. Cấu trúc này được sử dụng để quản lý các thông tin về độ hữu ích của các tập mục bên trong mỗi giao dịch, từ đó tính toán một cách nhanh chóng độ hữu ích của chúng trực tiếp trên bộ nhớ mà không cần phải quétlại CSDL. Tuy nhiên, CHUI-Miner sử dụng ngưỡng giới hạn trên TWU để tỉa các ứng viên không tiềm năng khỏi không gian tìm kiếm. Vì vậy, giải thuật này cần phải quét qua một lượng lớn các tập mẫu. Ngoài ra, việc kiểm tra tính đóng của các tập mục còn chưa hiệu quả.

Năm 2016, Fournier-Viger và đồng sự đề xuất một giải thuật khai thác CHUI có tên gọi là EFIM-Closed [18]. Giải thuật EFIM-Closed áp dụng mô hình một giai pha, và

là mở rộng của giải thuật EFIM trước đó. EFIM-Closed đưa ra ba kỹ thuật kiểm tra tính đóng của các tập HUI gồm Closure Jumping (CJ), Forward Closure Checker (FCC) và Backward Closure Checker (BCC). Thực nghiệm cho thấy giải thuật EFIM-Closed có hiệu năng khai thác tốt khi áp dụng lên các CSDL dày. Chi tiết về các kỹ thuật CJ, FCC và BCC được mô tả tại CHƯƠNG 2, **Định nghĩa 2.19** cùng các **Tính chất 2.5** đến **2.7**. Các tính chất này giúp cho việc xác định tính đóng của các tập mục được hiệu quả hơn.

Dam và đồng sự giới thiệu giải thuật có tên gọi là CLS-Miner [35] vào năm 2019. CLS-Miner thực hiện việc tỉa không gian tìm kiếm thông qua ba kỹ thuật có tên gọi lần lượt là Chain-EUCP, LBP và tỉa dựa trên độ phủ. Các kỹ thuật này giúp tỉa hiệu quả các tập mục không tiềm năng ra khỏi không gian tìm kiếm trước khi utility-list của chúng được xây dựng, cũng như tối ưu hóa quá trình kiểm tra tính đóng của các tập mục.

Mai và Nguyen vào năm 2017 giới thiệu một giải thuật khai thác CHUI và các tập sinh dựa trên dàn có tên gọi LHUCI-Miner [29]. Trong quá trình xây dựng dàn, các tập HUI được so sánh để gán nhãn là tập đóng hoặc tập sinh. Tiếp đến, giải thuật này có thể rút ra được tập các CHUI một cách nhanh chóng.

Tóm tắt các giải thuật khai thác tập mục hữu ích cao đóng cùng với các đặc điểm và năm công bố của các giải thuật này được cho tại Bảng 1.4.

Bảng 1.4. Các giải thuật khai thác tập mục hữu ích cao đóng

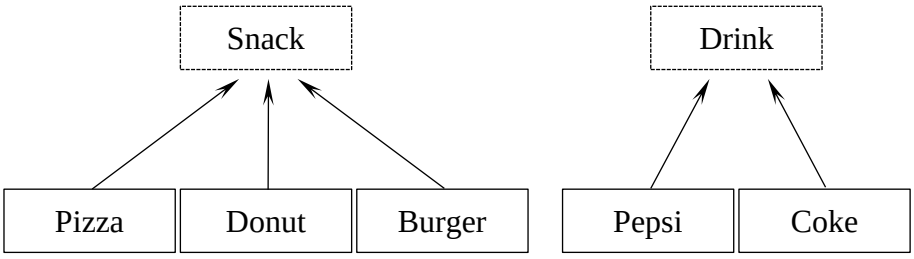
Giải thuật	**Mô tả**	**Đặc điểm**	**Năm**
AprioriHC AprioriHC-D CHUD	Đề xuất bài toán khai thác các tập hữu ích cao, đóng [70].	Hiệu năng khai thác các tập CHUI cao nhờ vào việc giảm số lượng ứng viên. Có khả năng kết xuất lại tập đầy các HUI	2011

Giải thuật	Mô tả	Đặc điểm	Năm
		từ các tập CHUI đã khai thác.	
CHUI-Miner	Sử dụng utility-list để khai thác các tập CHUI chỉ trong một giai đoạn. Là phiên bản mở rộng của HUI-Miner [27].	Mở rộng cấu trúc utility-list để tạo ra cấu trúc EU-List nhằm giảm thiểu lượng bộ nhớ sử dụng. Hiệu quả về mặt thời gian là tốt hơn so với CHUD.	2015
EFIM-Closed	Mở rộng giải thuật EFIM với các kỹ thuật kiểm tra bao đóng nhanh [18].	Các kỹ thuật tính nhanh bao đóng hiệu quả, tăng tốc quá trình khai thác các tập CHUI.	2016
CLS-Miner	Sử dụng cấu trúc utility-list để xác định trực tiếp các giá trị lợi ích mà không cần phát sinh các tập ứng viên [35].	Đề xuất 3 kỹ thuật cắt tỉa là Chain-EUCP mở rộng dựa trên EUCP của FHM; kỹ thuật Lower Branch Pruning loại bỏ các mở rộng không tiềm năng khỏi không gian tìm kiếm; kỹ thuật Coverage cho phép tính nhanh bao đóng và loại bỏ nhanh các tập hữu ích thấp.	2017
LHUCI-Miner	Khai thác CHUI và các tập sinh dựa trên dàn [29].	Sử dụng dàn để phát hiện nhanh CHUI và các tập	2017

Giải thuật	Mô tả	Đặc điểm	Năm
		sinh thông qua việc đánh nhãn các tập HUI.	

1.2.3 Bài toán khai thác tập mục hữu ích cao trên CSDL phân cấp

Khái niệm về khai thác HUI từ các CSDL có sự phân cấp các hạng mục được Cagliero và đồng sự đề xuất vào năm 2017 [41]. Đây chính là sự mở rộng từ bài toán khai thác tập phổ biến tổng quát đã được khảo sát chi tiết trước đó của [34–40]. Bài toán mở rộng này có vai trò quan trọng trong các ứng dụng thực tế, kết quả khai thác được mang lại nhiều ý nghĩa hơn cho người sử dụng vì có mức độ tổng quát cao hơn. Tuy nhiên, bài toán này có độ phức tạp cao hơn so với bài toán HUIM truyền thống vì không gian tìm kiếm được mở rộng thông qua cấu trúc cây phân cấp taxonomy.



Hình 1.1. Minh hoạ một cấu trúc taxonomy

Ví dụ, một cửa hàng bán đồ ăn nhanh có các món sau trong thực đơn: Coke, Pepsi, Burger, Donut và Pizza. Trong thực tế, các mặt hàng này có thể nhiều hơn và có thể được nhóm thành các nhóm có mức độ trừu tượng cao hơn, ví dụ như "*Drink*" để mô tả các món đồ uống như Coke, Pepsi, "*Snack*" để mô tả các món ăn nhanh Donut, Pizza, Burger, v.v… (Hình 1.1).

Các giải thuật HUIM truyền thống chỉ xem xét các hạng mục xuất hiện ở mức trừu tượng thấp nhất mà thôi. Để giải quyết vấn đề xem xét các hạng mục ở mức tổng quát cao hơn, Cagliero đề xuất một phương pháp khai thác có tên gọi là MLHUI-Miner [41] nhằm khai thác tất cả các tập mục hữu ích cao tổng quát từ CSDL phân cấp. Nhóm tác giả cũng đề xuất khái niệm về cây phân cấp các hạng mục, được gọi là

taxonomy, bên trong một CSDL. Kết xuất của MLHUI-Miner là các tập mục chứa những hạng mục xuất hiện trên cùng một mức trừu tượng và được gọi là các tập mục HUI đa mức.

Gần đây, bài toán này có thêm nhiều sự quan tâm do tính chất quan trọng trong các ứng dụng thực tế. Sivamathi và Vijayarani [72] giới thiệu một giải thuật có tên gọi MUMA để thực hiện công việc tương tự. Giải thuật này triển khai một cấu trúc có tên MUMT để chứa thông tin về độ hữu ích của các tập mục. Trong giải thuật này, mỗi hạng mục được mã hoá dựa trên mức trừu tượng của chúng thay vì sử dụng taxonomy, và ở mỗi mức khác nhau thì được gán một giá trị $minutil$ khác nhau. Tuy nhiên, các tác giả chỉ thực nghiệm đánh giá phương pháp đề xuất trên các CSDL tổng hợp và có kích thước nhỏ, do mức độ phức tạp của bài toán là rất lớn.

Tung và đồng sự giới thiệu giải thuật MLHMiner vào năm 2022 [73], và được xem là một biến thể của MLHUI-Miner với giải thuật lõi là HMiner [59] được sử dụng thay cho FHM [16] của MLHUI-Miner. Trong cùng năm, giải thuật CLH-Miner được đề xuất [74] để khai thác các tập mục HUI liên cấp. Các tác giả cũng đề xuất khái niệm về ngưỡng Generalized Weighted Utlization, một ngưỡng giới hạn trên mở rộng từ TWU để tỉa các ứng viên là các tập mục tổng quát. Tuy nhiên với không gian bài toán gia tăng đáng kể với khái niệm liên cấp – các tập mục giờ đây có thể chứa các hạng mục không thuộc cùng một mức trừu tượng – giải thuật này có thời gian khai thác rất dài, đặc biệt trên các CSDL lớn. CLH-Miner cũng được áp dụng trong giải thuật TKC, được đề xuất sau đó bởi Nouioua và đồng sự, để áp dụng vào bài toán top-k [75].

Gần đây nhất, Tung và đồng sự đề xuất giải thuật FEACP để giải quyết một cách hiệu quả bài toán khai thác HUI liên cấp [76]. Thông qua việc mở rộng các ngưỡng giới hạn trên hiệu quả trước đó như *local utility* và *subtree utility* để có thể hoạt động được với cấu trúc taxonomy, giải thuật đã cải thiện đáng kể hiệu suất của quá trình khai thác HUI liên cấp.

Với bài toán khai thác này, không gian tìm kiếm của bài toán được mở rộng thông qua cấu trúc taxonomy. Cấu trúc taxonomy có kích thước càng lớn, không gian bài toán càng mở rộng. Điều này cũng đồng nghĩa với thời gian khai thác của bài toán sẽ kéo dài vì phải quét qua không gian tìm kiếm khổng lồ của bài toán. Tuy đã có nhiều kỹ thuật loại bỏ sớm các ứng viên không tiềm năng, thời gian khai thác của các giải thuật vẫn còn rất dài so với bài toán khai thác HUI truyền thống. Vì vậy, các phương pháp này cần được tiếp tục đi sâu nghiên cứu để nhằm rút ngắn thời gian cũng như tăng hiệu quả quá trình khai thác.

Tóm tắt các giải thuật khai thác tập mục hữu ích cao tổng quát với các đặc điểm và năm công bố của các giải thuật này được cho tại Bảng 1.5.

Bảng 1.5. Các giải thuật khai thác tập mục hữu ích cao tổng quát

Giải thuật	**Mô tả**	**Đặc điểm**	**Năm**
MLHUI- Miner	Mở rộng giải thuật FHM để khai thác các tập mục hữu ích cao đa mức. Đề xuất cấu trúc phân cấp các hạng mục, dựa vào đó để khai thác tập mục hữu ích cao đa mức [41].	Tỉa ứng viên thông qua TWU. Thời gian khai thác dài và hoàn toàn không khả thi trên CSDL có kích thước lớn và số lượng hạng mục nhiều.	2017
CLH-Miner	Mở rộng bài toán khai thác tập mục hữu ích cao từ CSDL phân cấp để khai thác tập mục hữu ích cao liên cấp [74].	Đề xuất ngưỡng GWU để tỉa ứng viên. Độ phức tạp theo thời gian rất lớn, thời gian thực hiện khai thác rất dài.	2020
TKC	Kế thừa CLH-Miner để khai thác top-k tập mục hữu ích cao liên cấp [75].	Vì kế thừa CLH-Miner và kết hợp với bài toán	2020

Giải thuật	Mô tả	Đặc điểm	Năm
		top-k nên dẫn đến thời gian khai thác rất dài.	
MLHMiner	Mở rộng giải thuật HMiner cùng các chiến lược tỉa ứng viên để áp dụng được trên CSDL phân cấp [73].	Kế thừa các đặc điểm vượt trội của HMiner để áp dụng trên taxonomy.	2022
FEACP	Mở rộng giải thuật EFIM để khai thác các tập mục hữu ích cao liên cấp [76].	Là giải thuật khai thác CLHUI hiệu quả nhất tính đến nay.	2022

1.2.4 Bài toán khai thác song song các tập mục hữu ích cao

Mô hình tính toán song song hoặc phân tán cũng được áp dụng vào hướng tiếp cận HUIM bên cạnh FIM để giảm thời gian khai thác HUI.

Vo và các đồng sự giới thiệu một giải thuật cho phép khai thác HUI từ các CSDL dọc được lưu trữ phân tán vào năm 2009 [5] và giảm thiểu thời gian cần thiết cho quá trình khai thác. Giải thuật PHUI-Miner [45] được Chen và An đề xuất dựa trên việc mở rộng giải thuật HUI-Miner sử dụng mô hình tính toán phân tán. Tương tự, Sethi và đồng sự đề xuất một phiên bản phân tán của giải thuật FHM+ [77], có tên gọi là P-FHM+ [45] để khai thác HUI kết hợp với ràng buộc về độ dài tập. Các giải thuật phân tán này đều có chung một số đặc điểm: chúng là các giải thuật ở dạng xấp xỉ; quá trình khai thác diễn ra phân tán trên các nút của mạng lưới tính toán, tuy nhiên tại các nút này, quá trình khai thác là diễn ra dưới dạng áp dụng giải thuật tuần tự.

pEFIM [48] và MCH-Miner [48] lần lượt mở rộng các giải thuật EFIM và iMEFM thông qua kỹ thuật xử lý đa nhân để tăng tốc quá trình khai thác. Áp dụng kỹ thuật tương tự, Tung và đồng sự đề xuất một phiên bản song song hoá của giải thuật CLH-

Miner [74], có tên gọi pCLH-Miner [49]. Với tiếp cận này, tất cả các nhân tính toán của vi xử lý đều được tận dụng để giảm thời gian khai thác.

Hầu như các kỹ thuật khai thác song song nói trên đều chỉ áp dụng chiến lược điều phối đơn giản là "*đến trước, phục vụ trước*" để tiến hành khai thác, dẫn đến thời gian chờ giữa các nhân xử lý không tối ưu, và không hiệu quả trong việc rút ngắn hơn nữa thời gian khai thác. Ngoài ra, các giai đoạn của giải thuật cũng chưa được xem xét để song song hoá một cách triệt để, nhằm cải thiện hiệu năng, mà chỉ tập trung chủ yếu vào giai đoạn khai thác chính.

Tóm tắt các giải thuật khai thác tập mục hữu ích cao sử dụng mô hình xử lý song song hoặc tính toán phân tán cùng với các đặc điểm và năm công bố của các giải thuật này được cho tại Bảng 1.6.

Bảng 1.6. Các giải thuật khai thác tập mục hữu ích cao song song, phân tán

Giải thuật	Mô tả	Đặc điểm	Năm
DTWU-Mining	Sử dụng cấu trúc cây WIT-Tree và diffset để tăng hiệu năng khai thác HUI trên các tập dữ liệu dọc phân tán [5].	Có khả năng khai thác các tập dữ liệu được lưu trữ phân tán dọc.	2009
PHUI-Miner	Phiên bản xấp xỉ song song hóa HUI-Miner [45].	Khai thác HUI từ tập dữ liệu được lưu trữ phân tán với Spark.	2016
P-FHM+	Phiên bản song song của giải thuật FHM+ [45].	Cho hiệu suất khai thác cao hơn so với FHM+. Phân phối cân bằng tải chưa hiệu quả.	2018

Giải thuật	Mô tả	Đặc điểm	Năm
pEFIM	Mở rộng giải thuật EFIM sử dụng kỹ thuật song song hóa đa nhân [48].	Giảm thiểu đáng kể thời gian khai thác bằng cách tận dụng cách phân chia không gian tìm kiếm cho các nhân xử lý.	2019
MCH-Miner	Mở rộng giải thuật iMEFIM sử dụng kỹ thuật song song hóa đa nhân để khai thác các tập HUI từ những tập dữ liệu có lợi ích thay đổi [48].	Giảm thiểu đáng kể thời gian khai thác bằng cách tận dụng cách phân chia không gian tìm kiếm cho các nhân xử lý.	2020
pCLH-Miner	Mở rộng giải thuật CLH-Miner sử dụng kỹ thuật song song hoá đa nhân để khai thác các tập mục hữu ích cao liên cấp [49].	Rút ngắn thời gian khai thác của giải thuật CLH-Miner bằng song song hoá theo chiều sâu cấu trúc taxonomy.	2021

1.2.5 Các bài toán khai thác tập mục hữu ích cao khác

Bên cạnh các công trình tập trung vào bài toán khai thác tập mục hữu ích cao đã khảo sát nói trên, hàng loạt các biến thể của bài toán HUIM cũng đã được công bố trong thời gian qua. Một số hướng nghiên cứu, công trình đáng lưu ý như sau:

Các giải thuật như REPT [78], TKU/TKO [79], THUI [80], mlTKO [81], FTKHUIM [82] lần lượt được đề xuất để giải quyết bài toán khai thác top-k tập mục hữu ích cao.

Hướng tiếp cận áp dụng các mô hình meta-heuristic như giải thuật di truyền (genetic algorithm), tối ưu hoá bầy đàn (particle swarm optimization), v.v… vào bài toán HUIM cũng được xem xét và đề xuất. Một số giải thuật điển hình như HUPE$_{UMU}$-

GARM/HUPE$_{WUMU}$-GARM [83], HUIM-BPSOsig [84], Bio-HUIF-GA/Bio-HUIF-PSO/Bio-HUIF-BA [85], HUIM-ACO [86], ML-TKCE [87].

Trong tất cả các công trình liên quan đã khảo sát, ta có thể rút ra một số nhận định sau để định hướng cho quá trình nghiên cứu của luận án:

- Các công trình chủ yếu tập trung giải quyết vấn đề cải thiện hiệu năng khai thác dựa trên các chiến lược tỉa ứng viên để giảm thời gian và không gian của bài toán.
- Các tiếp cận sử dụng mô hình xử lý song song đa nhân chỉ áp dụng ở mức sơ khởi, với các chiến lược điều phối đơn giản. Đối với các tiếp cận song song hoá ở dạng phân tán, giải thuật được áp dụng tại các điểm phân tán đơn thuần là các giải thuật tuần tự. Ngoài ra, các tiếp cận phân tán đều cho kết quả khai thác ở dạng xấp xỉ.
- Đối với định dạng CSDL có chứa sự phân cấp các hạng mục, đây là một bài toán mới được đề xuất vào năm 2017. Bài toán này cũng được áp dụng trong các CSDL thực tế. Tuy nhiên, do bài toán có không gian tìm kiếm rất lớn, độ phức tạp cao, số lượng tập mục kết quả tìm được cũng gia tăng. Vì vậy, chưa có nhiều các nghiên cứu để giải quyết hiệu quả bài toán này cho đến nay.

Với các nhận định trên, luận án tập trung vào ba nội dung nghiên cứu chính đã trình bày trong phần MỞ ĐẦU. Các hướng nghiên cứu này là các hướng nghiên cứu mới và chưa có công trình nào được công bố hoặc đã có các công bố nhưng chưa giải quyết hiệu quả, triệt để các bài toán đã nêu. Vì vậy, các Chương 2, 3 và 4 sẽ tập trung khảo sát, làm rõ các bài toán cần giải quyết và đề xuất các phương pháp, giải thuật để giải quyết chúng một cách hiệu quả.

CHƯƠNG 2 KHAI THÁC HIỆU QUẢ TẬP MỤC HỮU ÍCH CAO ĐÓNG TỪ CƠ SỞ DỮ LIỆU CÓ ĐỘ HỮU ÍCH BIẾN ĐỘNG

2.1 Giới thiệu bài toán

Như đã giới thiệu ở phần 1.1, bài toán khai thác các tập mục hữu ích cao [13], [14], [55] là một mở rộng của bài toán khai thác tập phổ biến (FIM) [1]. Mục đích của bài toán là khảo sát một CSDL giao dịch để xác định các tập mục mang lại độ hữu ích thỏa một ngưỡng độ hữu ích tối thiểu *minutil* cho trước. Các tập mục thỏa ngưỡng *minutil* được gọi là tập mục hữu ích cao.

Với bài toán mới này, CSDL sẽ được mở rộng để bổ sung các thông tin vào các giao dịch, bao gồm các thông tin liên quan như số lượng mặt hàng đã được chọn mua (độ hữu ích nội) cũng như đơn giá (độ hữu ích ngoại) của chúng.

Bảng 2.1. Ví dụ về CSDL giao dịch có độ hữu ích tĩnh

TID	**Các hạng mục**	**Độ hữu ích nội**
T_1	b, c, d, g	$1, 2, 1, 1$
T_2	a, b, c, d, e	$4, 1, 3, 1, 1$
T_3	a, c, d	$4, 2, 1$
T_4	a, b, d, e	$5, 2, 1, 2$
T_5	a, b, c, f	$3, 4, 1, 2$

Bảng 2.2. Độ hữu ích ngoại của các hạng mục cho tại Bảng 2.1

Item	a	b	c	d	e	f	g
Độ hữu ích ngoại	1	2	1	5	4	3	1

Một ví dụ về CSDL giao dịch được cho tại Bảng 2.1. Đây là một ví dụ về các CSDL thường được sử dụng trong các cửa hàng bán lẻ, siêu thị, v.v… trong thực tế tại Việt Nam. Khi một khách hàng chọn mua các mặt hàng tại một cửa hàng bán lẻ, một giao dịch tương ứng sẽ được phát sinh, chứa danh sách các mặt hàng cùng với số lượng

các mặt hàng đã được chọn mua. Bên cạnh đó, đơn giá các mặt hàng này cũng được lưu trữ trong CSDL giao dịch tương ứng để giúp tính được giá trị doanh thu của các mặt hàng đó. Để đơn giản trong trình bày, các mặt hàng thể hiện từ Bảng 2.1 và các CSDL xuyên suốt luận án sẽ được đại diện dưới dạng các ký hiệu $a, b, c,$ v.v… Các mặt hàng được thống nhất sử dụng thông qua thuật ngữ là *hạng mục*.

CSDL giao dịch được cho tại Bảng 2.1 gồm có 5 giao dịch. Trong đó, mỗi giao dịch có một định danh duy nhất (TID) trong toàn bộ CSDL. Ứng với mỗi giao dịch, ta có các thông tin sau: các hạng mục cùng với số lượng đã được giao dịch (*độ hữu ích nội*). *Đơn giá*, *độ hữu ích đơn vị* hay còn gọi là *độ hữu ích ngoại* của các hạng mục này được cho tại Bảng 2.2. Giả sử, độ hữu ích của hạng mục a trong giao dịch T_2 có thể được xác định bằng cách lấy số lượng giao dịch của a tại T_2 là 4 nhân với đơn giá của a là \$1, ta được doanh thu khi bán hạng mục a là \$4. Tương tự với các hạng mục còn lại trong CSDL.

Bài toán HUIM đi tìm các kết hợp hạng mục mang lại độ hữu ích cao trong CSDL khi chúng được kết hợp cùng nhau. Người dùng sẽ được yêu cầu cung cấp một giá trị được gọi là ngưỡng độ hữu ích tối thiểu $minutil$, khi đó các giải thuật HUIM sẽ kết xuất ra tất cả các tập mục có độ hữu ích thoả ngưỡng $minutil$ đã cho.

Kể từ khi lớp bài toán khai thác tập mục hữu ích cao được đề xuất, đã có khá nhiều giải thuật được giới thiệu để nhằm giải quyết lớp bài toán này một cách hiệu quả. Tuy nhiên, tất cả các giải thuật nói trên đều đặt trên một giả định phi thực tế là độ hữu ích ngoại của các hạng mục không thay đổi theo thời gian. Xét ví dụ tại Bảng 2.1 và Bảng 2.2, ta nhận thấy độ hữu ích ngoại của hạng mục $\{a\}$ chẳng hạn là \$1 trong suốt tất cả các giao dịch. Đối với các CSDL giao dịch trong thực tế thì giá trị này là biến đổi, ví dụ như khi nguồn cung của mặt hàng đó khan hiếm làm cho giá của mặt hàng đó bị tăng, hoặc giảm xuống do thực hiện khuyến mãi chẳng hạn. Bỏ qua yếu tố quan trọng này sẽ dẫn đến các giải thuật khai thác HUI không thể áp dụng vào CSDL thực tế, vì tạo ra các kết quả không chính xác. Để giải quyết hạn chế nói trên, CSDL cần

được mở rộng để chứa thông tin về giá trị độ hữu ích ngoại của từng hạng mục tại mỗi giao dịch.

Ví dụ Bảng 2.3, cho thấy một CSDL giao dịch có độ hữu ích các hạng mục thay đổi và được thể hiện trong mỗi giao dịch. Độ hữu ích của hạng mục a, c và d trong giao dịch T_2 lần lượt là $\$1.0, \$0.9$ và $\$4.8$ tương ứng, giá trị này có thể thay đổi trong các giao dịch khác như ở T_5, độ hữu ích của a bây giờ là $\$1.2$. Với bài toán được đặt ra như trên, các giải thuật ban đầu áp dụng trong các bài toán HUIM hầu như sẽ bỏ qua các thông tin về độ hữu ích biến đổi này và sẽ cho các kết quả không chính xác. Vì vậy, ta cần định nghĩa lại phương pháp để xác định một cách đúng đắn độ hữu ích của các hạng mục bên trong CSDL giao dịch có độ hữu ích biến đổi này.

Bảng 2.3. CSDL mở rộng để biểu diễn sự thay đổi độ hữu ích của các hạng mục

TID	**Các hạng mục**	**Độ hữu ích nội**	**Độ hữu ích**
T_1	b, c, d, g	$1, 2, 1, 1$	$2, 1, 5, 1$
T_2	a, b, c, d, e	$4, 1, 3, 1, 1$	$1, 1.9, 0.9, 4.8, 4$
T_3	a, c, d	$4, 2, 1$	$1.1, 1.1, 5.5$
T_4	a, b, d, e	$5, 2, 1, 2$	$1.1, 2.2, 5.5, 4.4$
T_5	a, b, c, f	$3, 4, 1, 2$	$1.2, 2.4, 1.2, 3.6$

Đặc điểm của bài toán khai thác mẫu nói chung và HUIM nói riêng là khai thác một lượng lớn các tập mục thỏa điều kiện. Đặc biệt khi giá trị ngưỡng khai thác được cho thấp. Điều này làm tăng chi phí về mặt thời gian và không gian lưu trữ để phân tích các kết quả thu được này. Nhằm thu gọn tập kết quả nhưng vẫn bảo toàn lượng tri thức đã khai thác được, một biểu diễn cô đọng của tập mục HUI được đề xuất để sử dụng trong quá trình khai thác. Biểu diễn cô đọng này được gọi là biểu diễn tập mục hữu ích cao đóng [70].

Năm 2016, Fournier-Viger và đồng sự giới thiệu một giải thuật để giải quyết bài toán khai thác các tập mục hữu ích cao đóng trên CSDL giao dịch thông thường [18]. Giải thuật này có tên gọi là EFIM-Closed. Đây là một giải thuật một pha và được đánh giá là khá hiệu quả để khai thác các các tập mục hữu ích cao đóng. Giải thuật này được thiết kế sử dụng các ràng buộc chặt chẽ như sau: với mỗi tập mục có trong không gian tìm kiếm, tất cả các thao tác trên tập mục này phải được thực hiện trong thời gian và không gian tuyến tính.

Tuy nhiên, để áp dụng giải thuật này lên trên CSDL có độ hữu ích thay đổi, ta cần có một kỹ thuật khác để xác định chính xác độ hữu ích động của các hạng mục. Tuy giải thuật EFIM-Closed được đề xuất với hàng loạt các kỹ thuật tỉa ứng viên hiệu quả, các chiến lược giảm thiểu chi phí quét CSDL có mật độ cao, giải thuật này vẫn cần khá nhiều thời gian để khai thác trên các CSDL có mật độ thấp.

Mục đích của nội dung nghiên cứu này là đề xuất một giải thuật cải tiến có tên gọi iEFIM-Closed dựa trên EFIM-Closed với các đặc điểm như sau:

- Là giải thuật một pha.
- Kế thừa các chiến lược tỉa ứng viên hiệu quả của EFIM-Closed [18].
- Kế thừa các kỹ thuật giảm chi phí quét CSDL và kiểm tra nhanh tính đóng các tập mục của EFIM-Closed [18].
- Áp dụng **Định nghĩa 2.6** để triển khai giải thuật trên các CSDL có độ hữu ích thay đổi [17].
- Giảm thiểu hơn nữa chi phí quét CSDL có mật độ thấp để cải thiện hiệu năng toàn phần giải thuật thông qua việc áp dụng cấu trúc P-set [17].

2.2 Một số định nghĩa cơ sở

Định nghĩa 2.1 (CSDL giao dịch). Cho $\mathbb{I}$ là một tập hữu hạn gồm n hạng mục, $\mathbb{I} = \{i_1, i_2, \cdots, i_n\}$. Một tập mục $\mathcal{X}$ là một tập hữu hạn các hạng mục sao cho $\mathcal{X} \subseteq \mathbb{I}$. $\mathcal{X}$ được gọi là một k-itemset, với $k = |\mathcal{X}|$. Một CSDL giao dịch là tập các giao dịch, được định nghĩa như sau:

$$\mathbb{D} = \{T_1, T_2, \cdots, T_m\} \tag{2.1}$$

Trong đó, $m = |\mathbb{D}|$. Mỗi giao dịch T_c, $T_c \subseteq \mathbb{I}$, T_c có một định danh giao dịch duy nhất c, gọi là TID (Transaction ID), $1 \leq c \leq m$. Mỗi hạng mục $i \in \mathbb{I} \wedge i \in T_c$, hạng mục i này được kết hợp với một giá trị dương $eu(i)$, ta gọi là độ hữu ích ngoại của i. Mỗi hạng mục i xuất hiện trong giao dịch T_c có một giá trị dương $\text{iu}(i, T_c)$ gọi là độ hữu ích nội của i [13].

Một ví dụ về CSDL giao dịch, các giao dịch, hạng mục, các đại lượng hữu ích được cho tại Bảng 1.1 và Bảng 1.2. Trong đó, các hạng mục được ký hiệu lần lượt là a, b, c v.v… để thuận tiện trong việc trình bày.

Định nghĩa 2.2 (Độ hữu ích của một hạng mục). Cho hạng mục $i \in \mathbb{I}$ xuất hiện trong giao dịch $T_q \in \mathbb{D}$, độ hữu ích của i tại giao dịch T_q được xác định như sau [13]:

$$u(i, T_q) = iu(i, T_q) \times eu(i) \tag{2.2}$$

Định nghĩa 2.3 (Độ hữu ích của một tập mục trong một giao dịch). Cho $\mathcal{X}$ là một tập mục xuất hiện trong giao dịch T_q $(\mathcal{X} \subseteq \mathbb{I}, T_q \in \mathbb{D})$, độ hữu ích của $\mathcal{X}$ trong T_q, ký hiệu $u(\mathcal{X}, T_q)$, được xác định như sau [13]:

$$u(\mathcal{X}, T_q) = \sum_{i \in \mathcal{X} \wedge \mathcal{X} \subseteq T_q} u(i, T_q) \tag{2.3}$$

Độ hữu ích của $\mathcal{X}$ trong toàn CSDL $\mathbb{D}$, ký hiệu $u(\mathcal{X})$, được xác định như sau [13].

$$u(\mathcal{X}) = \sum_{T_i \in TidSet(\mathcal{X})} u(\mathcal{X}, T_i) \tag{2.4}$$

Trong đó, tập tất cả các giao dịch có chứa tập mục $\mathcal{X}$ được ký hiệu là $TidSet(\mathcal{X})$.

Định nghĩa 2.4 (Độ hữu ích của một giao dịch). Độ hữu ích của một giao dịch T_q trong CSDL $\mathbb{D}$, ký hiệu $TU(T_q)$ được định nghĩa như sau [14].

$$TU(T_q) = \sum_{i \in T_q} u(i, T_q) \tag{2.5}$$

Định nghĩa 2.5 (Tập mục hữu ích cao). Cho CSDL giao dịch $\mathbb{D}$, một ngưỡng độ hữu ích tối thiểu $minutil$, và tập mục $\mathcal{X}$, $\mathcal{X} \subseteq \mathbb{I}$. $\mathcal{X}$ là một tập mục hữu ích cao nếu và chỉ nếu $u(\mathcal{X}) \geq minutil$.

Tập đầy đủ các tập mục hữu ích cao trong CSDL $\mathbb{D}$ được ký hiệu là $\mathbb{H}$.

Các định nghĩa cơ bản vừa trình bày chỉ áp dụng đối với các bài toán khai thác tập mục hữu ích cao, tập mục hữu ích cao đóng thuần túy. Nếu các định nghĩa này được áp dụng lên các CSDL giao dịch có giá trị độ hữu ích các hạng mục thay đổi, các định nghĩa trên sẽ không còn mô tả chính xác bài toán. Điều này có thể đưa đến các kết quả khai thác không chính xác. Tuy nhiên, trong các định nghĩa nêu trên, ta nhận thấy cốt lõi của bài toán khai thác tập mục hữu ích cao, hữu ích cao đóng đều dựa vào **Định nghĩa 2.2**. Vì vậy, ta cần định nghĩa lại cách tính độ hữu ích mới trên CSDL có độ hữu ích biến đổi như sau:

Định nghĩa 2.6 (Độ hữu ích một hạng mục trong CSDL có độ hữu ích biến đổi). Độ hữu ích của hạng mục $i \in \mathbb{I}$ trong giao dịch T_c, ký hiệu $u(i, T_c)$ được định nghĩa lại là tích của $iu(i, T_c)$ và $eu(i, T_c)$. Với $eu(i, T_c)$ là giá trị độ hữu ích của hạng mục i trong giao dịch T_c.

$$u(i, T_c) = iu(i, T_c) \times eu(i, T_c) \tag{2.6}$$

Điểm khác biệt chính giữa **Định nghĩa 2.2** và **Định nghĩa 2.6** là cách xác định độ hữu ích ngoại cho hạng mục i: giá trị $eu(i)$ và $eu(i, T_c)$. Đối với bài toán HUIM thông thường trên CSDL có độ hữu ích tĩnh, $eu(i)$ là tĩnh trong toàn bộ CSDL và cả quá trình khai thác.

Tuy nhiên, như đã trình bày, giả định này là không thực tế. Vì vậy, ta cần có một cách tính khác phản ánh đúng đắn hơn về độ hữu ích của hạng mục. **Định nghĩa 2.6** giải quyết vấn đề này bằng cách cho phép độ hữu ích của hạng mục i có khả năng biến động ở từng giao dịch có chứa i thông qua $eu(i, T_c)$.

Để giảm thiểu không gian lưu trữ dữ liệu và tăng tốc quá trình tính toán cũng như xử lý của các giải thuật trên CSDL có độ hữu ích thay đổi này, ta thực hiện việc chuyển đổi dạng CSDL có độ hữu ích thay đổi ban đầu sang một định dạng thu gọn hơn.

Với định dạng CSDL mới này, ta chỉ cần lưu trữ giá trị $u(i, T_c)$ của mỗi hạng mục i thuộc giao dịch T_c thay vì lưu trữ cả số lượng hạng mục i được mua và cả độ hữu ích

đơn vị của hạng mục đó. Ví dụ, với CSDL có độ hữu ích thay đổi cho ở Bảng 2.3 ban đầu, ta chuyển đổi CSDL đó trở thành CSDL thu gọn như ở bảng Bảng 2.4.

Bảng 2.4. CSDL độ hữu ích biến đổi thu gọn

TID	**Các hạng mục**	**Độ hữu ích**
T_1	b, c, d, g	2, 2, 5, 1
T_2	a, b, c, d, e	4, 1.9, 2.7, 4.8, 4
T_3	a, c, d	4.4, 2.2, 5.5
T_4	a, b, d, e	5.5, 4.4, 5.5, 8.8
T_5	a, b, c, f	3.6, 9.6, 1.2, 7.2

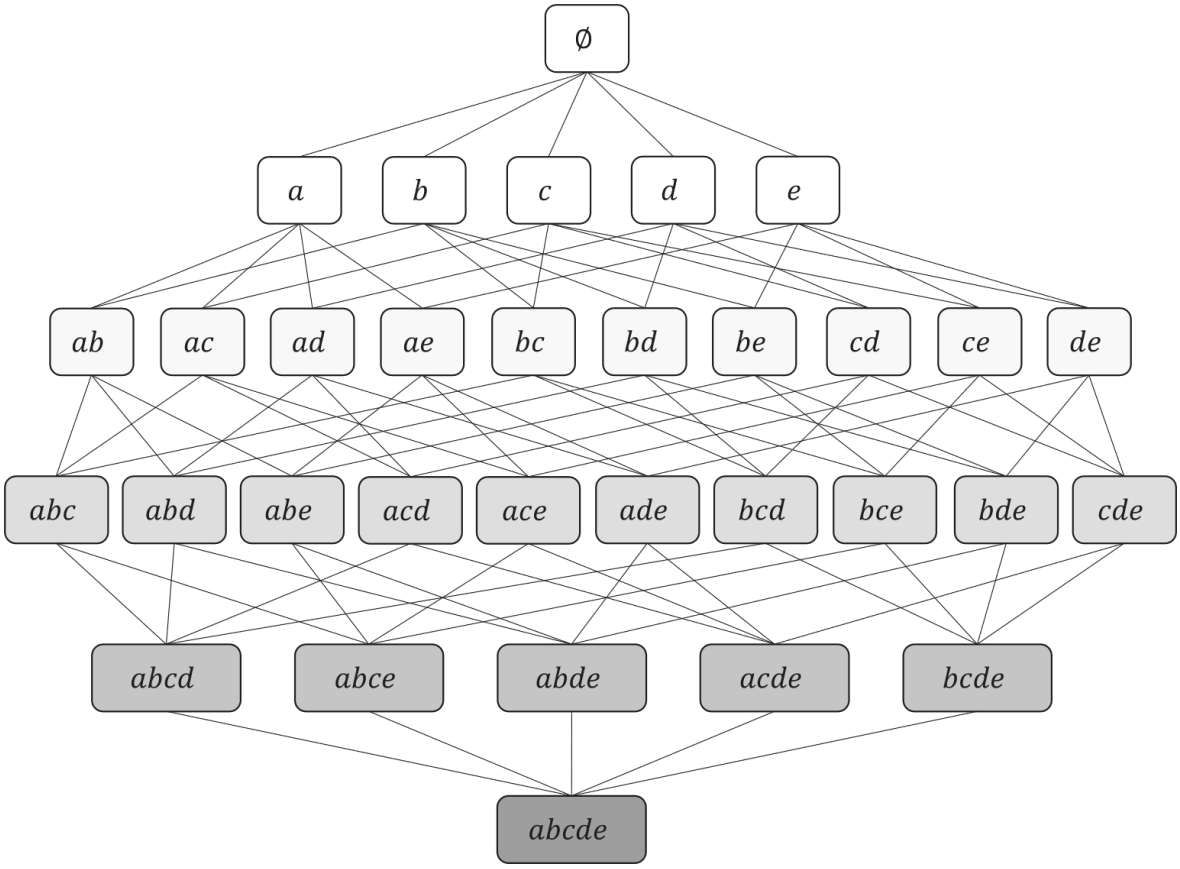
Bài toán khai thác mẫu nói chung và bài toán khai thác tập mục hữu ích cao nói riêng có không gian tìm kiếm rất lớn, phụ thuộc hoàn toàn vào số lượng các hạng mục thuộc tập $\mathbb{I}$. Giả sử ta có tập $\mathbb{I} = \{a, b, c, d, e\}$, Hình 2.1 mô tả một dàn liệt kê các kết hợp giữa các hạng mục thuộc tập $\mathbb{I}$ này.

Qua đó, ta thấy được rằng chỉ với $|\mathbb{I}| = 5$ hạng mục, số lượng các kết hợp cần phải xét là rất lớn. Trong thực tế khai thác, $|\mathbb{I}|$ có thể lên đến hàng ngàn, hoặc hàng chục ngàn hạng mục sẽ rất dễ dẫn đến sự bùng nổ tổ hợp khi số lượng tập hợp cần xét là $2^n - 1$, với $n = |\mathbb{I}|$.

Đối với bài toán HUIM, không phải kết hợp nào cũng có thể trở thành các tập mục hữu ích cao. Đa phần các kết hợp sẽ là các tập mục không có tiềm năng trở thành tập mục hữu ích cao. Vì vậy, nếu ta cần phải xét hết tất cả các kết hợp này bằng chiến lược *vét cạn*, các giải thuật khai thác sẽ trở nên không hiệu quả và hoàn toàn không khả thi trong thực tế.

Đối với bài toán FIM, các nghiên cứu đều tận dụng một tính chất mạnh là tính chất Apriori để thu nhỏ không gian tìm kiếm của bài toán [50]. Tuy nhiên, đối với bài toán

HUIM, độ hữu ích các hạng mục lại không thoả mãn tính chất này [14]. Vì vậy, các tiếp cận khai thác không thể tái sử dụng các chiến lược tỉa ứng viên hiệu quả từ bài toán FIM để áp dụng vào bài toán HUIM. Điều này đưa đến việc cần phải tìm kiếm một ngưỡng giới hạn trên thoả DCP để có thể loại bỏ một cách an toàn các tập mục không tiềm năng và thu gọn không gian tìm kiếm.



Hình 2.1. Không gian tìm kiếm cho tập $\mathbb{I} = \{a, b, c, d, e\}$

Định nghĩa 2.7 (Trọng số hữu ích giao dịch – TWU). Cho $\mathcal{X}$ là một tập mục trong CSDL $\mathbb{D}$, $\mathcal{X} \subseteq \mathbb{I}$. Trọng số hữu ích giao dịch của $\mathcal{X}$ được xác định như sau [14]:

$$TWU(\mathcal{X}) = \sum_{\mathcal{X} \subseteq T_q \wedge T_q \in \mathbb{D}} TU(T_q) \quad (2.7)$$

TWU được đề xuất bởi Liu và đồng sự cùng với giải thuật Two-Phase [14]. TWU được chứng minh là một ngưỡng giới hạn trên đúng đắn và có thể sử dụng để tỉa an toàn các tập mục không có khả năng trở thành tập mục hữu ích cao.

Tính chất 2.1 (Ước lượng độ hữu ích sử dụng dụng TWU). Cho $\mathcal{X}$ là một tập mục, TWU của $\mathcal{X}$ không bé hơn độ hữu ích của $\mathcal{X}$: $TWU(\mathcal{X}) \geq u(\mathcal{X})$. Hơn nữa, TWU của $\mathcal{X}$ cũng không bé hơn độ hữu ích tập cha của $\mathcal{X}$: $TWU(\mathcal{X}) \geq u(\mathcal{Y}), \forall \mathcal{Y} \supset \mathcal{X}$.

Tính đúng đắn của **Tính chất 2.1** đã được chứng minh tại [14]. Vì vậy, do TWU của $\mathcal{X}$ được tạo thành từ tổng độ hữu ích của các giao dịch có chứa $\mathcal{X}$, $TWU(\mathcal{X})$ phải lớn hơn hay bằng độ hữu ích của $\mathcal{X}$ và bất kỳ tập cha nào của $\mathcal{X}$.

Tính chất 2.2. (Tỉa ứng viên sử dụng TWU). Cho $\mathcal{X}$ là một tập mục trong CSDL $\mathbb{D}$, $\mathcal{X} \subseteq \mathbb{I}$, một ngưỡng độ hữu ích tối thiểu $minutil$ cho trước. $\mathcal{X}$ và các mở rộng của $\mathcal{X}$ là các tập mục không tiềm năng (tập mục hữu ích thấp) nếu như $TWU(\mathcal{X}) < minutil$.

Tính chất 2.2 là hệ quả của **Tính chất 2.1**. [14]

Khi áp dụng vào quá trình quét không gian tìm kiếm, **Tính chất 2.2** có thể loại bỏ một cách an toàn các tập mục không tiềm năng khỏi không gian tìm kiếm, và giúp tăng hiệu năng của quá trình khai thác.

Định nghĩa 2.8 (Không gian tìm kiếm của bài toán). Cho $\succ$ là một thứ tự toàn phần các hạng mục từ $\mathbb{I}$. Dựa trên thứ tự này, không gian tìm kiếm của tất cả các tập mục $2^{|\mathbb{I}|}$ có thể được biểu diễn dưới dạng dàn. Lấy ví dụ, không gian tìm kiếm của tập hợp $\mathbb{I} = \{a, b, c, d, e\}$ dựa trên thứ tự từ điển được cho tại Hình 2.1. Giải thuật EFIM-Closed quét qua không gian tìm kiếm này theo chiến lược tìm theo chiều sâu và xuất phát từ gốc của dàn này (tập rỗng). Trong quá trình quét theo chiều sâu, và một tập mục α bất kỳ, EFIM-Closed đệ quy quá trình ghép mỗi lần một hạng mục vào α dựa trên thứ tự $\succ$ để tạo ra một tập mục lớn hơn. Cụ thể, EFIM-Closed xem thứ tự $\succ$ là thứ tự tăng dần của TWU bởi vì điều này về tổng quát có thể giúp thu nhỏ không gian tìm kiếm các HUI [14], [16], [58]. Tuy nhiên, để đơn giản trong trình bày, ta sử dụng thứ tự $\succ$ là thứ tự từ điển.

Cho tập mục $\mathcal{X}$. Gọi $E(\mathcal{X})$ là tập tất cả các hạng mục có thể sử dụng để mở rộng $\mathcal{X}$ dựa trên quá trình tìm kiếm theo chiều sâu dựa trên cây liệt kê tập hợp.

$$E(\mathcal{X}) = \{z \mid z \in \mathbb{I} \wedge z \succ x, \forall x \in \mathcal{X}\} \quad (2.8)$$

Một tập mục $\mathcal{Z}$ là một mở rộng của α (xuất hiện trong cây con của $\mathcal{X}$ trong cây liệt kê tập hợp) nếu $\mathcal{Z} = \mathcal{X} \cup \mathcal{W}$ với một tập mục $\mathcal{W} \in 2^{E(\mathcal{X})}$ và $\mathcal{W} \neq \emptyset$. Một tập mục $\mathcal{Z}$ là một mở rộng đơn hạng mục của $\mathcal{X}$ (là con của $\mathcal{X}$ trong cây liệt kê tập hợp) nếu $\mathcal{Z} = \mathcal{X} \cup \{z\}$ với hạng mục $z \in E(\mathcal{X})$.

Định nghĩa 2.9 (High-utility Database Projection – HDP). Phép chiếu một giao dịch T sử dụng tập mục $\mathcal{X}$, ký hiệu $T_{\mathcal{X}}$ được định nghĩa như sau: $T_{\mathcal{X}} = \{i \mid i \in T \wedge i \in E(\mathcal{X})\}$. Phép chiếu CSDL $\mathbb{D}$ sử dụng tập mục $\mathcal{X}$, ký hiệu $\mathbb{D}_{\mathcal{X}}$ được định nghĩa như sau [58].

$$\mathbb{D}_{\mathcal{X}} = \{T_{\mathcal{X}} \mid T \in \mathbb{D} \wedge T_{\mathcal{X}} \neq \emptyset\} \quad (2.9)$$

Để giảm thiểu hơn nữa chi phí quét CSDL, giải thuật EFIM-Closed đề xuất một kỹ thuật hiệu quả nhằm ghép các giao dịch lại với nhau. Kỹ thuật này dựa trên các quan sát rằng trong một CSDL giao dịch thường sẽ có chứa các giao dịch tương tự nhau (các giao dịch có cùng số lượng hạng mục nhưng không cần thiết phải có cùng các giá trị độ hữu ích nội). Ta thay thế các giao dịch này bằng một giao dịch mới dựa trên định nghĩa sau.

Định nghĩa 2.10 (High-utility Transaction Merging – HTM). Xét tập các giao dịch tương tự nhau $T_{r_1}, T_{r_2}, \ldots T_{r_m}$ trong một CSDL chiếu $\mathcal{X}\mathbb{D}$, ta thay các giao dịch này bằng một giao dịch mới $T_M = T_{r_1} = T_{r_2} = \cdots = T_{r_m}$ với số lượng của các hạng mục $i \in T_M$ được định nghĩa là $q(i, T_M)$ [58].

$$q(i, T_M) = \sum_{k=1\ldots m} q\left(i, T_{r_k}\right) \quad (2.10)$$

Định nghĩa 2.11 (sub-tree utility). Cho tập mục $\mathcal{X}$ và một hạng mục z với z có thể mở rộng $\mathcal{X}$ dựa trên quá trình quét theo chiều sâu, $z \in E(\mathcal{X})$. Ngưỡng giới hạn trên sub-tree utility của z ứng với $\mathcal{X}$ được định nghĩa là $su(\mathcal{X}, z)$ [58].

$$su(\mathcal{X}, z) = \sum_{T \in g(\mathcal{X} \cup \{z\})} [u(\mathcal{X}, T) + u(z, T) + \sum_{i \in T \wedge E(\mathcal{X} \cup \{z\})} u(i, T)] \quad (2.11)$$

Tính chất 2.3 (Tỉa ứng viên sử dụng sub-tree utility). Cho tập mục $\mathcal{X}$ và một hạng mục $z \in E(\mathcal{X})$. Nếu $su(\mathcal{X}, z) < minutil$, khi đó mở rộng đơn hạng mục $\mathcal{X} \cup \{z\}$ và các mở rộng của nó là các tập độ hữu ích thấp và có thể loại bỏ khỏi không gian tìm kiếm. Nói cách khác, cây con $\mathcal{X} \cup \{z\}$ trong cây liệt kê tập hợp có thể được tỉa khỏi không gian tìm kiếm [58].

Tính đúng đắn của **Tính chất 2.3** đã được chứng minh tại công trình [58].

Định nghĩa 2.12 (Độ hữu ích cục bộ). Cho một tập mục $\mathcal{X}$ và hạng mục $z \in E(\mathcal{X})$. Khi đó, độ hữu ích cục bộ của z ứng với $\mathcal{X}$ được xác định theo công thức sau [58].

$$lu(\mathcal{X}, z) = \sum_{T \in g(\mathcal{X} \cup \{z\})} [u(\mathcal{X}, T) + re(\mathcal{X}, T)] \quad (2.12)$$

Tính chất 2.4 (Tỉa ứng viên sử dụng độ hữu ích cục bộ). Cho tập mục $\mathcal{X}$ và một hạng mục $z \in E(\mathcal{X})$. Nếu $lu(\mathcal{X}, z) < minutil$, khi đó tất cả các mở rộng của $\mathcal{X}$ có chứa z đều là các tập hữu ích thấp và ta có thể loại bỏ chúng. Nói cách khác, hạng mục z có thể được bỏ qua khi ta quét qua tất cả cây con của $\mathcal{X}$ [58].

Tương tự, **Tính chất 2.4** đã được chứng minh tại công trình [58].

Định nghĩa 2.13 (tập Primary và Secondary). Với một tập mục $\mathcal{X}$ cho trước, ta gọi tập $Primary(\mathcal{X})$ là tập các hạng mục có sub-tree utility không bé hơn $minutil$, tập $Secondary(\mathcal{X})$ là tập các hạng mục có độ hữu ích cục bộ không nhỏ hơn $minutil$. Định nghĩa hình thức của tập $Primary$ và $Secondary$ như sau [58]:

$$Primary(\mathcal{X}) = \{z | z \in E(\mathcal{X}) \wedge su(\mathcal{X}, z) \geq minutil\} \quad (2.13)$$

$$Secondary(\mathcal{X}) = \{z | z \in E(\mathcal{X}) \wedge lu(\mathcal{X}, z) \geq minutil\} \quad (2.14)$$

Do $lu(\mathcal{X}, z) \geq su(\mathcal{X}, z)$, do đó $Primary(\mathcal{X}) \subseteq Secondary(\mathcal{X})$ [58].

Định nghĩa 2.14 (Tidset và Support). Độ hỗ trợ (support) của một tập mục $\mathcal{X}$ trong CSDL $\mathbb{D}$ chính là số lượng các giao dịch có chứa $\mathcal{X}$, ký hiệu $sup(\mathcal{X})$. Tidset của một tập mục $\mathcal{X}$ trong CSDL $\mathbb{D}$, ký hiệu $TidSet(\mathcal{X})$, là tập hợp TID của các giao dịch có chứa $\mathcal{X}$ [19].

$$sup(\mathcal{X}) = |TidSet(\mathcal{X})| \quad (2.15)$$

Định nghĩa 2.15 (Tập mục đóng). Một tập mục $\mathcal{X}$ được gọi là tập mục đóng nếu và chỉ nếu không tồn tại một tập mục $\mathcal{Y}$ trong CSDL $\mathbb{D}$ thoả $\mathcal{Y} \supset \mathcal{X}$ và $sup(\mathcal{X}) = sup(\mathcal{Y})$. Tập đầy đủ các tập mục đóng được ký hiệu là $\mathbb{C}$ [19].

Lưu ý, định nghĩa này áp dụng đối với cả các tập mục có chứa thông tin về độ hữu ích vì từ một tập mục đóng có chứa thông tin về độ hữu ích, ta cũng có thể suy ngược ra được các tập mục không đóng có chứa thông tin về độ hữu ích.

Định nghĩa 2.16 (Bao đóng của một tập mục). Gọi $\mathcal{Y}$ là một tập cha của $\mathcal{X}$, $\mathcal{Y}$ được gọi là bao đóng của $\mathcal{X}$ nếu và chỉ nếu $\mathcal{Y}$ đóng và $sup(\mathcal{Y}) = sup(\mathcal{X})$. Bao đóng của tập mục $\mathcal{X}$, ký hiệu $Closure(\mathcal{X})$, được định nghĩa như sau [19].

$$Closure(\mathcal{X}) = \cap_{r \in TidSet(\mathcal{X})} T_r \tag{2.16}$$

Định nghĩa 2.17 (Tập mục hữu ích cao, đóng). Một tập HUI $\mathcal{X}$ được gọi là tập hữu ích cao, đóng nếu $\mathcal{X}$ là tập mục hữu ích cao và $\mathcal{X}$ đóng [70].

Tương tự **Định nghĩa 2.15**, từ một tập mục hữu ích cao đóng, ta cũng có thể suy ngược ra được các tập mục hữu ích cao không đóng.

Định nghĩa 2.18 (Bài toán khai thác tập mục hữu ích cao, đóng). Khai thác các tập mục hữu ích cao, hữu ích cao đóng là bài toán đi tìm tất cả HUI/CHUI thỏa mãn giá trị $minutil$ cho trước. Ta gọi $\mathbb{CH}$ là tập đầy đủ các tập mục hữu ích cao đóng trong $\mathcal{D}$ thoả một ngưỡng $minutil$ cho trước, $\mathbb{CH} = \mathbb{C} \cap \mathbb{H}$ [19].

Định nghĩa 2.19 (Mở rộng tiến/mở rộng lùi – FCC/BCC). Cho một tập mục $\mathcal{Y} = \mathcal{X} \cup \{i\}$. Tập mục $\mathcal{Y}$ được nói là có mở rộng tiến nếu tồn tại một hạng mục $z \succ i$ thỏa $z \in E(\mathcal{Y})$ và $\sup(\mathcal{Y} \cup \{z\}) = \sup(\mathcal{Y})$. Tập mục $\mathcal{Y}$ được nói là có mở rộng lùi nếu tồn tại một hạng mục $z \prec i$ thỏa $z \notin \mathcal{Y}$ và $\sup(\mathcal{Y} \cup \{z\}) = \sup(\mathcal{Y})$ [18].

Tính chất 2.5 (Kiểm tra tập HUI đóng). Một tập mục $\mathcal{Y} = \mathcal{X} \cup \{i\}$ là một CHUI nếu $\mathcal{Y}$ là một HUI và $\mathcal{Y}$ không tồn tại mở rộng tiến và lùi. Theo định nghĩa, một tập mục là không đóng nếu tồn tại một tập cha $\mathcal{W} = \mathcal{Y} \cup \{z\}$ có cùng $support$. Hạng

mục bổ sung z có thể là $z \succ i$ hoặc $z \prec i$, và tương ứng với các trường hợp kiểm tra mở rộng tiến và lùi [18].

Tính chất 2.6 (Loại bỏ ứng viên dựa vào mở rộng lùi). Toàn bộ cây con của tập mục $\mathcal{Y} = \mathcal{X} \cup \{i\}$ có thể được loại bỏ khỏi quá trình tìm kiếm theo chiều sâu nếu $\mathcal{Y}$ có một mở rộng lùi. Nếu tồn tại một mở rộng lùi có chứa hạng mục z, z khi đó sẽ xuất hiện trong các giao dịch có $\mathcal{Y}$, dẫn đến tất cả các tập mục trong cây con của $\mathcal{Y}$ cũng sẽ có mở rộng lùi với z và vì vậy chúng đều không phải là CHUI [18].

Tính chất 2.7 (Nhảy bao đóng – CJU). Cho một tập mục $\mathcal{Y}$ và một CSDL chiếu $\mathcal{Y}\mathbb{D}$. Nếu $\sup(\mathcal{Y}) = \sup(\mathcal{Y} \cup \{z\})$ với mọi hạng mục $z \in E(\mathcal{Y})$, khi đó tập mục $\mathcal{Y} \cup E(\mathcal{Y})$ là tập mục đóng duy nhất trên cây con $\mathcal{Y}$. Khi đó, toàn bộ cây con của $\mathcal{Y}$ có thể được loại bỏ và tập $\mathcal{Y} \cup E(\mathcal{Y})$ có thể được trả về nếu như nó là một HUI [18].

Tính đúng đắn của các **Tính chất 2.5**, **Tính chất 2.6** và **Tính chất 2.7** đã được lần lượt chứng minh ở công trình [18].

2.3 Cải thiện hiệu năng quét CSDL

Dựa trên các quan sát đối với giải thuật EFIM-Closed, có thể nhận thấy rõ rằng một trong các thao tác tốn nhiều chi phí nhất của EFIM-Closed chính là việc phải quét đi quét lại CSDL nhiều lần. Việc quét CSDL này là cần thiết để có thể xác định được các độ hữu ích cục bộ, sub-tree utility, hữu ích của các tập mục và cả support của chúng.

Cho một tập mục $\mathcal{X}$, các giá trị vừa nêu cần phải được tính toán để có thể xác định được tập các hạng mục $Secondary(\mathcal{X})$, và các hạng mục thuộc tập này sẽ được sử dụng để mở rộng $\mathcal{X}$. Quá trình này có chi phí thực hiện rất lớn do giải thuật EFIM-Closed cần phải quét qua CSDL chiếu trên $\mathcal{X}$ là $\mathcal{X}\mathbb{D}$ mỗi khi cần phải tính các giá trị liên quan ứng với mỗi hạng mục $z \in Secondary(\mathcal{X})$ thay vì chỉ cần phải quét qua các giao dịch trong tập $\mathbb{D}_{\mathcal{X} \cup \{z\}}$.

Để khắc phục nhược điểm nêu trên nhằm chỉ cần phải quét qua các giao dịch thuộc tập $\mathbb{D}_{\mathcal{X}\cup\{z\}}$, một cấu trúc dữ liệu mới được đề xuất có tên gọi là tập P-set. Ta định nghĩa cấu trúc dữ liệu tập P-set mới này như sau:

Định nghĩa 2.20 (Phép chiếu TID của một tập mục trên CSDL). Cho CSDL giao dịch $\mathbb{D}$ và một tập mục $\mathcal{X}$. Phép chiếu TID của tập mục $\mathcal{X}$ lên CSDL $\mathbb{D}$, được gọi là tập P-set của $\mathcal{X}$, là tập định danh của các giao dịch có chứa $\mathcal{X}$. Khi đó:

$$P\text{-}set(\mathcal{X}) = \{T.id \mid T \in \mathbb{D} \wedge \mathcal{X} \subseteq T\} \quad (2.17)$$

Định nghĩa 2.21 (Mở rộng P-set của tập mục sử dụng hạng mục đơn). Cho CSDL giao dịch $\mathbb{D}$ và tập mục $\mathcal{X}$. Việc mở rộng phép chiếu TID của tập mục $\mathcal{X}$ trên $\mathbb{D}$ sử dụng hạng mục z, ký hiệu $P\text{-}set(\mathcal{X}, z)$ được định nghĩa là:

$$P\text{-}set(\mathcal{X}, z) = \{T'.id \mid T' \in \mathcal{X}\mathbb{D} \wedge z \in T'\} \quad (2.18)$$

2.4 Giải thuật iEFIM-Closed

Để tránh phải quét đi quét lại toàn bộ CSDL giao dịch nhiều lần trong quá trình tính toán các giá trị giới hạn trên, từ cấu trúc $P\text{-}set$ đã cho tại **Định nghĩa 2.20** và **2.21**, ta kết hợp với giải thuật EFIM-Closed ban đầu để có được một giải thuật mới và gọi là iEFIM-Closed (*improved EFIM-Closed).*

Mã giả của thủ tục chính và hàm đệ quy theo chiều sâu $Search()$ của giải thuật iEFIM-Closed được thể hiện tại Giải thuật 2.1 và Giải thuật 2.2.

Thay đổi cốt lõi của giải thuật chính là việc thực hiện tính toán cho $P\text{-}set(\mathcal{X}, i)$ xuất hiện tại dòng 7, 9 của giải thuật chính và dòng 3, 6, 11 của thủ tục $Search$. Do $P\text{-}set(\mathcal{X}, i)$ chứa tất cả các TID trong $\mathcal{X}\mathbb{D}$, độ phức tạp theo thời gian của việc tính toán $su(\mathcal{X}, i)$ không thay đổi so với giải thuật ban đầu.

Tương tự đối với hàm $Search$, hiệu quả của việc sử dụng cấu trúc dữ liệu $P\text{-}set(\mathcal{X}, i)$ được thể hiện tại dòng 3 của hàm $Search$. Tại đây, chỉ có các giao dịch có định danh giao dịch TID xuất hiện trong $P\text{-}set(\mathcal{X}, i)$ là được quét qua thay vì toàn bộ CSDL chiếu $\mathcal{X}\mathbb{D}$, từ đó giúp giảm thiểu rõ rệt chi phí cho việc quét qua CSDL chiếu $\mathcal{X}\mathbb{D}$.

Input: CSDL $\mathbb{D}$, ngưỡng $minutil$
Output: tập các tập mục hữu ích cao, đóng

1 $\mathcal{X} = \emptyset$
2 Tính $lu(\mathcal{X}, i)$ cho tất cả các hạng mục $i \in \mathbb{I}$ bằng cách quét CSDL $\mathbb{D}$.
3 $Secondary(\mathcal{X}) = \{i \mid i \in \mathbb{I} \wedge lu(\mathcal{X}, i) \geq minutil\}$
4 Gọi $\succ$ là thứ tự toàn phần không giảm của TWU trên $Secondary(\mathcal{X})$.
5 Quét $\mathbb{D}$ để loại bỏ $i \notin Secondary(\mathcal{X})$ khỏi các giao dịch, sắp xếp các hạng mục trong mỗi giao dịch theo thứ tự $\succ$ và loại bỏ các giao dịch rỗng.
6 Sắp thứ tự các giao dịch trong $\mathbb{D}$ dựa trên thứ tự $\succ$.
7 Tính $su(\mathcal{X}, i)$, $P\text{-}set(\mathcal{X}, i)$, $\forall i \in Secondary(\mathcal{X})$ bằng cách quét $\mathbb{D}$.
8 $Primary(\mathcal{X}) = \{i \mid i \in Secondary(\mathcal{X}) \wedge su(\mathcal{X}, i) \geq minutil\}$;
9 **RETURN** $Search(\mathcal{X}, \mathbb{D}, Primary(\mathcal{X}), Secondary(\mathcal{X}), minutil, P\text{-}set(\mathcal{X}, i))$;

Giải thuật 2.1. Giải thuật iEFIM-Closed.

Input: tập mục $\mathcal{X}$, CSDL chiếu $\mathbb{D}_{\mathcal{X}}$, tập primary $Primary(\mathcal{X})$, tập $Secondary(\mathcal{X})$, ngưỡng $minutil$, tập chiếu $P\text{-}set(\mathcal{X}, i)$ mở rộng.
Output: tập các tập mục hữu ích cao là mở rộng của $\mathcal{X}$.

1 **FOREACH** $i \in Primary(\mathcal{X})$ **DO**
2 $\mathcal{Y} = \mathcal{X} \cup \{i\}$;
3 Quét $\mathbb{D}_{\mathcal{X}}$ trong $P\text{-}set(\mathcal{X}, i)$ để tính $u(\mathcal{Y})$ và tạo ra $\mathbb{D}_{\mathcal{Y}}$; // gồm HTM.
4 **IF** $\mathcal{Y}$ không có mở rộng lùi **THEN**
5 Tính $sup(\mathcal{Y}, z), su(\mathcal{Y}, z), lu(\mathcal{Y}, z), P\text{-}set(\mathcal{Y}, z)$ với mỗi hạng mục $z \in Secondary(\mathcal{X})$ bằng cách quét $\mathbb{D}_{\mathcal{Y}}$.
6 **IF** $sup(\mathcal{Y}) = sup(\mathcal{Y} \cup \{z\}), \forall z \succ i \wedge z \in E(\mathcal{X})$ **THEN**
7 **RETURN** $\mathcal{Y} \cup \bigcup_{z \succ i \wedge z \in E(\mathcal{X})}\{z\}$ nếu là HUI; // nhảy bao đóng
8 **ELSE**
9 $Primary(\mathcal{Y}) = \{z \in Secondary(\mathcal{X}) | su(\mathcal{Y}, z) \geq minutil\}$;
10 $Secondary(\mathcal{Y}) = \{z \in Secondary(\mathcal{X}) | lu(\mathcal{Y}, z) \geq minutil\}$;
11 $Search(\mathcal{Y}, \mathbb{D}_{\mathcal{Y}}, Primary(\mathcal{Y}), Secondary(\mathcal{Y}), minutil, P\text{-}set(\mathcal{Y}, z))$;
12 **IF** $\mathcal{Y}$ không có mở rộng tiến **AND** $u(\mathcal{Y}) \geq minutil$ **THEN**
13 **RETURN** $\mathcal{Y}$;
14 **END ELSE**
15 **END IF**
16 **END FOR**

Giải thuật 2.2. Quá trình tìm kiếm theo chiều sâu của giải thuật iEFIM-Closed

2.5 Đánh giá mức độ hiệu quả của P-set

Trong thực tế áp dụng trên các CSDL thực nghiệm, số lượng giao dịch cần phải xét giảm mạnh khi các tập mục cần xét là ít phổ biến – xuất hiện không nhiều trong tập các giao dịch – các CSDL có mật độ thấp. Tuy nhiên, khi phải xử lý trên các CSDL có mật độ cao, các hạng mục xuất hiện trong nhiều giao dịch thì số lượng giao dịch cần phải quét qua được giảm không đáng kể. Trong trường hợp đặc biệt, khi một tập mục xuất hiện trong tất cả các giao dịch của CSDL $\mathbb{D}$, $P\text{-}set$ không giúp giảm số lượng giao dịch, mà làm gia tăng chi phí quét CSDL vì cần phát sinh và quản lý các tập $P\text{-}set$. Tuy nhiên, các CSDL có đặc điểm này là không phổ biến trong thực tế khai thác mà chỉ tồn tại dưới dạng các CSDL tổng hợp để đánh giá hiệu quả của các giải thuật khai thác.

2.6 Đánh giá độ phức tạp của giải thuật iEFIM-Closed

Giả sử n là số lượng giao dịch có trong CSDL $\mathbb{D}$, w là chiều dài trung bình các giao dịch này. Về tổng quát, độ phức tạp của giải thuật iEFIM-Closed là tương tự giải thuật EFIM-Closed, ngoại trừ việc xây dựng và sử dụng cấu trúc P-Set. Xét một tập mục $\mathcal{X}$ bất kỳ, $\mathcal{X} \subseteq \mathbb{I}$, khi đó $m = |P\text{-}Set(\mathcal{X})|$ là số giao dịch cần quét.

Chi phí về mặt thời gian để xây dựng cấu trúc P-Set cho từng hạng mục đơn ban đầu là tuyến tính theo n vì giải thuật cần quét qua toàn bộ $\mathbb{D}$ và được thực hiện đồng thời với quá trình tính *subtree utility* [18].

Tại quá trình đệ quy quét không gian tìm kiếm Giải thuật 2.2, số lượng giao dịch cần quét qua trở thành m do thủ tục đệ quy dựa hoàn toàn vào P-Set để quét CSDL. Hơn nữa, chi phí để thực hiện các kỹ thuật HDP, HTM và kiểm tra tính đóng là tuyến tính dựa trên w [18]. Vì vậy, độ phức tạp trung bình của quá trình đệ quy xấp xỉ $O(mw)$.

Khi triển khai iEFIM-Closed trên các CSDL có mật độ thấp, khi đó $m \ll n$ và hiệu quả của P-Set được thể hiện một cách rõ rệt. Trong trường hợp áp dụng iEFIM-Closed trên các CSDL có mật độ cao, m sẽ tiến gần về n và độ phức tạp của iEFM-Closed sẽ tiến dần về độ phức tạp của giải thuật EFIM-Closed.

Vì vậy, độ phức tạp trung bình của iEFIM-Closed xấp xỉ $O(nw \times \log(w))$.

Các chiến lược tỉa ứng viên của iEFIM-Closed được dựa hoàn toàn trên các tính chất đã được chứng minh từ công trình [18], vì vậy tập các kết quả thu được từ giải thuật iEFIM-Closed là đúng đắn.

2.7 Thực nghiệm

2.7.1 Môi trường thực nghiệm

Phần nội dung này mô tả cách thức thực hiện các thực nghiệm để đánh giá hiệu năng của giải thuật iEFIM-Closed được đề xuất. Các thực nghiệm đều được thực hiện trên cùng một máy tính cá nhân với cấu hình chi tiết như sau: Vi xử lý Intel Core i5 64-bit thế hệ 5, xung nhịp cơ bản là 2.7 Ghz. Dung lượng bộ nhớ RAM là 8.0 GB LPDDR3 1867 Mhz. Nền tảng hệ điều hành thực hiện các thực nghiệm là Apple macOS Mojave.

Giải thuật đã đề xuất được hiện thực trên ngôn ngữ lập trình Java và triển khai trên nền Java SDK 8.0. Môi trường phát triển được sử dụng để triển khai các giải thuật là Eclipse. Môi trường thực thi JVM đều được thiết lập ở các thông số mặc định.

2.7.2 Cơ sở dữ liệu thực nghiệm

Các thực nghiệm được thực hiện trên những CSDL giao dịch chuẩn thường được sử dụng trong các báo cáo khoa học về HUIM/CHUIM nhằm đánh giá hiệu năng của các giải thuật HUIM/CHUIM bao gồm Accidents, BMS, Chainstore, Connect, Foodmart, Pumsb, Retail và Chainstore. Trong đó, CSDL Foodmart và Chainstore là hai CSDL được thu thập tại các cửa hàng bán lẻ với thông tin về độ hữu ích của các hạng mục trong hai CSDL này là các dữ liệu thực tế. Các CSDL còn lại như Accidents, BMS, Connect, Pumsb và Retail có độ hữu ích nội và độ hữu ích ngoại được phát sinh ngẫu nhiên trong phạm vi [1, 10], sử dụng phân bố chuẩn [88].

Các CSDL thực nghiệm sử dụng được cung cấp tại thư viện khai thác dữ liệu mã nguồn mở SPMF [88]. Đặc điểm của các CSDL giao dịch sử dụng trong các thực nghiệm này được trình bày tại Bảng 2.5. Trong đó, $|\mathbb{D}|$ biểu diễn kích thước của

CSDL tính trên số lượng các giao dịch mà CSDL đó chứa. |𝕀| thể hiện tổng số các hạng mục phân biệt trong CSDL tương ứng. T_{AVG} cho biết độ dài trung bình của các giao dịch trong CSDL 𝔻 và *Mật độ* biểu thị mức độ *dày* hoặc *thưa* của phân bố các hạng mục trên độ dài trung bình của các giao dịch. Mật độ CSDL được xác định bằng tỷ lệ giữa số lượng hạng mục của tập 𝕀 và chiều dài trung bình các giao dịch trong CSDL tương ứng. Các CSDL có mật độ thấp được cho tại Bảng 2.5 gồm Chainstore, Kosarak, Retail, BMS và Foodmart. Các CSDL còn lại thuộc nhóm có mật độ cao.

Các CSDL được chọn thể hiện nhiều đặc điểm khác nhau của các dạng dữ liệu như độ lớn số lượng giao dịch, độ lớn tập hạng mục, mật độ, v.v… để nhằm đánh giá hiệu quả của giải thuật iEFIM-Closed so với các giải thuật cùng lớp như EFIM-Closed và CLS-Miner.

Bảng 2.5. Các CSDL thực nghiệm

Database	$\|\mathbb{D}\|$	$\|\mathbb{I}\|$	T_{AVG}	**Mật độ**
Chainstore	1,112,949	46,086	7.2	$1.57E-04$
Kosarak	990,002	41,270	8.1	$1.96E-04$
Accidents	340,183	468	33.8	$7.22E-02$
Retail	87,943	16,465	10.3	$6.26E-04$
Connect	67,557	129	43	$3.33E-01$
BMS	59,601	497	4.8	$5.03E-03$
Pumsb	49,046	7117	74	$3.50E-02$
Foodmart	4,141	1,559	4.4	$2.82E-03$

2.7.3 Phương pháp đánh giá

Giải thuật iEFIM-Closed được so sánh với hai giải thuật cùng lớp là EFIM-Closed và CLS-Miner để đánh giá hiệu năng mà iEFIM-Closed đạt được. Các tiêu chí để so sánh bao gồm:

- ***Thời gian khai thác:*** Tính từ lúc giải thuật khởi chạy cho đến khi kết thúc (và kết xuất được tập kết quả trả về), hoặc xuất hiện lỗi như tràn bộ nhớ, hoặc thời gian thực hiện quá dài trong khi các giải thuật so sánh khác đã có kết quả.

- ***Bộ nhớ sử dụng:*** Lượng bộ nhớ tối đa được giải thuật sử dụng trong toàn bộ quá trình thực thi.

- ***Chi phí quét CSDL:*** so sánh số lượng các giao dịch được quét qua trong quá trình khai thác. Tiêu chí này chỉ áp dụng đối với 2 giải thuật là EFIM-Closed và iEFIM-Closed.

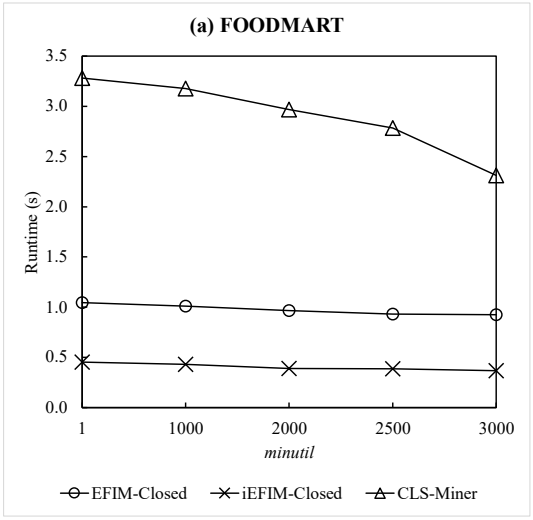
- ***Khả năng mở rộng:*** Áp dụng trên các CSDL thực nghiệm có kích thước lớn. Thay đổi số lượng giao dịch trong mỗi CSDL theo một tỷ lệ xác định như 25%, 50%, 75% và 100% so với tập dữ liệu ban đầu và đánh giá thời gian khai thác của các giải thuật khi lượng dữ liệu gia tăng.

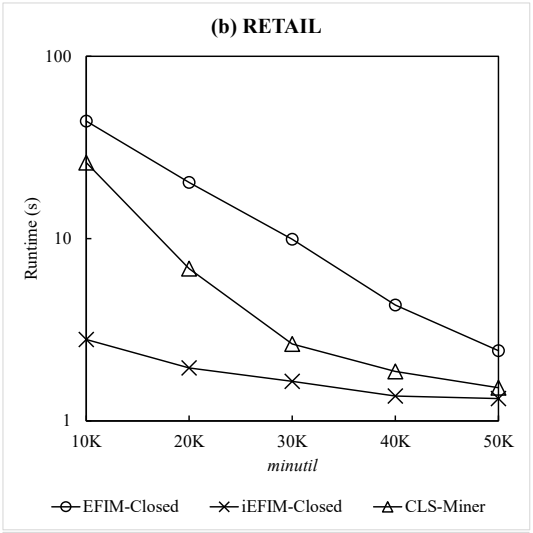
Quá trình tính toán, thu thập số liệu được triển khai bằng cách cho thực hiện các giải thuật mười lần ở các ngưỡng $minutil$ khác nhau, sau đó xác định giá trị trung bình thu nhận được. Các hàm thư viện chuẩn của Java được sử dụng để ghi nhận thời gian thực hiện (đơn vị là giây – s) và lượng bộ nhớ đã cấp phát (đơn vị là megabyte – MB). Các tiêu chí đánh giá này có giá trị càng thấp càng thể hiện hiệu quả tốt hơn. Mã nguồn của giải thuật EFIM-Closed được cung cấp tại thư viện khai thác dữ liệu mã nguồn mở SPMF [88]. Mã nguồn của giải thuật CLS-Miner được nhóm tác giả của giải thuật này cung cấp [30].

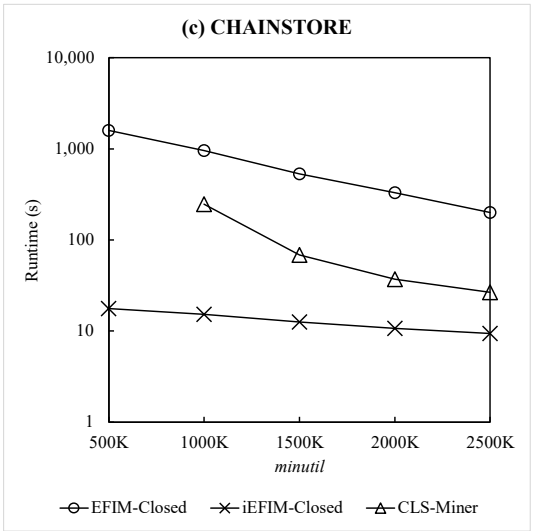
2.7.4 Thời gian thực hiện

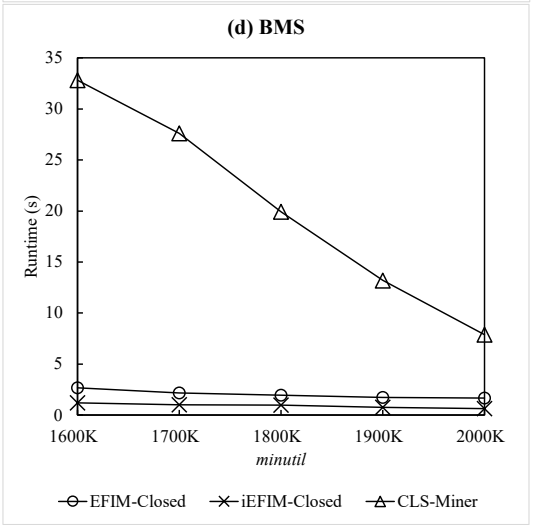
Bằng cách sử dụng các kỹ thuật tối ưu hóa đã trình bày trong chương trước như HDP, HTM, FUC, sub-tree utility, độ hữu ích cục bộ, hai giải thuật EFIM-Closed và iEFIM-Closed cho thấy hiệu năng vượt trội về mặt thời gian so với giải thuật CLS-Miner trên các CSDL có mật độ cao. Tuy nhiên, EFIM-Closed lại không có ưu thế về mặt tốc độ trên các CSDL có mật độ thấp thể hiện qua tốc độ thực hiện trở nên chậm khi giảm ngưỡng $minutil$. Khi đưa vào áp dụng $P\text{-}set$, iEFIM-Closed cho thấy một sự

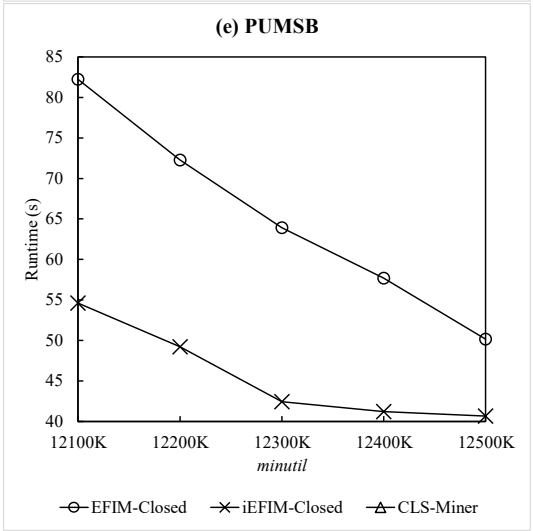
cải thiện rõ rệt về tốc độ trên các CSDL có mật độ thấp như Foodmart, Retail, Chainstore, BMS và Kosarak.

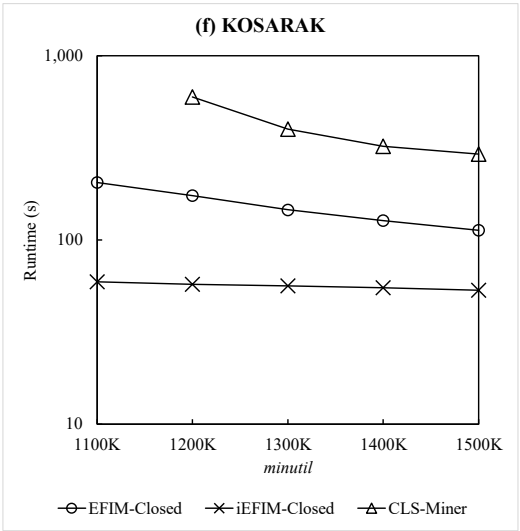


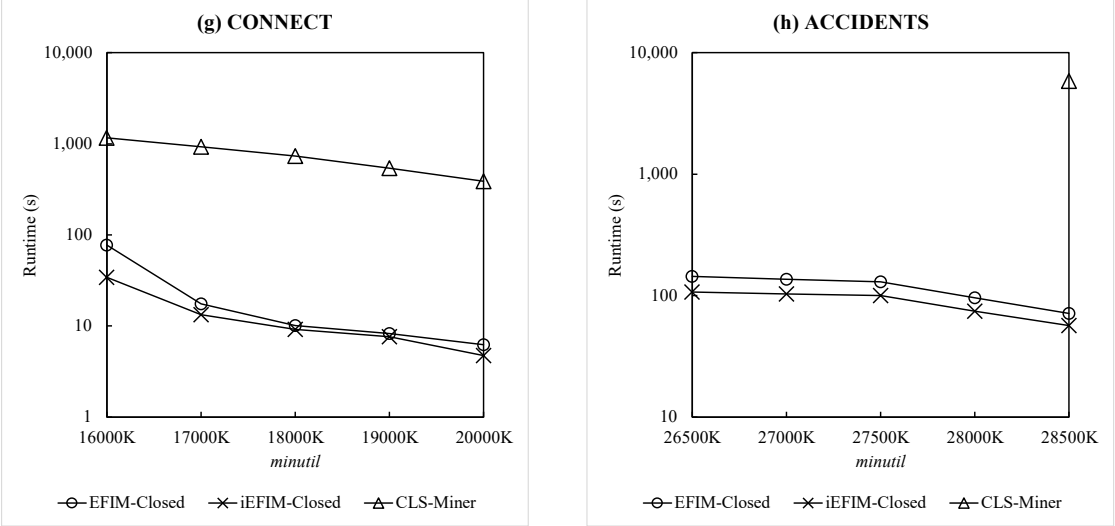












Hình 2.2. So sánh thời gian thực hiện trên các CSDL [CT4]

Các kết quả thực nghiệm thể hiện hiệu năng về mặt thời gian của giải thuật cải tiến iEFIM-Closed từ xấp xỉ cho đến vượt trội gấp nhiều lần so với giải thuật ban đầu là EFIM-Closed (Hình 2.2).

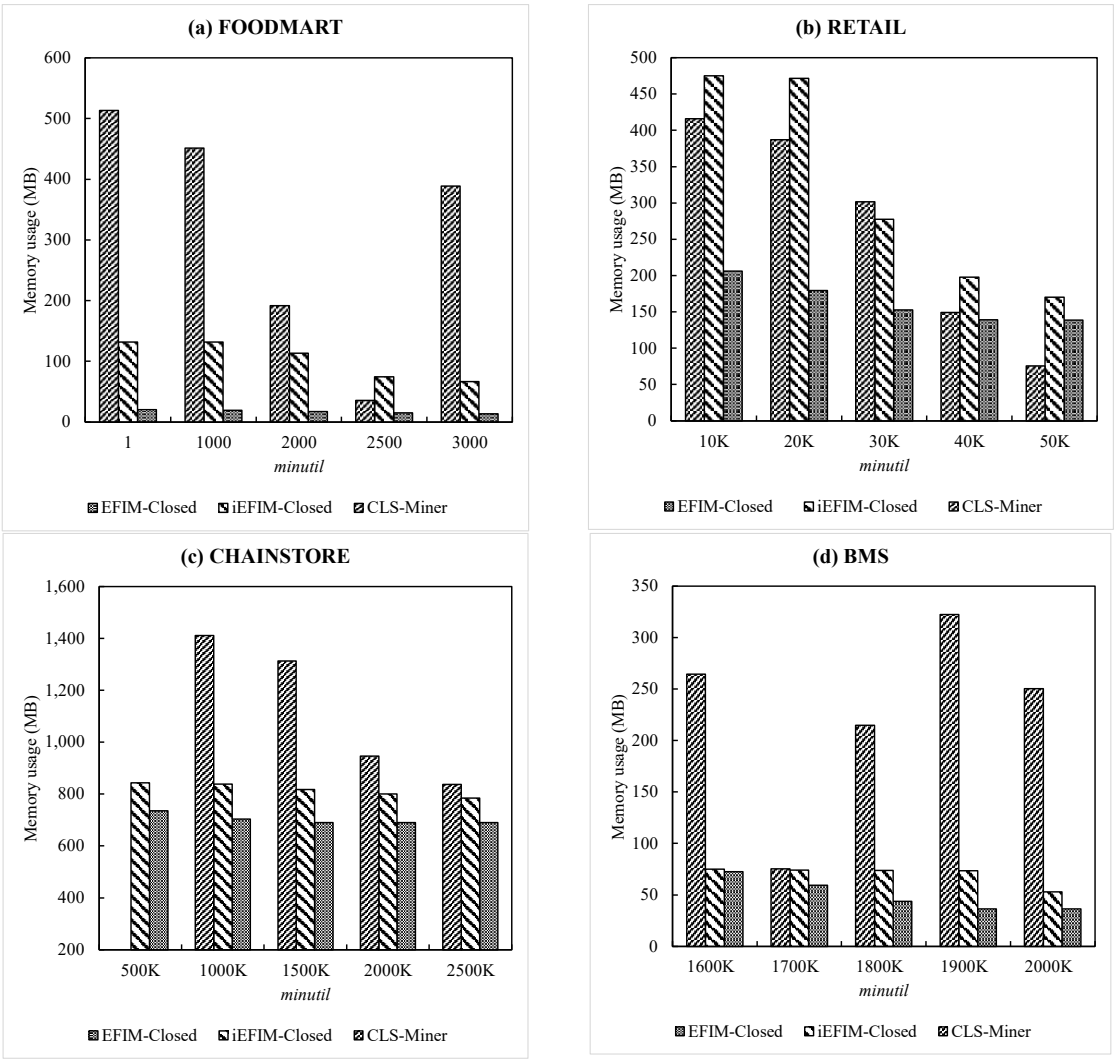
Giải thuật iEFIM-Closed đặc biệt cho kết quả thật sự tốt trên các CSDL có dạng *thưa* như Retail (tốt hơn 15.8 lần) và Chainstore (tốt hơn 90 lần), điều mà giải thuật EFIM-Closed phải mất rất nhiều chi phí để quét toàn bộ CSDL giao dịch trong quá trình tính toán các giới hạn trên, phát sinh CSDL chiếu. Đối với các CSDL *dày* là Accidents, Connect và Pumsb, thời gian chạy của hai giải thuật EFIM-Closed và iEFIM-Closed là gần như tương đương nhau, tuy nhiên iEFIM-Closed có nhỉnh hơn chút ít so với EFIM-Closed nhờ vào việc giảm được một số lượng các giao dịch không cần thiết trong quá trình khai thác.

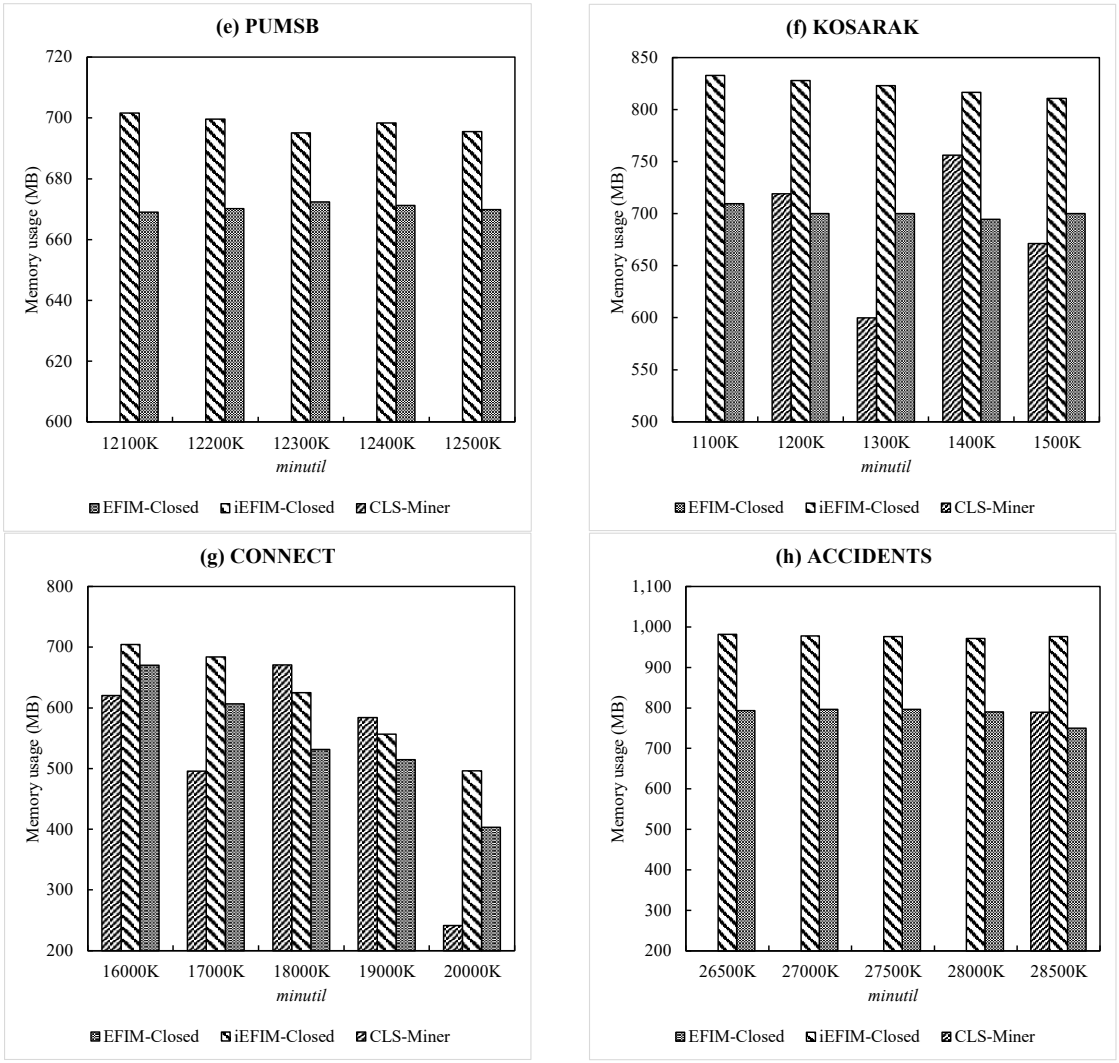
Giải thuật CLS-Miner không hoàn thành ở bốn trên tám CSDL thực nghiệm. Cụ thể, CLS-Miner không thể hoàn thành quá trình khai thác ở $minutil = 500K$ đối với CSDL Chainstore và $minutil = 1100K$ đối với CSDL Kosarak. Đối với CSDL Accidents, CLS-Miner chỉ hoàn thành được ngưỡng cao nhất trong các ngưỡng $minutil$ được thực nghiệm. Đối với CSDL Pumsb, CLS-Miner hoàn toàn không thể thực hiện quá trình khai thác ở tất cả các ngưỡng $minutil$.

2.7.5 Mức độ sử dụng bộ nhớ

Tương tự quá trình so sánh hiệu quả về thời gian, để xác định lượng bộ nhớ cần thiết sử dụng trong các giải thuật, ta cũng cho các giải thuật cần so sánh thực hiện nhiều lần trên nhiều ngưỡng *minutil* khác nhau. Lượng bộ nhớ sử dụng tối đa sau mỗi lần thực hiện sẽ được ghi nhận lại và tính trung bình theo đơn vị *megabyte* (MB). Các kết quả đánh giá được cho trong Hình 2.3. Việc đo đạc, ghi nhận mức độ sử dụng bộ nhớ được thực hiện thông qua các hàm thư viện chuẩn của Java.

Tương tự như quá trình thực nghiệm trên thời gian thực hiện, CLS-Miner cũng chỉ hoàn thành bốn trên tám CSDL đã đưa vào thực nghiệm.





Hình 2.3. So sánh mức độ sử dụng bộ nhớ trên các CSDL [CT4]

Cần lưu ý rằng CLS-Miner không hoàn thành tại một số ngưỡng *minutil* trên CSDL như Kosarak, Chainstore và Accidents. Đối với Pumsb, CLS-Miner không hoàn thành được trên tất cả các ngưỡng *minutil* đã thực nghiệm. Vì vậy, trên các biểu đồ này không có sự xuất hiện của giải thuật CLS-Miner.

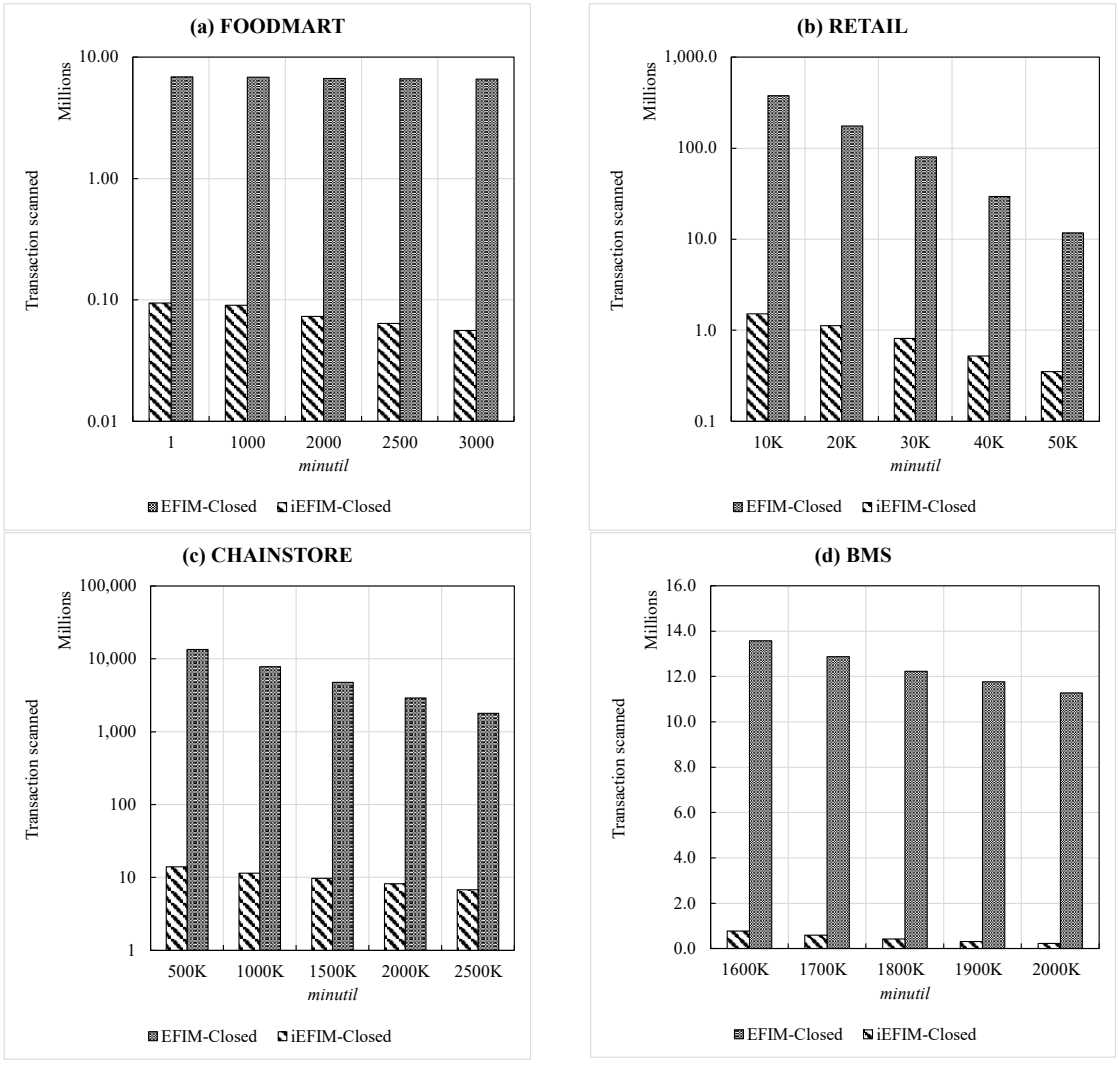
Bảng 2.6. So sánh số lượng giao dịch cần quét trong quá trình khai thác

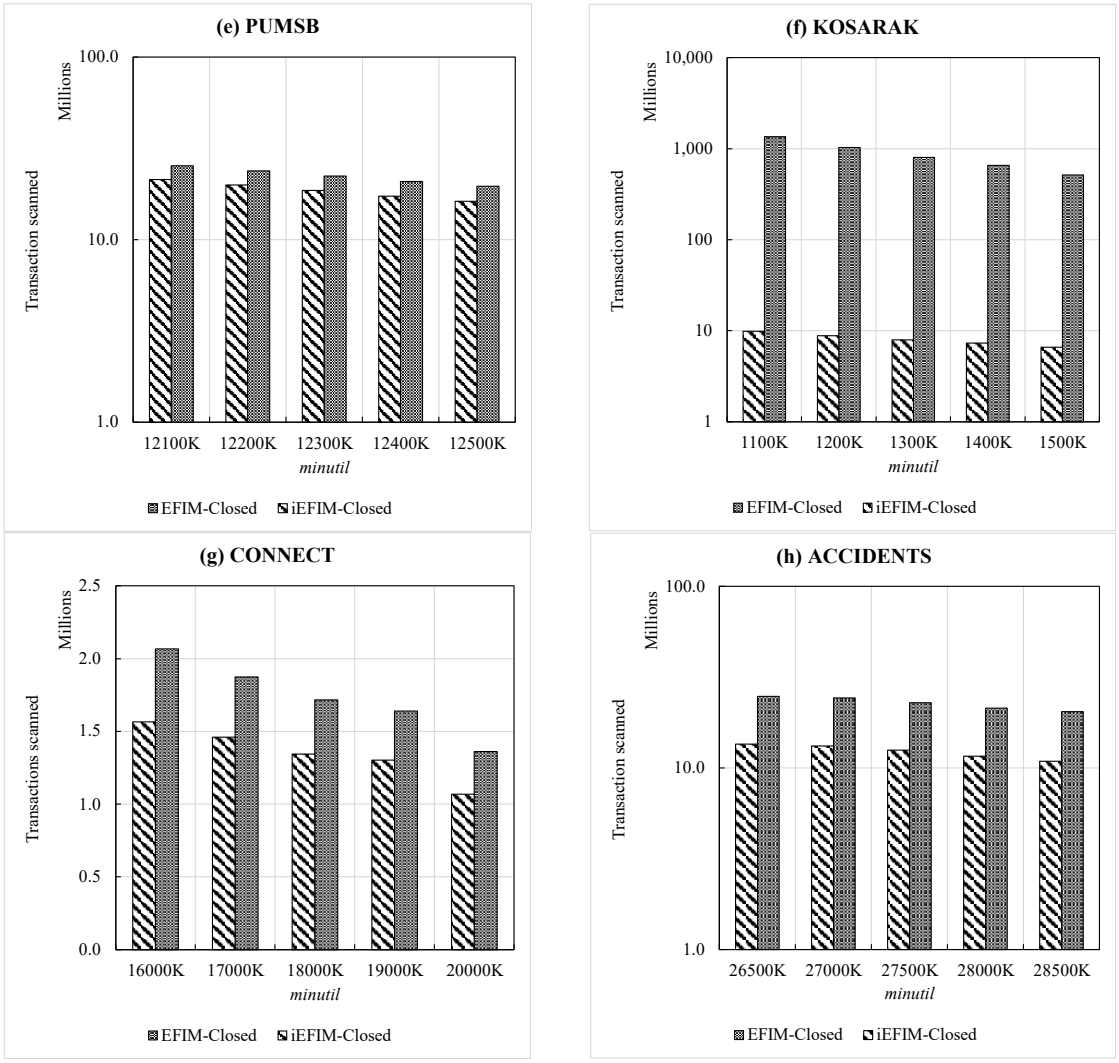
CSDL	iEFIM-Closed	EFIM-Closed	Giảm	Quét
Chainstore	10,021,074	6,141,636,120	**99.8%**	0.2%
Retail	861,798	134,978,290	**99.4%**	0.6%
Kosarak	8,101,676	872,102,126	**99.1%**	0.9%

Foodmart	75,484	6,716,739	**98.9%**	1.1%
BMS	175,270	10,002,024	**98.2%**	1.8%
Accidents	12,357,609	22,724,687	**45.6%**	54.4%
Connect	1,348,154	1,732,230	**22.2%**	77.8%
Pumsb	18,679,150	22,393,751	**16.6%**	83.4%

2.7.6 Chi phí quét CSDL

Số lần quét các giao dịch trung bình của cả hai giải thuật được cho tại Bảng 2.6. Ta nhận thấy trong tất cả các CSDL thưa, kết quả đạt được của giải thuật iEFIM-Closed là tốt hơn gấp nhiều so với giải thuật ban đầu. Lượng giao dịch cần quét trên các CSDL thưa giảm lên đến 99.8% đối với Chainstore và 98.2% đối với BMS.





Hình 2.4. So sánh số lượng giao dịch được quét trên các CSDL [CT4]

Một cách chi tiết hơn, Hình 2.4 cung cấp góc nhìn tổng quát về số lượng các giao dịch được cả hai giải thuật là EFIM-Closed lẫn iEFIM-Closed quét qua trong quá trình khai thác.

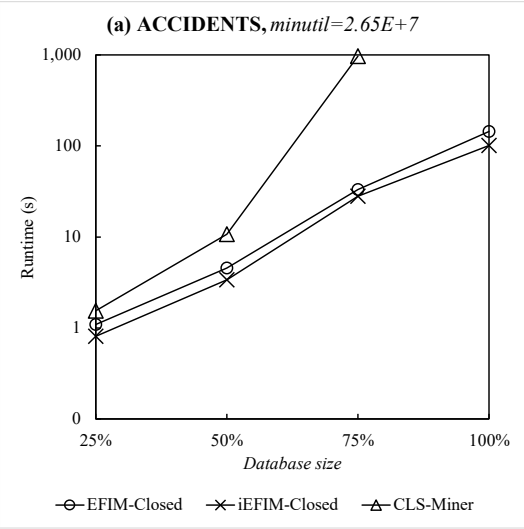
Kết quả cho thấy, với việc ứng dụng cấu trúc P-set vào giải thuật đã giúp giảm một cách đáng kể số lượng giao dịch phải xét. Qua đó, cũng giảm chi phí về mặt thời gian của giải thuật, đặc biệt trên các CSDL có kích thước lớn và mật độ thấp như Chainstore, Kosarak.

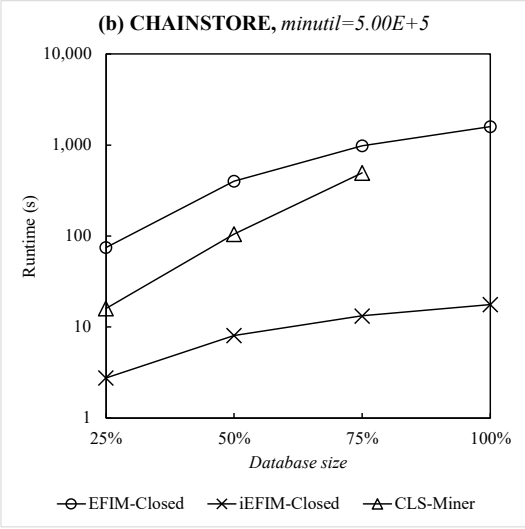
2.7.7 Khả năng thích nghi với việc mở rộng CSDL

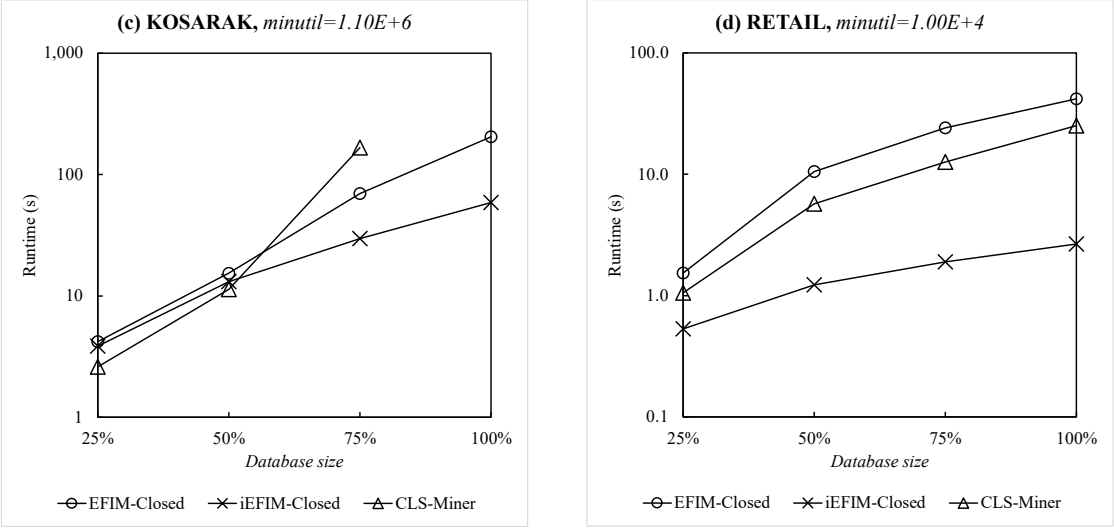
Để đánh giá khả năng mở rộng của giải thuật iEFIM-Closed, một số CSDL có kích thước lớn được lựa chọn để thực nghiệm. Cụ thể, Chainstore, Kosarak, Accidents và Retail lần lượt được lựa chọn. Chi tiết các CSDL này được liệt kê tại Bảng 2.5 .

Trong thực nghiệm này, kích thước CSDL thực nghiệm sẽ được thay đổi theo các tỷ lệ 25%, 50%, 70% và 100% kích thước CSDL ban đầu. Ngưỡng *minutil* của mỗi CSDL được đặt ở mức thấp nhất trong bài so sánh về thời gian thực hiện. Thời gian thực hiện của các giải thuật sẽ được ghi nhận lại và tính trung bình sau nhiều lần thực nghiệm (tối thiểu là 10 lần). Kết quả thực nghiệm được cho tại Hình 2.5.

Có thể thấy rằng iEFIM-Closed cho khả năng mở rộng tốt hơn rõ rệt so với giải thuật gốc là EFIM-Closed, đặc biệt trên các CSDL thưa. Thời gian thực hiện của iEFIM-Closed tăng khi kích thước CSDL mở rộng. Tuy nhiên, mức độ gia tăng thời gian thực hiện của iEFIM-Closed thể hiện tốt hơn đáng kể so với EFIM-Closed và CLS-Miner. Trong đó CLS-Miner không hoàn thành ở ba trên bốn CSDL thực nghiệm là Accidents, Chainstore và Kosarak, vì vậy các biểu đồ này không thể hiện kết quả của giải thuật.

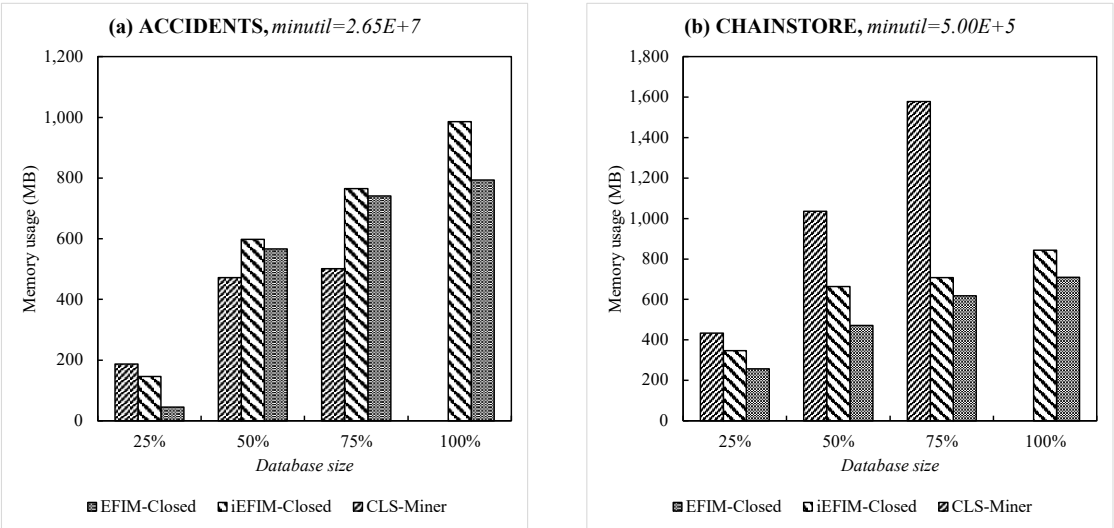


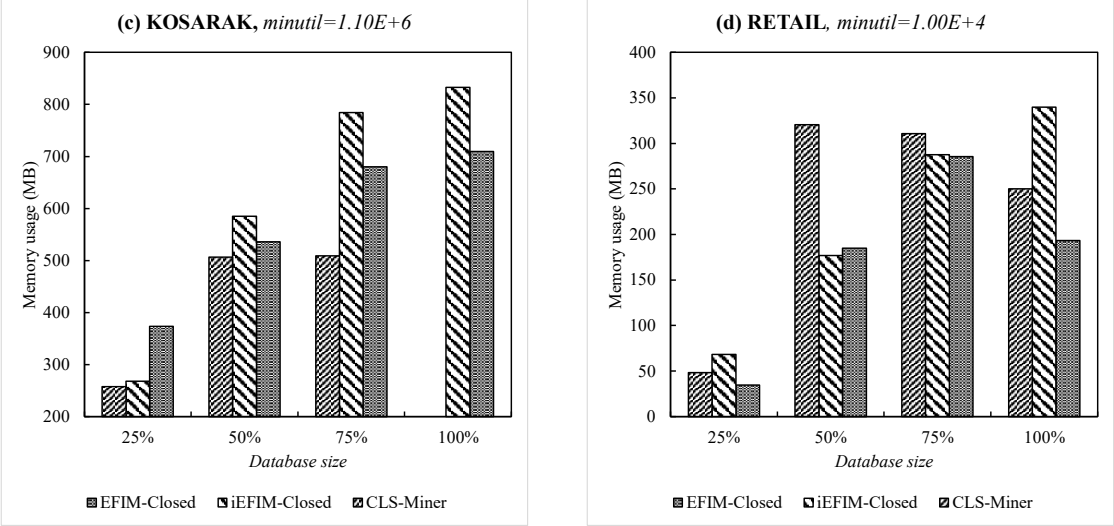




Hình 2.5. So sánh khả năng mở rộng theo tiêu chí thời gian thực hiện. [CT4]

Khả năng mở rộng của các giải thuật cũng được đánh giá dưới góc độ mức độ sử dụng bộ nhớ. Hình 2.6 thể hiện kết quả thu được trong quá trình thực nghiệm. Điều này cũng tương đồng với phép so sánh mức độ sử dụng bộ nhớ ở phần trước. iEFIM-Closed luôn có mức độ yêu cầu bộ nhớ cao hơn so với EFIM-Closed trong tất cả các CSDL thực nghiệm. Đối với CSDL Chainstore, giải thuật CLS-Miner có mức độ sử dụng bộ nhớ gấp nhiều lần EFIM-Closed và iEFIM-Closed. Lý do chính là do Chainstore có tập các hạng mục rất lớn, trên 46 ngàn hạng mục. Điều này làm phát sinh một lượng lớn utility-list để lưu trữ các thông tin của chúng. Quan sát tương tự cũng diễn ra ở CSDL Retail, tuy nhiên mức độ chênh lệch là không lớn như so với CSDL Chainstore.





Hình 2.6. So sánh khả năng mở rộng theo tiêu chí bộ nhớ tiêu thụ. [CT4]

2.8 Kết chương

Nội dung nghiên cứu này đề xuất một giải thuật có tên gọi là iEFIM-Closed. Đây là giải thuật phục vụ cho bài toán khai thác các tập mục hữu ích cao đóng dựa trên việc cải tiến và cập nhật giải thuật EFIM-Closed. So với EFIM-Closed, giải thuật iEFIM-Closed có khả năng thực hiện:

- Khai thác một cách hiệu quả tất cả các tập mục hữu ích cao đóng trong một CSDL giao dịch.
- Có thể áp dụng lên các cơ sở dữ liệu có độ hữu ích thay đổi.
- Áp dụng một kỹ thuật mới nhằm tăng hiệu năng thực hiện của giải thuật, giảm thiểu chi phí quét CSDL nhiều lần khi áp dụng lên các CSDL thực tế.
- Giải thuật có hiệu năng khai thác tốt trên cả hai định dạng CSDL có mật độ cao và mật độ thấp. Tuy nhiên, sự cải thiện đáng kể về hiệu năng khai thác được thể hiện một cách rõ rệt trên các CSDL thưa, lên đến 90 lần so với giải thuật EFIM-Closed.

Qua một số thực nghiệm của iEFIM-Closed trên một số CSDL giao dịch với độ hữu ích thay đổi, ta nhận thấy giải thuật này hoạt động đúng theo mục đích đã đề ra ban đầu: khai thác các tập mục hữu ích cao đóng trên CSDL có độ hữu ích thay đổi. Hơn nữa, bằng việc áp dụng cấu trúc dữ liệu $P\text{-}set$ vào giải thuật đã giúp cho giải thuật có

hiệu năng từ tương đương cho đến tốt hơn rất nhiều lần so với các giải thuật EFIM-Closed và CLS-Miner, giúp rút ngắn thời gian khai thác các tập mục hữu ích cao.

So với giải thuật tốt nhất hiện nay để khai thác tập mục hữu ích cao đóng trên CSDL giao dịch là EFIM-Closed, giải thuật iEFIM-Closed cho thấy có hiệu năng khai thác vượt trội, đặc biệt là trên các CSDL có mật độ thấp (thưa). Hiệu năng cải thiện của iEFIM-Closed lên đến 90 lần so với giải thuật ban đầu. Qua đó, cho thấy giải thuật iEFIM-Closed đã giải quyết được nội dung nghiên cứu đầu tiên của luận án.

Tuy nhiên, với cơ chế kiểm tra tính đóng của các tập mục đã trình bày như mở rộng tiến, mở rộng lùi, nhảy bao đóng đều tồn tại phụ thuộc dữ liệu vào các tập mục đã khai thác trước đó. Vì vậy, việc triển khai mô hình song song hoá trên iEFIM-Closed là không khả thi.

Vì vậy, hướng phát triển của giải thuật có thể được xem xét gồm:

- Thông qua các khảo sát đã thực hiện ở CHƯƠNG 1, các CSDL phân cấp cung cấp nhiều các tri thức quan trọng hơn và tổng quát hơn so với các CSDL giao dịch truyền thống. Bên cạnh đó, định dạng CSDL này cũng làm cho tập kích thước tập kết quả gia tăng theo một cách đáng kể. Việc mở rộng bài toán khai thác tập mục hữu ích cao đóng để có thể hoạt động trên các CSDL có phân cấp các hạng mục, từ đó khai thác được các tập mục đa mức (multi-level) có tính tổng quát cao hơn so với các CSDL giao dịch thông thường là cần thiết.
- Với kích thước của không gian bài toán gia tăng, đồng nghĩa với thời gian khai thác của các giải thuật cũng sẽ kéo dài hơn. Việc nghiên cứu áp dụng cơ chế song song hoá theo từng mức của CSDL giao dịch có sự phân cấp các hạng mục là cần thiết để giảm thiểu thời gian khai thác.

Đây cũng là các trọng tâm của **Nội dung nghiên cứu 2** đã nêu ở phần MỞ ĐẦU. Các nội dung này sẽ được tập trung khảo sát và triển khai trong chương tiếp theo của luận án, CHƯƠNG 2.

CHƯƠNG 3 MÔ HÌNH SONG SONG HÓA KHAI THÁC TẬP MỤC HỮU ÍCH CAO ĐA MỨC

3.1 Giới thiệu

Như đã trình bày trong CHƯƠNG 1, phần 1.1, mục tiêu của bài toán HUIM là tìm ra các nhóm hạng mục mang lại giá trị độ hữu ích cao trong CSDL khi chúng được giao dịch cùng nhau. Người dùng khi đó sẽ được yêu cầu cung cấp một giá trị gọi là ngưỡng độ hữu ích tối thiểu $minutil$. Các giải thuật khai thác HUI sẽ quét không gian tìm kiếm của bài toán để xác định các tập mục thoả ngưỡng đã cho trước và kết xuất ra tất cả các tập hạng mục này.

Tuy nhiên, nếu ngưỡng $minutil$ được ấn định với giá trị thấp, số lượng tập mục trả về cho người dùng là rất lớn. Để phân tích đầy đủ các tập mục này, người sử dụng cần mất rất nhiều thời gian. Bên cạnh đó, trong tập kết quả nhận được, số lượng tập mục mang ý nghĩa trùng lặp là không nhỏ.

Bảng 3.1. Ví dụ về CSDL giao dịch giao dịch

TID	**Các hạng mục**	**Độ hữu ích nội**	**TU**
T_1	a, c	$1, 2$	21
T_2	b, c	$2, 2$	22
T_3	b, d, e	$3, 2, 1$	12
T_4	a, c, e	$2, 2, 1$	27

Bảng 3.2. Độ hữu ích đơn ngoại của các hạng mục cho tại Bảng 3.1

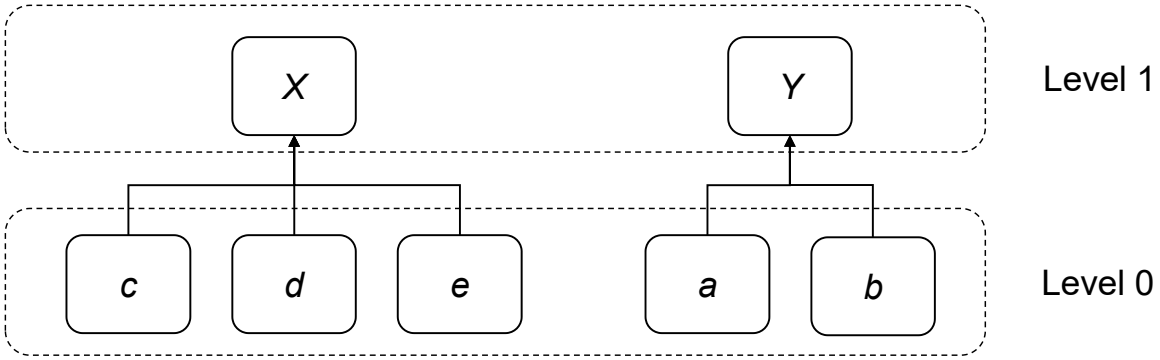
Các hạng mục	a	b	c	d	e
Độ hữu ích ngoại	\$1	\$1	\$10	\$2	\$5

Nội dung nghiên cứu 1, thông qua toàn bộ CHƯƠNG 2, đã đề cập đến hạn chế của bài toán HUIM, CHUIM thông thường và đề xuất giải pháp để tăng hiệu quả khai thác. Kết quả của CHƯƠNG 2 nói riêng và **Nội dung nghiên cứu 1** nói chung cho

thấy có sự cải thiện rõ rệt về hiệu năng khai thác với mức độ cải thiện lên đến trên 90 lần so với các giải thuật cùng lớp bài toán.

Tuy nhiên, giải thuật được đề xuất trong nội dung nghiên cứu này lại không khả thi để áp dụng mô hình xử lý song song nhằm tăng hiệu quả khai thác của bài toán do các ràng buộc về dữ liệu phát sinh trong không gian tìm kiếm. Do vậy, mục đích của **Nội dung nghiên cứu 2** là tìm kiếm giải pháp để có thể triển khai mô hình xử lý song song hóa vào bài toán HUIM với mục tiêu giảm thời gian cần thiết cho quá trình khai thác. Ngoài ra, giải thuật đề xuất chỉ quan tâm đến các hạng mục cụ thể có trong CSDL mà không xem xét đến các nhóm hạng mục, nhóm mặt hàng thường được kinh doanh.

Trong thực tế, các CSDL được sử dụng thường được tổ chức dưới dạng phân cấp [34], [89], [90], [91] để tăng tính hiệu quả quản lý, có thể lưu trữ và biểu diễn được nhiều các thông tin hơn như minh họa ở Hình 3.1. Cấu trúc cho ở Hình 3.1 là cấu trúc phân cấp các hạng mục trong CSDL và được gọi là *taxonomy*. Tuy nhiên, với giải thuật iEFIM-Closed đề xuất ở **Nội dung nghiên cứu 1**, các thông tin này sẽ bị bỏ qua mặc dù chúng mang lại nhiều tri thức hữu ích phục vụ cho quá trình kinh doanh.



Hình 3.1. Cấu trúc phân cấp gồm 2 mức trong CSDL tại Bảng 3.1

Mở rộng từ bài toán khai thác tập mục phổ biến trên CSDL phân cấp [34], [89], [90], [91], một số tiếp cận khai thác tập mục hữu ích cao gần đây đã xem xét đến sự phân cấp các hạng mục trong CSDL [40], [73], [75], [92] để tìm ra các tập HUI tổng quát. Kết quả tìm được là các tập mục hữu ích cao đa mức hoặc liên cấp. Tuy nhiên, cùng với việc gia tăng số lượng mức trừu tượng trong cấu trúc phân cấp các hạng mục

taxonomy, số lượng các tập mục hữu ích cao được tìm thấy cũng gia tăng với số lượng đáng kể. Bảng 3.3 cho ta thấy rõ điều này, với $minutil = 60$, số tập mục tìm được nếu không xét đến sự phân cấp hạng mục như trong CSDL cho ở Bảng 3.1, ta chỉ có một tập mục duy nhất là c. Tuy nhiên, khi xét đến sự phân cấp các hạng mục như minh hoạ tại Hình 3.1, số tập mục tìm được gia tăng thành ba tập. Số lượng tập mục tổng quát này cũng sẽ gia tăng khi cấu trúc phân cấp taxonomy có chiều sâu lớn và số lượng các hạng mục tổng quát tăng. Đây cũng là yếu tố làm tăng kích thước không gian tìm kiếm và đồng thời tăng đáng kể độ phức tạp của các giải thuật HUI.

Bảng 3.3. MLHUI tìm được trong CSDL cho ở Bảng 3.1, $minutil = 60$.

MLHUI	**Utility**
$\{X, Y\}$	\$82
$\{X\}$	\$74
$\{c\}$	\$60

Hình 3.1 thể hiện không gian tìm kiếm của bài toán ứng với CSDL phân cấp được cho tại Bảng 3.1. Qua đây, ta có thể nhận thấy rõ ràng rằng mỗi mức trong taxonomy có không gian tìm kiếm độc lập nhau, và các hạng mục thuộc mức nào chỉ tham gia vào việc xây dựng các tập mục thuộc mức đó mà thôi. Các tập mục tìm được trong CSDL sử dụng cấu trúc này với ràng buộc nói trên được gọi là các tập mục hữu ích cao đa mức.

Với không gian bài toán mở rộng thông qua taxonomy τ, số lượng tập mục nhận được sẽ càng lớn hơn một cách đáng kể. Dẫn đến thời gian cần thiết để khai thác và phân tích các kết quả này là không nhỏ.

Tuy nhiên, như đã trình bày trong phần đặt vấn đề, cho đến nay vẫn chưa có một giải thuật nào được đề xuất để giải quyết bài toán khai thác tập mục hữu ích cao đóng trên CSDL phân cấp nhằm khắc phục các vấn đề đã nêu của bài toán khai thác tập mục

hữu ích cao trên CSDL thông thường cũng như trên CSDL phân cấp, đó là số lượng tập mục nhận lại từ giải thuật quá lớn.

Vì thế, ta cần có giải pháp cho phép khai thác hiệu quả các tập mục hữu ích cao từ những CSDL có sự phân cấp hạng mục. Ngoài ra, nhằm thu gọn số lượng tập mục kết quả nhưng vẫn bảo toàn độ chính xác của quá trình khai thác, biểu diễn đóng các tập mục hữu ích cao cũng giúp giảm thiểu thời gian cần thiết để phân tích các tập mục HUI đã khai thác được.

Hơn nữa, các CSDL được sử dụng trong các nghiên cứu trước đó đều sử dụng cùng một giả định là độ hữu ích của các hạng mục là tĩnh. Vì vậy, từ nội dung chương này trở về sau, các tiếp cận, giải thuật được đề xuất sẽ áp dụng trên định dạng CSDL có sự biến đổi về độ hữu ích của các hạng mục cùng với nền tảng tính toán độ hữu ích động như đã trình bày trong chương trước. Điều này giúp cho các nghiên cứu, đóng góp của luận án đi sát hơn vào các ứng dụng thực tế. Đây cũng chính là một trong những điểm mới của luận án và là trọng tâm của toàn bộ chương này.

Nội dung chính của chương này bao gồm:

- Đề xuất chiến lược song song hoá để khai thác các tập mục hữu ích cao đa mức từ CSDL phân cấp.
- Đề xuất giải thuật tận dụng chiến lược song song hoá đã đề xuất để khai thác tập mục hữu ích cao đa mức, tập mục hữu ích cao đa mức - đóng từ CSDL phân cấp.

3.2 Một số định nghĩa

Phần này trình bày một số định nghĩa cần thiết để phục vụ cho bài toán khai thác tập mục hữu ích cao đa mức. Các định nghĩa cơ bản của bài toán khai thác tập mục hữu ích cao đã được nêu tại CHƯƠNG 2.

Để làm rõ các định nghĩa được trình bày trong nội dung này, ta xét một CSDL mẫu được cho tại Bảng 3.1 và Bảng 3.2.

Bảng 3.4. TWU của các hạng mục đơn từ CSDL ở Bảng 3.1 và Bảng 3.2

Hạng mục	c	a	e	b	d
TWU	\$70	\$48	\$39	\$34	\$12

Bảng 3.5. CSDL sau lần quét thứ 2, với $minutil = 30$.

TID	**Giao dịch**	**Độ hữu ích**	**TU**
T_1	c, a	$20, 1$	21
T_2	c, b	$20, 2$	22
T_3	e, b	$5, 3$	8
T_4	c, a, e	$20, 2, 5$	27

Lưu ý rằng độ hữu ích của các hạng mục trong CSDL có sự phân cấp các hạng mục cũng tuân theo **Định nghĩa 2.6** để bảo đảm tính biến động trong thực tế khai thác. Bên cạnh đó, khái niệm về các tập mục được sử dụng trong chương này sẽ được ký hiệu thông qua các ký tự $\mathcal{X}, \mathcal{Y}$, v.v… thay vì X, Y, v.v…. Lý do cho việc sử dụng các ký hiệu này là để phân biệt các hạng mục tổng quát với các tập mục.

Đối với các giải thuật kế thừa từ mô hình của tiếp cận HUI-Miner đề xuất bởi Liu và đồng sự [57], quá trình xác định TWU của các hạng mục đơn sẽ diễn ra ở lần quét CSDL đầu tiên. Với CSDL đã cho ở Bảng 3.1 và Bảng 3.2, giá trị TWU tính được cho các hạng mục đơn được cho tại Bảng 3.4, và được sắp xếp giảm dần theo TWU (Bảng 3.5).

Trong lần quét CSDL thứ hai, các hạng mục không tiềm năng sẽ được loại bỏ khỏi các giao dịch. Các hạng mục tiềm năng còn lại trong các giao dịch sẽ được sắp xếp theo một thứ tự toàn phần $\prec$. Thông thường thứ tự được chọn là chiều giảm dần của TWU các hạng mục [57].

Ví dụ, với $minutil = 30$, hạng mục d được xem là không tiềm năng và sẽ được loại bỏ khỏi tất cả các giao dịch trong $\mathbb{D}$. Bảng 3.5 cho thấy CSDL $\mathbb{D}$ sau khi đã loại bỏ các hạng mục không tiềm năng và sắp xếp lại theo thứ tự $\prec$ [57].

Định nghĩa 3.1 (Độ hữu ích còn lại của một hạng mục trong giao dịch). Độ hữu ích còn lại của tập mục $\mathcal{X} = \{i_1, i_2, \dots, i_k\}$ trong một giao dịch T_c đã được sắp dựa trên thứ tự toàn phần $\prec$ được định nghĩa như sau: [57]

$$ru(\mathcal{X}, T_c) = \sum_{i_k \in T_c \wedge i_k \prec i_j} u(i_j, T_c) \quad (3.1)$$

Đinh nghĩa 3.2 (Độ hữu ích còn lại). Độ hữu ích còn lại của một tập mục $\mathcal{X} = \{i_1, i_2, \dots, i_k\}$ trong toàn bộ CSDL $\mathbb{D}$, được định nghĩa như sau: [57]

$$ru(\mathcal{X}) = \sum_{r \in TidSet(i_j)} ru(i_j, T_r) \quad (3.2)$$

c		
Tid	***EU***	***RU***
1	20	1
2	20	2
4	20	7

a		
Tid	***EU***	***RU***
1	1	0
4	2	5

e		
Tid	***EU***	***RU***
3	5	3
4	5	0

b		
Tid	***EU***	***RU***
2	2	0
3	3	0

Hình 3.2. Utility-list của các hạng mục tiềm năng tại $minutil = 30$

Định nghĩa 3.3 (Cấu trúc Utility-list). Utility-list của một tập mục $\mathcal{X}$ được ký hiệu $UL(\mathcal{X})$, là một danh sách chứa $|TidSet(\mathcal{X})|$ các bộ có dạng sau: $\langle Tid, EU, RU \rangle$. Trong đó, Tid và EU lần lượt là định danh r của giao dịch T_r có chứa $\mathcal{X}$ và EU là độ hữu ích của $\mathcal{X}$ tại T_r. RU chính là độ hữu ích còn lại của $\mathcal{X}$ trong T_r. Mỗi bộ $\langle Tid, EU, RU \rangle$ được gọi là một phần tử trong danh sách và chứa thông tin về độ hữu ích của $\mathcal{X}$ tại mỗi giao dịch T_r với $r \in TidSet(\mathcal{X})$. Một phần tử ứng với giao dịch T_r trong utility-list của $\mathcal{X}$ ký hiệu là $UL(\mathcal{X}, T_r)$, chứa các thông tin sau $\langle r, EU(\mathcal{X}, T_r), RU(\mathcal{X}, T_r) \rangle$. Các thành phần EU và RU của mỗi phần tử trong utility-list được ký hiệu tương ứng là $UL(\mathcal{X}, T_r).EU$ và $UL(\mathcal{X}, T_r).RU$ [57].

Utility-list của các hạng mục tiềm năng với $minutil = 30$ được xây dựng sau khi thực hiện xong lần quét CSDL thứ hai [57]. Hình 3.2 thể hiện cấu trúc utility-list của các hạng mục tiềm năng có được tại CSDL ở Bảng 3.5.

Định nghĩa 3.4 (Mở rộng của một tập mục). Cho hai tập mục $\mathcal{X} = \{x_1, x_2, \dots, x_u\}$ và $\mathcal{Y} = \{y_1, y_2, \dots, y_v\}$ với $x_i, y_j \in \mathbb{I}$. $\mathcal{Y}$ là một mở rộng của $\mathcal{X}$ nếu $\mathcal{X} \subset \mathcal{Y}$ và mỗi hạng mục $y_j \notin \mathcal{X}$ đều đứng sau các hạng mục trong $\mathcal{X}$ dựa trên thứ tự toàn phần $\prec$ [57].

Ví dụ, xét thứ tự $\prec$ là $a \prec c \prec e \prec d \prec b \prec f$, tập mục $\{ace\}$ là một mở rộng của tập mục $\{ac\}$ chứ không phải tập mục $\{ae\}$.

Định nghĩa 3.5 (Tổng EU và RU trong utility-list). Trong utility-list của một tập mục $\mathcal{X}$, tổng tất cả các thành phần EU và RU lần lượt được ký hiệu là $SumEU(\mathcal{X})$ và $SumRU(\mathcal{X})$ [57].

Tính chất 3.1 (Tỉa ứng viên không tiềm năng dựa trên utility-list). Cho tập mục $\mathcal{X}$, nếu $SumEU(\mathcal{X}) + SumRU(\mathcal{X}) < minutil$, tất cả các mở rộng của $\mathcal{X}$ đều là các tập hữu ích thấp và có thể được loại bỏ khỏi không gian tìm kiếm [57].

Tính đúng đắn của **Tính chất 3.1** khi áp dụng vào quá trình loại bỏ các tập mục ứng viên không tiềm năng đã được chứng minh tại công trình [57].

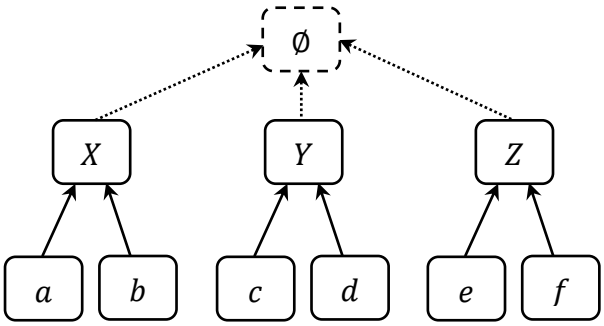
Ví dụ, dựa vào cấu trúc utility-list ở Hình 3.2 và $minutil = 30$, ta có thể thấy được hạng mục c là hạng mục tiềm năng do thoả **Tính chất 3.1**. Các hạng mục còn lại là a, e, b đều được xem là các hạng mục không tiềm năng mặc dù khi áp dụng TWU, chúng đều được xem là tiềm năng. Vì vậy, quá trình khai thác có thể loại bỏ chúng khỏi không gian tìm kiếm. Điều này giúp giảm thiểu đáng kể việc phải xét và xây dựng utility-list cho các tập ứng viên không tiềm năng, rút ngắn thời gian khai thác và giảm thiểu mức độ sử dụng bộ nhớ.

Định nghĩa 3.6 (Taxonomy). Một cấu trúc phân cấp các hạng mục trong CSDL $\mathbb{D}$, ký hiệu τ, là một cấu trúc cây được định nghĩa trên $\mathbb{D}$. Trong τ, mỗi nút lá biểu diễn

duy nhất một hạng mục $i \in \mathbb{I}$. Mỗi nút trong của τ biểu diễn một hạng mục tổng quát ở mức trừu tượng cao hơn làm nhiệm vụ tổng hợp các nút con hoặc các hạng mục tổng quát ở mức trừu tượng thấp hơn [41]. Giả sử rằng tất cả các hạng mục $i \in \mathbb{I}$ chỉ có thể được tổng quát hoá bởi τ vào duy nhất một hạng mục tổng quát g mà thôi. Mỗi hạng mục tổng quát g cũng có thể được tổng quát hoá thành một hạng mục tổng quát khác ở mức trừu tượng cao hơn trong τ. Tập các hạng mục tổng quát g trong τ được ký hiệu là $\mathbb{G}$. Tất cả các nút con của g trong τ được ký hiệu là $\Delta(g, \tau)$, là một tập con của $\mathbb{I}$, $\Delta(g, \tau) \subseteq \mathbb{I}$ [41]. Chiều sâu tối đa của taxonomy τ được ký hiệu là δ.

Để đơn giản, ta gọi CSDL có sự phân cấp các hạng mục $\mathbb{D}$ sử dụng taxonomy τ là CSDL giao dịch $\mathbb{D}$.

Ví dụ, với CSDL $\mathbb{D}$ được cho tại Bảng 1.1 *và* Bảng 1.2, taxonomy τ của CSDL này được thể hiện tại Hình 3.3. Trong ví dụ này, các hạng mục a, b là được tổng quát hoá vào hạng mục tổng quát X, tương tự với c, d cho hạng mục Y, và e, f đối với hạng mục Z. Các hạng mục tổng quát X, Y, Z lại cũng có thể được tổng quát hoá thành một hạng mục tổng quát ở mức trừu tượng cao hơn nếu cần thiết.



Hình 3.3. Cấu trúc taxonomy của CSDL $\mathbb{D}$ cho tại Bảng 1.1

Định nghĩa 3.7 (Cấp của hạng mục tổng quát đa mức). Cho $g \in \mathbb{G}$ là một hạng mục tổng quát sử dụng taxonomy τ, cấp của g trong τ, ký hiệu $l(g, \tau)$, được định nghĩa là chiều dài đường đi ngắn nhất giữa g và bất kỳ nút lá nào của nó trong τ [41].

Tính chất 3.2 (Cấp của các hạng mục chuyên biệt). Tất cả các hạng mục chuyên biệt $i \in \mathbb{I}$, là các nút lá trong τ, có $l(i, \tau) = 0$.

Tính chất 3.3 (Cấp tối đa của hạng mục tổng quát đa mức). Cấp tối đa của một hạng mục tổng quát $g \in \mathbb{G}$ tương ứng với chiều sâu của taxonomy τ.

Ví dụ, cấp của hạng mục chuyên biệt e và f trong taxonomy τ thể hiện tại Hình 3.3 là $l(\{e\},\tau) = level(\{f\},\tau) = 0$. Tương tự, cấp của hạng mục tổng quát Z trong τ là $l(\{Z\},\tau) = 1$.

Định nghĩa 3.8 (Tập mục tổng quát đa mức). Một tập mục tổng quát $\mathcal{X}$ là một tập các hạng mục $g \in \mathbb{G}$ ở cùng một mức trừu tượng thuộc taxonomy τ. Cấp của $\mathcal{X}$ được xác định dựa trên cấp của các hạng mục tổng quát mà nó chứa [41].

Kế thừa từ **Định nghĩa 2.15** kết hợp với taxonomy τ, ta mở rộng khái niệm tập mục đóng cho một tập mục tổng quá đa mức như sau.

Định nghĩa 3.9 (Tập mục đa mức đóng). Cho CSDL $\mathbb{D}$ sử dụng taxonomy τ, một tập mục $\mathcal{X}$ được gọi là tập mục đa mức đóng trong $\mathbb{D}$ nếu và chỉ nếu không tồn tại một tập mục đa mức $\mathcal{Y}$ thoả $\mathcal{Y} \supset \mathcal{X}$, $l(\mathcal{X}) = l(\mathcal{Y})$ và $sup(\mathcal{X}) = sup(\mathcal{Y})$.

Định nghĩa 3.10 (Độ hữu ích của hạng mục tổng quát đa mức). Cho $g \in \mathbb{G}$ là một hạng mục tổng quát trong CSDL $\mathbb{D}$ sử dụng taxonomy τ. Độ hữu ích của g trong giao dịch $T_q \in \mathbb{D}$, ký hiệu $u(g, T_q)$, được định nghĩa như sau [41].

$$u(g, T_q) = \sum_{i \in \Delta(g,\tau)} iu(i, T_q) \times eu(i, T_q) \quad (3.3)$$

Định nghĩa 3.11 (Độ hữu ích của tập mục tổng quát đa mức). Cho $\mathcal{X}$ là một tập mục tổng quát trong CSDL $\mathbb{D}$ sử dụng taxonomy τ. Độ hữu ích của $\mathcal{X}$ trong giao dịch $T_q \in \mathbb{D}$, ký hiệu $u(\mathcal{X}, T_q)$, được xác định như sau [41].

$$u(\mathcal{X}, T_q) = \sum_{g \in \mathcal{X}} u(g, T_q) \quad (3.4)$$

Độ hữu ích của tập mục tổng quát $\mathcal{X}$ trên toàn CSDL $\mathbb{D}$, ký hiệu $u(\mathcal{X})$, được xác định như sau [41].

$$u(\mathcal{X}) = \sum_{T_q \in \mathbb{D}} u(\mathcal{X}, T_q) \quad (3.5)$$

Nếu $\mathcal{X} \not\subseteq T_q$, khi đó $u(\mathcal{X}, T_q) = 0$.

Định nghĩa 3.12 (Tập mục hữu ích cao đa mức). Cho CSDL giao dịch $\mathbb{D}$ cùng với thông tin phân cấp các hạng mục τ và một ngưỡng $minutil$ cho trước. Một tập mục tổng quát $\mathcal{X}$ được gọi là tập mục hữu ích cao đa mức nếu và chỉ nếu $u(\mathcal{X}) \geq minutil$ và $\mathcal{X}$ là tập mục tổng quát đa mức [41].

Định nghĩa 3.13 (Tập mục hữu ích cao đa mức, đóng). Cho CSDL giao dịch $\mathbb{D}$ cùng với thông tin phân cấp các hạng mục τ và một ngưỡng độ hữu ích tối thiểu ($minutil$) cho trước. Một tập mục (tổng quát) $\mathcal{X}$ được gọi là tập mục hữu ích cao đa mức đóng nếu $\mathcal{X}$ là tập mục hữu ích cao đa mức và $\mathcal{X}$ là một tập mục đóng.

Ngưỡng giới hạn trên TWU cũng được mở rộng để tỉa các hạng mục tổng quát không tiềm năng trên các CSDL có sự phân cấp các hạng mục sử dụng taxonomy τ [41] và được trình bày như sau.

Định nghĩa 3.13 (Trọng số hữu ích giao dịch tổng quát). Trọng số giao dịch tổng quát của một tập mục tổng quát $\mathcal{X}$, ký hiệu $TWU(\mathcal{X})$, được định nghĩa như sau:

$$TWU(\mathcal{X}) = \sum_{T_r \in \mathbb{G}(\mathcal{X})} TU(\mathcal{X}) \tag{3.6}$$

Trong đó:

$$\mathbb{G}(\mathcal{X}) = \{T_q \in \mathbb{D} \mid \forall\, g \in \mathcal{X}: \exists v_i \subset T_q \wedge v_i \in \Delta(g, \tau)\} \tag{3.7}$$

Các **Tính chất 2.1** và **Tính chất 2.2** cũng được mở rộng dựa trên **Định nghĩa 3.13** để có thể làm việc được trên các hạng mục, tập mục tổng quát trong CSDL phân cấp, kết hợp với taxonomy τ.

Định nghĩa 3.14 (Cấu trúc ước lượng hữu ích đồng xuất hiện - EUCS). Cấu trúc dữ liệu EUCS là một mảng hai chiều với các phần tử trong mảng được định nghĩa như sau [16]:

$$EUCS[i, j] = TWU(\{i, j\}) \tag{3.8}$$

Bản chất của EUCS là một mảng hai chiều lưu trữ TWU của các cặp hạng mục i và j. Cấu trúc này có thể được hiện thực hoá dưới dạng bảng băm của một bảng băm và chỉ lưu trữ các $TWU(\{i, j\}) \neq 0$ để giảm thiểu lượng bộ nhớ cần sử dụng. Quá trình xây dựng EUCS rất nhanh và chỉ cần một lần quét CSDL. Tương tự như TWU đã được mở rộng để làm việc với hạng mục tổng quát ở **Định nghĩa 2.19**, cấu trúc EUCS

này cũng được mở rộng để tỉa các hạng mục tổng quát không tiềm năng khỏi không gian tìm kiếm.

Tính chất 3.4 (Chiến lược tỉa ứng viên EUCP). Chiến lược tỉa ứng viên không tiềm năng EUCP dựa vào các bộ giá trị trong cấu trúc EUCS và tính chất của TWU để thu gọn không gian tìm kiếm. Nếu một tập $\mathcal{X}$ gồm hạng mục có giá trị $TWU < minutil$ thì các mở rộng của $\mathcal{X}$ đều không phải là tập mục hữu ích cao [16].

Chiến lược này dựa trên quan sát thấy rằng thao tác mất chi phí nhiều nhất trong quá trình khai thác chính là thao tác kết các utility-list nhỏ hơn để tạo ra các utility-list của các tập mục lớn hơn. Giả sử m, n là độ dài của hai utility-list, chi phí cho việc thực hiện phép kết hai utility-list này trong trường hợp xấu nhất là $O(m \times n)$. Phép kết hai utility-list được cải tiến để giúp giảm chi phí này về $O(m \times \log_2 n)$ [67].

Tính đúng đắn của **Tính chất 3.6** trong việc thu gọn không gian tìm kiếm cũng đã được chứng minh tại công trình [16].

Bằng việc xác định các mở rộng không tiềm năng và loại bỏ sớm chúng thông qua EUCP giúp giảm thiểu đáng kể chi phí xây dựng utility-list của các mở rộng không tiềm năng này.

3.3 Mô hình song song hoá quá trình khai thác tập mục hữu ích cao đa mức trên CSDL phân cấp

Với sự phát triển nhanh của công nghệ bán dẫn, các vi xử lý ngày nay được cung cấp dưới dạng đa nhân. Trong đó, mỗi vi xử lý có thể chứa từ hai nhân trở lên để tăng tốc quá trình thực thi các tác vụ. Số nhân xử lý càng nhiều, việc thời gian cần thiết để xử lý các tác vụ càng nhanh. Để có thể tận dụng được sức mạnh của các nhân xử lý này, các giải thuật phía sau chương trình ứng dụng cần phải được triển khai ở dạng song song hoá. Tuy nhiên, việc song song hoá các giải thuật tuần tự là không hề đơn giản. Có nhiều phương pháp để thực hiện song song hoá [93], ví dụ như song song hoá thông qua tập lệnh vi xử lý (SIMD, MIMD), song song hoá thông qua cơ chế xử lý đa luồng [94], [95], v.v… Các hệ điều hành hiện đại ngày nay cũng cung cấp khả

năng cho phép các giải thuật được phép truy cập đến các tập lệnh, nhân xử lý để áp dụng các mô hình song song hoá. Nhiều các ứng dụng hiện nay đều được triển khai dưới dạng song song hoá, đặc biệt tận dụng mô hình xử lý đa luồng, đa nhân để tăng hiệu quả của ứng dụng. Các hệ thống dịch vụ web (Apache HTTP Server, Tomcat), các ứng dụng ray tracing (Blender, 3dsmax) và nhiều ứng dụng khác đều áp dụng mô hình xử lý song song. Đối với lĩnh vực khai thác dữ liệu, nhiều tiếp cận khai thác cũng đã được mở rộng theo hướng song song hoá để tăng hiệu quả.

Các bài toán khai thác tập mục hữu ích cao, một số tiếp cận song song hoá cũng đã được đề xuất [47], [48]. Tuy nhiên các giải thuật này cũng chỉ song song hoá ở mức đơn giản hoặc chưa tận dụng hiệu quả sức mạnh xử lý của các vi xử lý đa nhân. Đối với bài toán khai thác tập mục hữu ích cao trên CSDL phân cấp xuất hiện gần đây, chưa có tiếp cận nào được đề xuất với chiến lược song song hoá nhằm giảm thiểu thời gian khai thác rất lớn của bài toán này.

Trong quá trình nghiên cứu, một số nhận định sau được rút ra như sau:

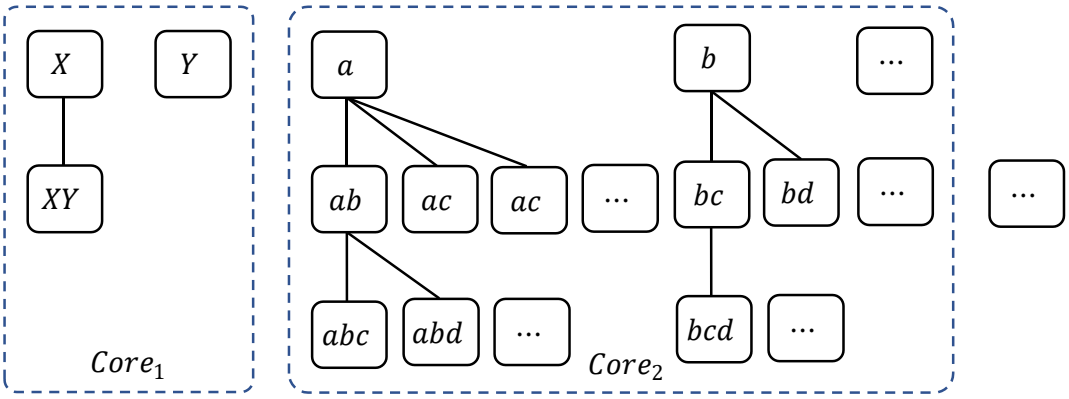
- Quá trình khai thác tập mục hữu ích cao trên CSDL phân cấp chủ yếu diễn ra dựa trên taxonomy của CSDL.
- Đối với bài toán khai thác tập mục hữu ích cao đa mức, các tập mục khai thác là độc lập theo từng cấp trên taxonomy. Các tập mục thuộc một cấp sẽ chỉ chứa các hạng mục ở cấp đó.
- Kích thước taxonomy càng lớn (độ sâu và chiều rộng mỗi level), thời gian khai thác càng tăng.

3.4 Khai thác song song các tập mục hữu ích cao đa mức từ CSDL phân cấp

3.4.1 Giải thuật MCML-Miner

Từ các nhận định trên, giải thuật đầu tiên được đề xuất để giải quyết bài toán này là giải thuật MCML-Miner (Multi-Core Multi-Level high utility itemset Miner). Đây là giải thuật đầu tiên triển khai mô hình song song cho bài toán khai thác tập mục hữu ích cao đa mức trên CSDL phân cấp. Mô hình song song hoá được lựa chọn áp dụng

là mô hình xử lý đa nhân. Giải thuật MCML-Miner được mở rộng dựa trên giải thuật tuần tự MLHUI-Miner [40]. Giải thuật này cho phép trích xuất đồng thời tập mục MLHUI từ CSDL giao dịch có sự phân cấp các hạng mục. Hình 3.4 minh hoạ không gian tìm kiếm trong quá trình khai thác của MCML-Miner trên một taxonomy có 2 cấp và CSDL được cho tại Bảng 3.1. Giải thuật này giải quyết cơ bản yêu cầu giảm thiểu thời gian khai thác thông qua việc song song hoá giai đoạn chính của giải thuật MLHUI-Miner. Tại đây, quá trình khai thác tại mỗi cấp của taxonomy được diễn ra độc lập và sẽ được xử lý trên mỗi nhân của vi xử lý đa nhân. Taxonomy có chiều sâu càng lớn kết hợp với các vi xử lý có nhiều nhân sẽ càng làm tăng hiệu quả của quá trình khai thác.



Hình 3.4. Không gian tìm kiếm được song song hoá của mô hình khai thác [CT2]

Ngoài việc áp dụng mô hình xử lý song song, MCML-Miner cũng áp dụng chiến lược tỉa ứng viên dựa trên chiến lược EUCP để giảm bớt số lượng các ứng viên cần xét, qua đó giảm chi phí xây dựng các utility-list không cần thiết. Chi tiết về nguyên lý, cách thức hoạt động và tỉa của chiến lược EUCP được trình bày trong [16]. Chiến lược EUCP cùng với cấu trúc dữ liệu EUCS được tóm tắt và nêu lại trong luận án này tại **Định nghĩa 2.20**.

Để giải quyết bài toán khai thác song song các tập mục hữu ích cao đa mức từ CSDL phân cấp, luận án đề xuất một tiếp cận mở rộng từ giải thuật MLHUI-Miner [41] kết hợp với chiến lược song song hoá đa nhân, giải thuật có tên gọi MCML-Miner. Các đặc điểm của giải thuật MCML-Miner gồm:

- Là giải thuật một pha mở rộng dựa trên MLHUI-Miner [41] nhằm khai thác tập mục hữu ích cao đa mức dựa trên taxonomy [41].
- MCML-Miner kế thừa các đặc điểm của MLHUI-Miner như:
    - Hoạt động trên CSDL phân cấp hoặc cả CSDL thông thường.
    - Tỉa ứng viên với chiến lược EUCP.
- Áp dụng **Định nghĩa 2.6** để triển khai trên các CSDL có độ hữu ích thay đổi [17].
- MCML-Miner hoạt động đúng đắn vì kế thừa các chiến lược tỉa ứng viên hiệu quả và đã được chứng minh như tỉa ứng viên dựa trên độ hữu ích còn lại, EUCP.
- Giảm thời gian khai thác thông qua mô hình song song hóa đa nhân ở mỗi cấp của taxonomy.

Mã giả của giải thuật MCML-Miner được trình bày tại Giải thuật 3.1.

Input:	CSDL $\mathbb{D}$, taxonomy τ, ngưỡng $minutil$
Output:	$\mathcal{R}$ tập các tập mục hữu ích cao đa mức
1	$\mathcal{R} \leftarrow \emptyset$; $EUCS \leftarrow \emptyset$;
2	Quét $\mathbb{D}$ sử dụng τ để xây dựng tập $\mathbb{I}$, $\mathbb{G}$ và xác định TWU của các hạng mục $i \in \mathbb{I}$.
3	Quét $\mathbb{D}$ sử dụng τ để tính TWU cho các hạng mục tổng quát $g \in \mathbb{G}$.
4	Tỉa các hạng mục chuyên biệt: $\mathbb{I}^+ \leftarrow \{i \in \mathbb{I} \mid TWU(i) \geq minutil\}$.
5	Tỉa các hạng mục tổng quát: $\mathbb{G}^+ \leftarrow \{g \in \mathbb{G} \mid TWU(g) \geq minutil\}$.
6	Xây dựng $EUCS$ cho các hạng mục thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$
7	**FOR EACH** level $l \in \tau$ **DO PARALLEL**
8	$\mathbb{IG}^+ \leftarrow$ các hạng mục tiềm năng thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$ tại cấp l.
9	$\mathcal{R}_l \leftarrow$**FHM_Search** $(\emptyset, \mathbb{IG}^+, \emptyset, minutil, EUCS)$;
10	**END**
11	**RETURN** $\mathcal{R} \leftarrow \cup_{l \in \tau} \mathcal{R}_l$.

Giải thuật 3.1 Giải thuật MCML-Miner

Giải thuật 3.1 cho thấy MCML-Miner hoàn toàn tương tự MLHUI-Miner. Điểm khác biệt duy nhất giữa MCML-Miner bắt đầu từ dòng 7 đến 11. Trong đó mỗi cấp thuộc taxonomy τ được gán cho một nhân của vi xử lý để tiến hành khai thác song song, giảm thiểu tổng thời gian khai thác của bài toán. Tại mỗi mức $l \in \tau$, hàm

FHM_Search [41] sẽ được gọi ứng với các tham số đầu vào cần thiết để quét không gian tìm kiếm ứng với mức l.

3.4.2 Độ phức tạp của giải thuật MCML-Miner

Gọi $\theta = |\tau|$ là chiều sâu của taxonomy, ρ là số lượng nhân của vi xử lý. Quá trình khai thác sẽ diễn ra đồng thời nếu như $\theta \leq \rho$. Ngược lại, các mức chưa được xử lý sẽ được đưa vào một hàng chờ có dạng "*đến trước, phục vụ trước*" để lần lượt gán tiếp tục vào các nhân xử lý nhàn rỗi.

Độ phức tạp của MCML-Miner là hoàn toàn tương tự MLHUI-Miner vì MCML-Miner kế thừa trực tiếp từ giải thuật MLHUI-Miner và kết hợp chiến lược xử lý song song dựa trên cấu trúc taxonomy.

Một cách chi tiết, gọi m là số lượng giao dịch trong CSDL phân cấp $\mathbb{D}$, w là chiều dài trung bình của các giao dịch trong $\mathbb{D}$ và l là số lượng các tập mục trong toàn bộ không gian tìm kiếm, độ phức tạp về mặt thời gian của MLHUI-Miner trong trường hợp xấu nhất xấp xỉ $O(\theta \times m \times w \times l)$. Có thể nhận thấy độ phức tạp của MLHUI-Miner là rất lớn. Bằng việc áp dụng mô hình song song hóa theo cấp của taxonomy thông qua giải thuật MCML-Miner, trong trường hợp tốt nhất với $\rho \geq \theta$, MCML-Miner có độ phức tạp về thời gian là tương đương giải thuật FHM [16], là giải thuật cơ sở của MLHUI-Miner.

3.4.3 Đánh giá hiệu năng của giải thuật MCML-Miner

Để đánh giá hiệu năng, giải thuật MCML-Miner được thực nghiệm trên một máy tính có cấu hình như sau: vi xử lý Intel® Xeon® E5-2678v3 (12 nhân vật lý, 24 nhân luận lý) với tốc độ xung nhịp cơ bản là 2.5Ghz; 128GB RAM DDR4 ECC, sử dụng hệ điều hành Windows 10 Pro Workstation.

Cho đến thời điểm thực hiện nội dung nghiên cứu này, bài toán khai thác tập mục hữu ích cao đa mức từ CSDL phân cấp vẫn chưa có một tiếp cận nào áp dụng mô hình song song hóa đa nhân. Vì vậy, MCML-Miner được so sánh với giải thuật tuần

tự ban đầu là MLHUI-Miner. Cả hai giải thuật đều được hiện thực hoá thông qua ngôn ngữ lập trình Java sử dụng môi trường phát triển Eclipse.

Kết quả được cho tại Hình 3.5 và Hình 3.6. MLHUI-Miner được lựa chọn đưa vào so sánh vì lý do MCML-Miner là phiên bản kế thừa trực tiếp từ MLHUI-Miner. Ngoài ra, cho đến nay chưa có một giải thuật nào triển khai mô hình xử lý song song cho bài toán này.

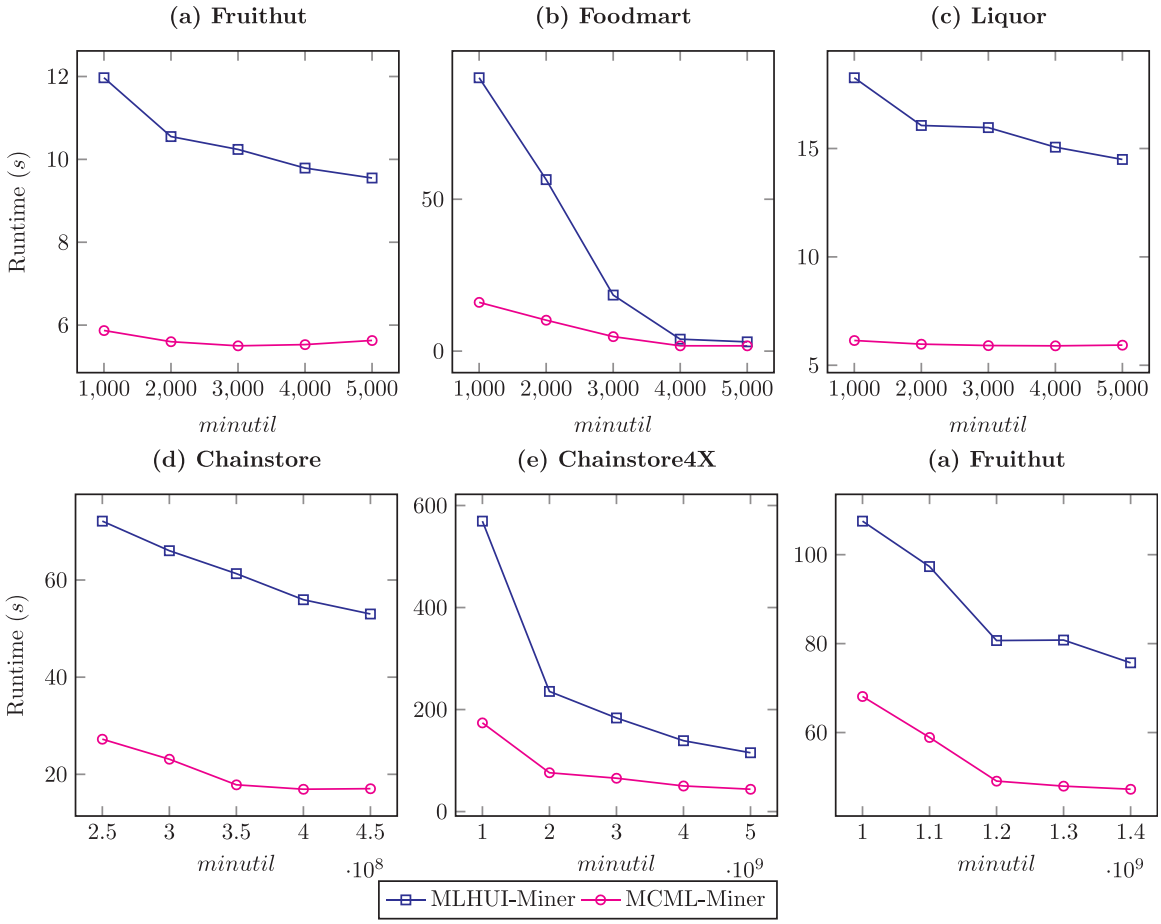
Bảng 3.6. Các CSDL thực nghiệm giải thuật MCML-Miner

Dataset	$\|\mathbb{D}\|$	$\|\mathbb{I}\|$	$\|\mathbb{G}\|$	$\|\tau\|$	T_{MAX}	T_{AVG}	**Mật độ**
Fruithut	181,970	1,265	43	4	36	3.58	$2.86E-03$
Foodmart	53,537	1,560	102	5	28	4.60	$2.82E-03$
Liquor	90,826	2,626	78	7	15	10.3	$1.03E-03$
Chainstore	1,112,949	46,086	9,674	9	170	7.2	$1.57E-04$
Chainstore4X	4,451,796	46,086	9,674	9	170	7.2	$1.57E-04$
SUSY	5,000,000	190	63	4	19	19.0	$1.00E-01$

Bảng 3.6 thể hiện chi tiết các CSDL phân cấp được sử dụng cho cả hai giải thuật MLHUI-Miner và MCML-Miner. Cột $|\tau|$ thể hiện số cấp trong taxonomy, đây cũng chính là số tác vụ song song mà MCML-Miner cấp phát để thực hiện quá trình khai thác. Xét CSDL Chainstore, CSDL này có taxonomy được tổng hợp với chín cấp, vì vậy khi thực hiện, giải thuật sẽ cấp phát chín tác vụ khai thác để thực hiện trên vi xử lý. CSDL Chainstore cũng được nhân bản lên bốn lần để tạo ra CSDL Chainstore4X với số lượng giao dịch lên đến hơn bốn triệu giao dịch. CSDL lớn nhất được thực nghiệm trong nội dung này là SUSY với năm triệu giao dịch và taxonomy được tổng hợp thành bốn cấp. Taxonomy của các CSDL phân cấp được tổng hợp có thể tải về tại liên kết: https://github.com/dzutrinh/MLC-Miner/tree/main/taxonomy.

Đối với thời gian khai thác, giải thuật MCML-Miner cho kết quả tốt hơn đáng kể so với MLHUI-Miner, đặc biệt trên các CSDL có số lượng giao dịch lớn như Chainstore,

Chainstore4X và SUSY. Đối với các CSDL như Fruitmart, Fruithut và Liquor, thời gian cải thiện lên đến 5.6 lần so với giải thuật MLHUI-Miner. Kết quả phép so sánh được cho tại Hình 3.5. Số lần thực nghiệm đã tiến hành ở mỗi CSDL là trên 10 lần và kết quả ghi nhận được lấy trung bình.

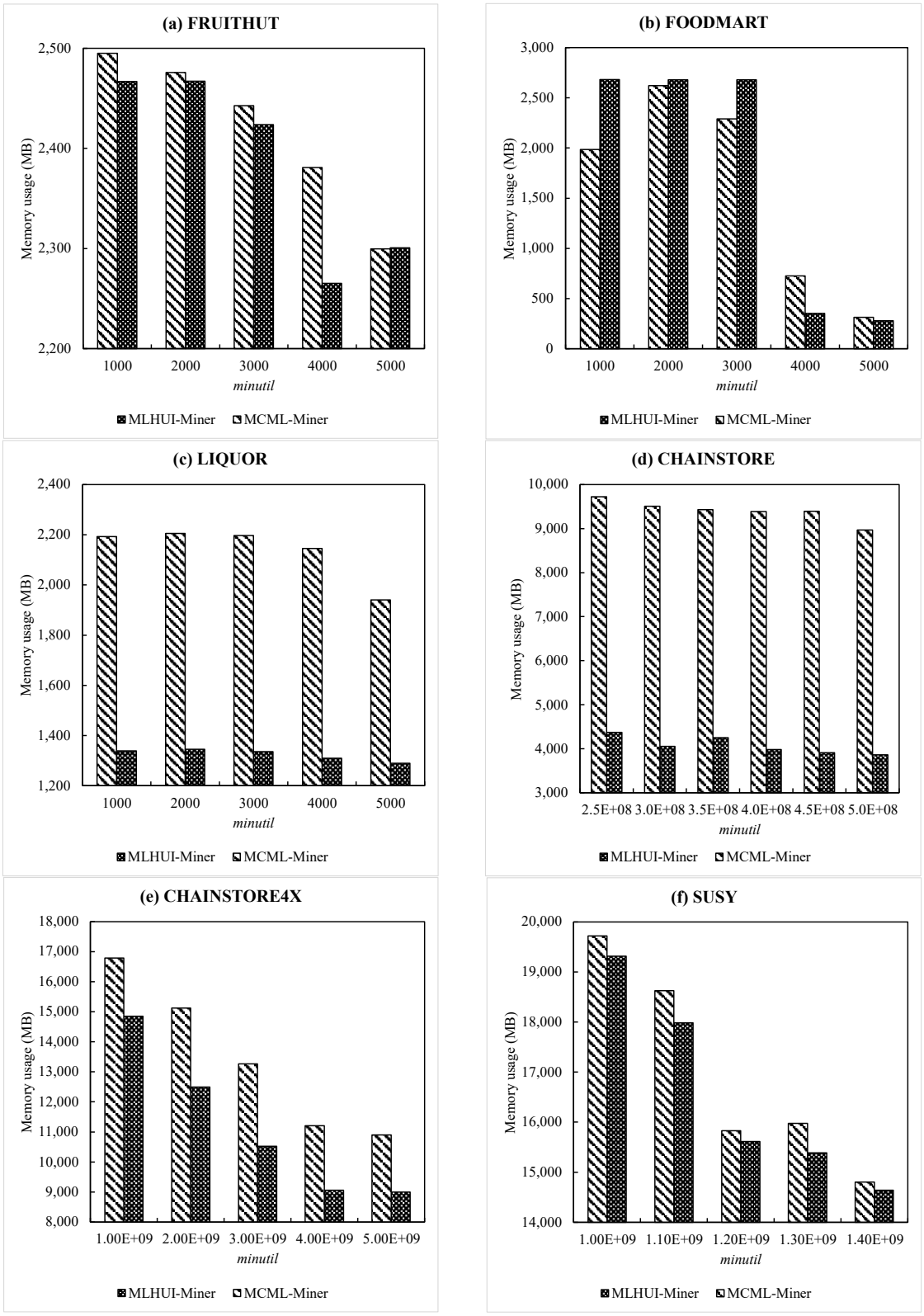


Hình 3.5. Thời gian khai thác của MCML-Miner và MLHUI-Miner [CT3]

Về mặt sử dụng bộ nhớ, MCML-Miner cho thấy lượng bộ nhớ mà giải thuật sử dụng là cao hơn so với MLHUI-Miner, tuy nhiên con số này không đáng kể trên các CSDL như Foodmart, Liquor và Fruithut. Lượng bộ nhớ mà MCML-Miner yêu cầu nằm dưới ngưỡng 4.0GB. Đối với các CSDL lớn, lượng bộ nhớ tối đa mà MCML-Miner yêu cầu là khoảng 19.8GB đối với CSDL SUSY (năm triệu giao dịch và bốn cấp taxonomy) tại $minutil = 1.0E + 9$.

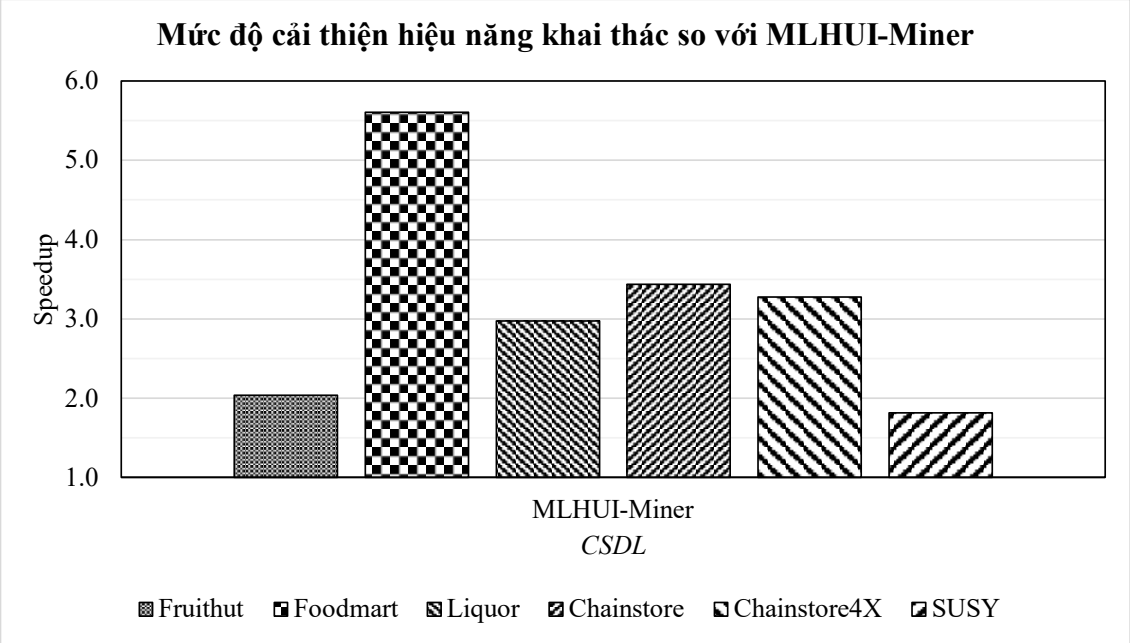
Tương tự phép so sánh về thời gian thực hiện, trên các CSDL lớn này, MLHUI-Miner cũng không thể triển khai. Bằng việc tích hợp các chiến lược tối ưu hóa việc sử dụng

bộ nhớ của MCML-Miner vào MLHUI-Miner, giải thuật này là khả thi trên các CSDL lớn như thể hiện tại Hình 3.6.



Hình 3.6. Lượng bộ nhớ sử dụng của MCML-Miner và MLHUI-Miner [CT3]

Qua thực nghiệm, có thể thấy tính hiệu quả của mô hình song song hoá đa nhân khi áp dụng trên giải thuật MCML-Miner. Quá trình khai thác được cải thiện hiệu quả lên đến 5.6 lần (Hình 3.7). Tuy nhiên, chiến lược song song hoá của giải thuật MCML-Miner là phụ thuộc vào số cấp của cấu trúc taxonomy.



Hình 3.7. Hiệu năng của MCML-Miner và MLHUI-Miner trên các CSDL [CT3]

Nếu ta áp dụng MCML-Miner lên các CSDL chỉ có một cấp taxonomy, hoặc áp dụng trên các CSDL thông thường, quá trình song song hoá sẽ bị triệt tiêu hoàn toàn. Hơn nữa, nếu CSDL có số cấp thấp hơn số nhân xử lý hiện có, giải thuật cũng không tận dụng các nhân còn lại đang nhàn rỗi vào quá trình khai thác. Ngoài ra, quá trình song song hoá chỉ diễn ra tại quá trình khai thác. Vì vậy, mặc dù hiệu năng khai thác được cải thiện, MCML-Miner vẫn chưa tận dụng hết khả năng của các vi xử lý đa nhân đưa vào khai thác.

3.5 Khai thác tập mục hữu ích cao đa mức - đóng từ CSDL phân cấp

Trong **Nội dung nghiên cứu 1** ở CHƯƠNG 2, bài toán khai thác tập mục hữu ích cao đóng đã được đặt ra, khảo sát và giải quyết hiệu quả. Tuy nhiên, việc song song hoá giải thuật là không khả thi do các ràng buộc dữ liệu xuất hiện trong quá trình kiểm tra bao đóng của các tập mục. Bên cạnh đó, Phần 3.3 và Phần 3.4 của chương

này cũng đã nêu ra tầm quan trọng của định dạng CSDL phân cấp cũng như đề xuất một mô hình song song hoá để áp dụng vào quá trình khai thác này.

Để nhằm mục đích thu gọn và bảo toàn tập kết quả của quá trình khai thác, biểu diễn đóng cũng được xem xét. Phần nội dung nghiên cứu này đề xuất một giải thuật để mở rộng bài toán khai thác tập mục hữu ích cao đóng nhằm có thể làm việc trên định dạng CSDL phân cấp và khả thi khi áp dụng mô hình xử lý song song, từ đó áp dụng mô hình song song hoá ở Phần 3.3 vào bài toán.

Giải thuật được đề xuất để giải quyết bài toán khai thác tập mục hữu ích cao đa mức-đóng dựa trên tiếp cận mà giải thuật CHUI-Miner đã áp dụng: dựa vào mô hình utility-list của giải thuật HUI-Miner [57], kết hợp với các kỹ thuật xác định tính đóng của các tập mục từ giải thuật CHUI-Miner [26]. Lý do chính để lựa chọn hướng tiếp cận này là để khảo sát các kỹ thuật, giải thuật mới nhằm đóng góp vào bài toán còn rất mới nhưng có ý nghĩa thực tế này trước khi tiến hành các nghiên cứu sâu hơn.

3.5.1 Kiểm tra nhanh tính đóng của một tập mục

Trong suốt quá trình khai thác các tập mục hữu ích cao đóng đa mức, mỗi khi một tập mục ứng viên được phát sinh, giải thuật cần phải tiến hành kiểm tra tính đóng của tập mục này. Số lượng ứng viên phát sinh càng lớn, chi phí tính toán cho thao tác này cũng càng gia tăng. Giải thuật CHUI-Miner [26] đề xuất việc kiểm tra tính đóng thông qua việc kiểm tra $Closure$ của các tập mục dựa trên **Định nghĩa 2.16**. Mã giả của thao tác này được cho tại Giải thuật 3.2.

Input:	Tập mục $\mathcal{Y}$ cùng với $PrevSet$ của $\mathcal{Y}$
Output:	TRUE nếu $\mathcal{Y}$ không đóng, FALSE nếu $\mathcal{Y}$ đóng

```
FUNC IsSubsumCheck(𝒴, PrevSet(𝒴))
  FOREACH tập mục 𝒥 ∈ PrevSet(𝒴) DO
    IF TidSet(𝒴) ⊆ TidSet(𝒥) THEN
      RETURN TRUE
  RETURN FALSE
```

Giải thuật 3.2 Giải thuật kiểm tra bao đóng của một tập mục từ CHUI-Miner

Giải thuật được trình bày khá đơn giản. Tuy nhiên, khi triển khai thực tế, giải thuật này phát sinh nhược điểm là chi phí tính toán lớn. Giả sử $k = |PrevSet(\mathcal{Y})|$, $u = |TidSet(\mathcal{Y})|$ và $v = |TidSet(\mathcal{J})|$, ta có thể dễ dàng nhận thấy độ phức tạp của giải thuật này là $O(k \times u \times \log_2 v)$. Khi phải liên tục thực hiện việc kiểm tra tính đóng của các tập ứng viên thông qua hàm $IsSubsumCheck$ nói trên, sẽ làm gia tăng đáng kể thời gian khai thác, và làm giảm đi hiệu suất của toàn bộ giải thuật.

Để khắc phục nhược điểm lớn này của CHUI-Miner, chương này đề xuất một kỹ thuật tối ưu hoá. Kỹ thuật này có tên gọi là *kiểm tra nhanh bao đóng hữu ích* (Rapid Utility Closure Checker – RUCK). Đây là đóng góp chính của toàn bộ giải thuật, giúp cho thời gian khai thác được giảm đáng kể, cải thiện hiệu năng khai thác.

Dựa trên các nhận định về giải thuật kiểm tra tính đóng các tập mục của CHUI-Miner là kém hiệu quả, chiến lược RUCK được đề xuất như sau:

- Chuyển đổi các utility-list truyền thống từ dạng danh sách thông thường trở thành dạng bảng băm với chi phí truy xuất các phần tử có độ phức tạp là hằng số, $O(1)$.
- Các phần tử của bảng băm này là một bộ $\langle TID, item \rangle$ trong utility-list, và được khởi tạo trong quá trình xây dựng các utility-list tương ứng.
- Vì trường TID của utility-list đã có trong bảng băm, vì vậy, ta không cần lưu trữ trường này thêm nữa. Điều này làm giảm chi phí về bộ nhớ của chiến lược.
- Quá trình kiểm tra tính đóng được thực hiện thông qua quá trình quét $TidSet$ của các tập mục, chính là tập khoá của các bảng băm.

Với các cải tiến nói trên, kỹ thuật này sẽ giúp giảm đáng kể độ phức tạp của quá trình kiểm tra, và từ đó hạ thấp thời gian cần thiết cho toàn bộ quá trình khai thác cũng như lượng bộ nhớ cần sử dụng, đặc biệt trên các CSDL có số lượng giao dịch lớn.

Với kỹ thuật RUCK được trình bày trong Giải thuật 3.3, ta có thể dễ dàng nhận thấy độ phức tạp của giải thuật này là $O(k \times u)$. Trước hết, nếu tập mục $\mathcal{Y}$ có độ dài lớn hơn tập mục $\mathcal{J}$ ta bỏ qua tập mục này và xét tập mục $\mathcal{J}$ kế đến, nếu tập khoá ($TidSet$)

của tập mục $\mathcal{Y}$ không chứa trong tập khoá của tập mục $\mathcal{J}$, ta kết luận $\mathcal{Y}$ không bị phủ bởi bất kỳ tập mục nào thuộc $PrevSet(\mathcal{Y})$.

Quá trình này thực hiện dựa trên các định nghĩa từ **Định nghĩa 2.6**, **Định nghĩa 2.7** và **Định nghĩa 2.8**.

Input:	Tập mục $\mathcal{Y}$ cùng với $PrevSet$ của $\mathcal{Y}$
Output:	TRUE nếu $\mathcal{Y}$ không đóng, FALSE nếu $\mathcal{Y}$ đóng

FUNC IsSubsumeRevisedCheck($\mathcal{Y}$, $PrevSet(\mathcal{Y})$)
  FOR EACH tập mục $\mathcal{J} \in PrevSet(\mathcal{Y})$ **DO**
    IF $|\mathcal{Y}| > |\mathcal{J}|$ **THEN CONTINUE**;
    IF $hashMap.keySet(\mathcal{Y}) \nsubseteq hashMap.keySet(\mathcal{J})$ **THEN**
      RETURN FALSE
  RETURN TRUE

Giải thuật 3.3 Giải thuật Rapid Utility Closure Checker – RUCK

3.5.2 Giải thuật MLC-Miner

Giải thuật CHUI-Miner để khai thác các tập mục hữu ích cao đóng trên định dạng CSDL thông thường, không phân cấp được đề xuất năm 2015 bởi Wu và đồng sự [26]. Giải thuật này là giải thuật một pha đầu tiên được giới thiệu để giải quyết bài toán khai thác này. CHUI-Miner được mở rộng dựa trên giải thuật HUI-Miner với một số cải tiến như cấu trúc EU-List, kỹ thuật kiểm tra tính đóng của tập mục.

Tuy nhiên, giải thuật này được đề xuất để áp dụng trên CSDL giao dịch thông thường chứ không thể hoạt động trên CSDL có sự phân cấp các hạng mục. Vì vậy, mục đích của nội dung nghiên cứu này là đề xuất một phiên bản cải tiến của CHUI-Miner, có tên gọi là MLC-Miner (Multi-Level Closed HUI Miner), để có thể khai thác được các tập mục hữu ích cao đóng từ CSDL phân cấp. Các đặc điểm của giải thuật MLC-Miner được tóm tắt như sau:

- Là giải thuật một pha.
- Kế thừa các chiến lược tỉa ứng viên từ CHUI-Miner [26].
- Áp dụng kỹ thuật tỉa ứng viên EUCP để thu nhỏ hơn nữa không gian tìm kiếm, tăng tốc quá trình khai thác [16].

- Tối ưu hoá quá trình kiểm tra tính đóng của các tập ứng viên thông qua chiến lược RUCK.
- Hoạt động được trên cả định dạng CSDL thông thường và định dạng CSDL phân cấp dựa trên taxonomy.
- Áp dụng **Định nghĩa 2.6** để áp dụng trên CSDL có độ hữu ích thay đổi [17].
- Giải thuật tập trung vào việc khai thác các tập mục chứa các hạng mục xuất hiện trên cùng một mức trừu tượng trong cấu trúc phân cấp.

Để tăng tốc quá trình khai thác, giảm thiểu việc quét đi quét lại CSDL nhiều lần và bỏ qua quá trình phát sinh ứng viên, MLC-Miner thực hiện hai lần quét CSDL để tính TWU và xây dựng cấu trúc utility-list cho các hạng mục. Các tập mục mở rộng cũng được liên kết với các utility-list tương ứng. Sau lần quét CSDL thứ hai, cấu trúc EU-List của các hạng mục/tập mục được xây dựng. Cấu trúc EU-List của một hạng mục/tập mục $\mathcal{X}$ này bao gồm các thành phần như sau: utility-list của $\mathcal{X}$, $\sup(\mathcal{X})$, mảng độ hữu ích đơn vị của $\mathcal{X}$, hai tập thứ tự $PostSet(\mathcal{X})$ và $PreSet(\mathcal{X})$ được dùng để mở rộng $\mathcal{X}$.

Input:	CSDL $\mathbb{D}$, taxonomy τ, ngưỡng $minutil$, cờ useEUCP
Output:	tập các tập mục hữu ích cao đóng, đa mức
1	$\mathcal{R} \leftarrow \emptyset$; $EUCS \leftarrow \emptyset$;
2	Quét $\mathbb{D}$ sử dụng τ để xây dựng tập $\mathbb{I}$ và $\mathbb{G}$ và xác định TWU của các hạng mục $i \in \mathbb{I}$.
3	Tính TWU cho các hạng mục tổng quát $g \in \mathbb{G}$.
4	Tỉa các hạng mục chuyên biệt: $\mathbb{I}^+ \leftarrow \{i \in \mathbb{I} \mid TWU(i) \geq minutil\}$.
5	Tỉa các hạng mục tổng quát: $\mathbb{G}^+ \leftarrow \{g \in \mathbb{G} \mid TWU(g) \geq minutil\}$.
6	Quét $\mathbb{D}$ một lần nữa để xây dựng EU-List, kết hợp xây dựng $EUCS$ cho các hạng mục thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$ nếu cờ $useEUCP =$ **TRUE**.
7	**FOR EACH** level $l \in \tau$ **DO**
8	$\mathbb{IG}^+ \leftarrow$ các hạng mục tiềm năng thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$ tại cấp l.
9	$\mathcal{R}_l \leftarrow$**Explore**$(\emptyset, \mathbb{IG}^+, \emptyset, minutil, EUCS)$;
10	**END**
11	**RETURN** $\mathcal{R} \leftarrow \cup_{l \in \tau} \mathcal{R}_l$.

Giải thuật 3.4 Mã giả giải thuật MLC-Miner

Mã giả của giải thuật **MLC-Miner** được thể hiện tại Giải thuật 3.4, với các tham số đầu vào gồm: CSDL phân cấp $\mathbb{D}$ và taxonomy τ, ngưỡng độ hữu ích tối thiểu $minutil$, cờ bật/tắt cơ chế tỉa ứng viên EUCP. Đầu ra của giải thuật là tập đầy đủ các tập mục hữu ích cao đóng đa mức từ CSDL phân cấp $\mathbb{D}$ và taxonomy τ.

Input:	tập mục $\mathcal{X}$, $PostSet(\mathcal{X})$, $PreSet(\mathcal{X})$, ngưỡng $minutil$, cấu trúc EUCS.
Output:	tập các tập mục hữu ích cao đóng, đa mức là mở rộng của $\mathcal{X}$.

```
1   WHILE PostSet(X) ≠ ∅ DO
2     ℐ ← min ≺ (PostSet(𝒳));
3     PostSet(𝒳) ← PostSet(𝒳)\ℐ;
4     𝒴 ← 𝒳 ∪ ℐ;
5     Xây dựng EU-List cho 𝒴;
6     PreSet(𝒴) ← PreSet(𝒳);
7     IF SumEU(𝒴) + SumRU(𝒴) ≥ minutil THEN
8       IF NOT IsSubsumeRevisedCheck(𝒴, PreSet(𝒴)) THEN
9         EUL(𝒴_C) ← EUL(𝒴)
10        PreSet(𝒴_C) ← PreSet(𝒴)
11        PostSet(𝒴_C) ← ∅
12        IF EUCS[𝒳, 𝒴] < minutil THEN CONTINUE;     // pruning
13        𝒴_C ←ComputeClosure(𝒴, PostSet(𝒴_C), PostSet(𝒳));
14        IF SumEU(𝒴_C) ≥ minutil THEN
15          OUTPUT 𝒴_C
16        END IF
17        RETURN Explore(𝒴_C,PostSet(𝒴_C),PreSet(𝒴_C),
                         minutil,EUCS);
18        PreSet(𝒳) ← PreSet(𝒳) ∪ ℐ
19      END IF
20    END IF
21  END WHILE
```

Giải thuật 3.5 Mã giả quá trình tìm kiếm theo chiều sâu của giải thuật MLC-Miner

Đầu tiên giải thuật MLC-Miner khởi tạo tập mục kết quả và cấu trúc EUCS rỗng. Tiếp theo, nó thực hiện bước quét CSDL lần đầu để xây dựng các tập $\mathbb{I}$ và $\mathbb{G}$, đồng thời tính luôn TWU của các hạng mục chuyên biệt $i \in \mathbb{I}$. Từ các TWU của các hạng mục chuyên biệt đã có, thực hiện việc xác định TWU cho các hạng mục tổng quát

$g \in \mathbb{G}$. Từ các thông tin đã có, giải thuật tiến hành tỉa các hạng mục đơn chuyên biệt và tổng quát không tiềm năng khỏi các giao dịch, chỉ giữ lại các hạng mục tiềm năng $\mathbb{I}^+$ và $\mathbb{G}^+$. Quá trình này diễn ra tại dòng 4 và 5 của giải thuật.

Giải thuật tiến hành quét CSDL lần thứ 2 để xây dựng cấu trúc EU-List như đã nêu, đồng thời xây dựng cấu trúc EUCS cho các hạng mục thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$nếu như cờ *useEUCP* được bật. Đến đây, giải thuật đã có đầy đủ các thông tin cần thiết cho quá trình khai thác. Quá trình khai thác diễn ra độc lập tại từng mức l thuộc taxonomy τ. Tại mỗi mức, tập con các hạng mục đơn tiềm năng tại mức l đó sẽ được trích xuất vào tập $\mathbb{IG}^+$ và dùng chúng để mở rộng không gian tìm kiếm. Các tham số cần thiết này sẽ được truyền vào thủ tục $Explore$ để tiến hành quét không gian tìm kiếm tại mức l để phát hiện các tập mục hữu ích cao đóng tại mức đó. Kết quả trả về sẽ được giữ tại tập $\mathcal{R}_l$. Sau cùng, sau khi tất cả các mức trong τ đã được quét xong, kết quả của toàn bộ giải thuật sẽ được tổng hợp lại và lưu trữ trong tập kết quả $\mathcal{R}$.

Thủ tục tìm kiếm theo chiều sâu để quét không gian tìm kiếm của bài toán là $Explore$ được dựa chủ yếu từ giải thuật CHUI-Miner [26] với các mở rộng để hoạt động trên các mức khác nhau của taxonomy và kết hợp chiến lược tỉa ứng viên EUCP. Thủ tục này nhận tham số đầu vào gồm: tập mục $\mathcal{X}$ cần mở rộng, các tập $PostSet(\mathcal{X})$ và $PreSet(\mathcal{X})$, ngưỡng $minutil$ và cấu trúc $EUCS$ để tỉa ứng viên. Nếu $EUCS = \emptyset$, EUCP xem như bị vô hiệu hoá. Thủ tục $Explore$ trả về tập đầy đủ các tập mục hữu ích cao đóng là mở rộng của $\mathcal{X}$ tại mức l.

Mã giả của thủ tục này được trình bày ở Giải thuật 3.5. Thủ tục con được triệu gọi trong thủ tục **Explore** là **ComputeClosure** là tương tự giải thuật CHUI-Miner và có thể tham khảo tại [26].

Điểm khác biệt giữa giải thuật CHUI-Miner ban đầu được trình bày trong [26] và giải thuật MLC-Miner (Giải thuật 3.5) tại dòng 8 và dòng 12. Tại dòng 8, kỹ thuật tối ưu hoá cho việc kiểm tra tập đóng được thực hiện thông qua hàm **IsSubsumeRevisedCheck** thể hiện ở Giải thuật 3.3. Đối với dòng 12, tại đây chiến

lược EUCP được áp dụng để tỉa các mở rộng không tiềm năng của tập mục $\mathcal{X}$ khỏi không gian tìm kiếm.

3.5.3 Độ phức tạp của giải thuật MLC-Miner

MLC-Miner được thiết kế dựa trên giải thuật CHUI-Miner cùng với hàng loạt các mở rộng, cải tiến để có thể triển khai trên CSDL phân cấp và rút ngắn thời gian khai thác.

Gọi $\theta = |\tau|$ là chiều sâu tối đa của cấu trúc taxonomy, $m = |\mathbb{D}|$, w là độ dài trung bình các giao dịch trong $\mathbb{D}$, l là số tập mục trong không gian tìm kiếm của bài toán. Vậy, độ phức tạp về mặt thời gian của MLC-Miner trong trường hợp xấu nhất được xác định như sau: $O(\theta \times m \times w \times l)$.

Trong đó, l có thể được xem là số tập mục ứng viên cần phải xét. Ứng với mỗi ứng viên, ta cần xét tính đóng của các tập mục này sử dụng chiến lược RUCK với chi phí là $O(k \times u)$. Như đã trình bày ở Giải thuật 3.3, với $k = |PrevSet(\mathcal{Y})|$, $u = |TidSet(\mathcal{Y})|$.

Về tổng quát, có thể nhận thấy độ phức tạp của MLC-Miner là tương đồng với MLHUI-Miner. Điều này là do các giải thuật khai thác theo chiều sâu đều dựa trên nền tảng giải thuật HUI-Miner [57]. Mặc dù các tiếp cận sau đó như FHM [16], CHUI-Miner [27,31], CLS-Miner [31], v.v… đều đề xuất và áp dụng chiến lược cải thiện hiệu quả khai thác, trong trường hợp xấu nhất, độ phức tạp của các tiếp cận đều tương đương giải thuật cơ sở là HUI-Miner [57].

3.5.4 Thực nghiệm

3.5.4.1 Môi trường thực nghiệm

Phần nội dung này mô tả cách thức thực hiện các thực nghiệm để đánh giá hiệu năng của giải thuật được đề xuất. Các phép thực nghiệm đều được thực hiện trên cùng một máy tính sử dụng vi xử lý Intel® Xeon® E5-2678 v3 64-bit với tốc độ xung cơ bản là 2.5 Ghz. Dung lượng bộ nhớ RAM là 128GB DDR4 ECC. Nền tảng hệ điều hành thực hiện các thực nghiệm là Microsoft Windows 10 Pro Workstation.

Giải thuật đã đề xuất cùng với các giải thuật sử dụng trong quá trình thực nghiệm được hiện thực trên nền ngôn ngữ lập trình Java và triển khai trên nền Java SDK 8.0 thông qua môi trường phát triển Eclipse.

3.5.4.2 Phương pháp đánh giá

Tính đến thời điểm thực hiện nghiên cứu này, MLC-Miner là giải thuật đầu tiên được đề xuất để thực hiện việc khai thác tập mục hữu ích cao đa mức đóng. Vì vậy, để so sánh, đánh giá hiệu năng giải thuật MLC-Miner, giải thuật MLC-Miner với chiến lược EUCP được vô hiệu hoá, ký hiệu MLC-Miner(nop) và giải thuật CHUI-Miner* sẽ được sử dụng làm baseline để so sánh với giải thuật MLC-Miner có sử dụng EUCP (ký hiệu MLC-Miner).

Giải thuật CHUI-Miner* cũng được đưa vào so sánh. Lưu ý rằng, giải thuật CHUI-Miner khi áp dụng lên các CSDL phân cấp sẽ không quan tâm đến cấu trúc taxonomy. Vì vậy, nếu đưa vào so sánh với MLC-Miner và MLC-Miner(nop) sẽ không khách quan. Vì vậy, CHUI-Miner cũng được mở rộng để có thể hoạt động được trên CSDL phân cấp và cũng được sử dụng như một baseline để đánh giá. Giải thuật này được ký hiệu là CHUI-Miner*. CHUI-Miner* cũng được thực nghiệm để bảo đảm rằng giải thuật cho ra các kết quả tương đương với MLC-Miner.

Các tiêu chí để so sánh trong phần thực nghiệm này bao gồm:

- ***Thời gian khai thác:*** Tính từ lúc giải thuật khởi chạy cho đến khi kết thúc (và kết xuất được tập kết quả trả về), hoặc xuất hiện lỗi như tràn bộ nhớ, hoặc thời gian thực hiện quá dài trong khi các giải thuật so sánh khác đã có kết quả.
- ***Bộ nhớ sử dụng:*** Lượng bộ nhớ sử dụng trung bình của giải thuật trong toàn bộ quá trình thực thi.
- ***Số lượng tập ứng viên được tỉa:*** so sánh số lượng các tập ứng viên không tiềm năng được tỉa bởi chiến lược EUCP.

Quá trình tính toán, thu thập số liệu được triển khai bằng cách cho thực hiện các giải thuật mười lần ở nhiều ngưỡng $minutil$ khác nhau, sau đó xác định giá trị trung bình thu nhận được. Các hàm thư viện chuẩn của Java được sử dụng để ghi nhận thời gian

thực hiện (đơn vị là giây – s) và lượng bộ nhớ đã cấp phát (đơn vị là megabyte – MB). Các tiêu chí đánh giá này có giá trị càng thấp càng thể hiện hiệu quả tốt hơn.

3.5.4.3 Cơ sở dữ liệu thực nghiệm

Các thực nghiệm được triển khai trên những CSDL giao dịch có phân cấp hạng mục chuẩn thường hay được sử dụng trong các báo cáo khoa học về bài toán khai thác tập mục hữu ích cao đa mức. Điều này giúp đánh giá hiệu năng của các giải thuật được thực nghiệm. Các CSDL phân cấp bao gồm Liquor, Fruithut và Foodmart. Trong đó, CSDL Fruithut và Liquor cùng với taxonomy được cung cấp tại thư viện khai thác dữ liệu mã nguồn mở SPMF [88]. CSDL Foodmart cùng với taxonomy gốc được cung cấp tại https://github.com/arunkjn/foodmart-mysql. CSDL và taxonomy này được biến đổi từ dạng CSV được cung cấp thành định dạng văn bản thô theo chuẩn của thư viện SPMF để tiện trong việc nạp và xử lý các CSDL.

Bảng 3.7. Các CSDL thực nghiệm giải thuật MLC-Miner

Dataset	$\|\mathbb{D}\|$	$\|\mathbb{I}\|$	$\|\mathbb{G}\|$	$\|\boldsymbol{\tau}\|$	$\boldsymbol{T_{MAX}}$	$\boldsymbol{T_{AVG}}$	**Mật độ**
Fruithut	181,970	1,265	43	4	36	3.58	$2.86E-03$
Foodmart	53,537	1,560	102	5	28	4.60	$2.82E-03$
Liquor	90,826	2,626	78	7	15	10.3	$1.03E-03$
Chainstore	1,112,949	46,086	9,674	9	170	7.2	$1.57E-04$
Chess	3,196	75	30	3	37	37.0	$4.93E-01$
SUSY	5,000,000	190	63	4	19	19.0	$1.00E-01$

Các CSDL thực nghiệm nói trên đều là CSDL thu thập được từ các cửa hàng bán lẻ thực tế [88], cụ thể Liquor chứa các giao dịch từ các cửa hàng đồ uống tại tiểu bang Iowa, Hoa Kỳ; Fruithut chứa các giao dịch từ một cửa hàng trái cây ở Hoa Kỳ; và Foodmart là tập các giao dịch được ghi nhận từ một cửa hàng bán lẻ, cũng tại Hoa Kỳ.

Ngoài ba CSDL phân cấp với taxonomy thực nói trên, ba CSDL giao dịch là Chainstore, Chess và SUSY cũng được đưa vào thực nghiệm. Chainstore là một CSDL giao dịch kích thước trung bình và có mật độ thấp và số lượng hạng mục lớn, lên đến hơn 46 ngàn hạng mục.

Chess là một CSDL nhỏ tuy nhiên lại có mật độ lớn. Chiều dài tối đa và chiều dài trung bình của Chess là 37.0. Chess được cung cấp tại thư viện SPMF. Tuy nhiên CSDL này không có sẵn taxonomy, vì vậy một taxonomy được tổng hợp cho CSDL này với chiều sâu là 3 mức và có 30 hạng mục tổng quát. Việc đưa Chess vào thực nghiệm là nhằm đánh giá mức độ hiệu quả của các chiến lược tỉa ứng viên.

SUSY là một CSDL giao dịch lớn có số lượng giao dịch lên đến 5 triệu. Đây cũng là CSDL giao dịch có kích thước lớn nhất trong các CSDL được đưa vào thực nghiệm. Tuy nhiên, ba CSDL giao dịch Chainstore, Chess và SUSY đều không có một taxonomy thực. Vì vậy, ứng với mỗi CSDL, một taxonomy sẽ được tổng hợp với số mức khác nhau để đánh giá giải thuật MLC-Miner đã đề xuất.

Đặc điểm của các CSDL giao dịch sử dụng trong các thực nghiệm này được trình bày tại Bảng 3.7. Trong đó, $|\mathbb{D}|$ thể hiện kích thước của CSDL tính trên số lượng các giao dịch mà CSDL đó chứa. $|\mathbb{I}|$ thể hiện tổng số các hạng mục chuyên biệt trong CSDL tương ứng. $|\mathbb{G}|$ biểu thị số lượng hạng mục tổng quát. $|\tau|$ biểu thị chiều cao tối đa của cây phân cấp hạng mục trong CSDL. $\boldsymbol{T_{AVG}}$ cho biết độ dài trung bình của các giao dịch trong CSDL $\mathbb{D}$ và $\boldsymbol{T_{MAX}}$ cho biết độ dài tối đa của các giao dịch trong CSDL.

Cấu trúc taxonomy của tất cả các CSDL thực nghiệm ở Bảng 3.7 được cho tại địa chỉ: https://github.com/dzutrinh/MLC-Miner/tree/main/taxonomy.

3.5.4.4 Đánh giá về thời gian thực hiện

Kết quả thực nghiệm theo tiêu chí thời gian thực hiện của các giải thuật được cho tại Hình 3.8. Nhìn chung, trên tất cả các CSDL thực nghiệm, MLC-Miner đều cho kết quả khai thác tốt hơn hẳn so với MLC-Miner(nop) và vượt trội hơn hẳn CHUI-Miner* nhiều lần. Điều này cho thấy chiến lược EUCP có khả năng loại bỏ sớm một lượng

đáng kể các tập mục ứng viên không tiềm năng và làm giảm thời gian cần thiết để thực hiện việc khai thác. Ngoài ra, kỹ thuật tối ưu hoá việc kiểm tra tính đóng tập mục RUCK cũng thật sự giúp cho thời gian khai thác của cả MLC-Miner và MLC-Miner(nop) cải thiện rõ rệt so với CHUI-Miner*.

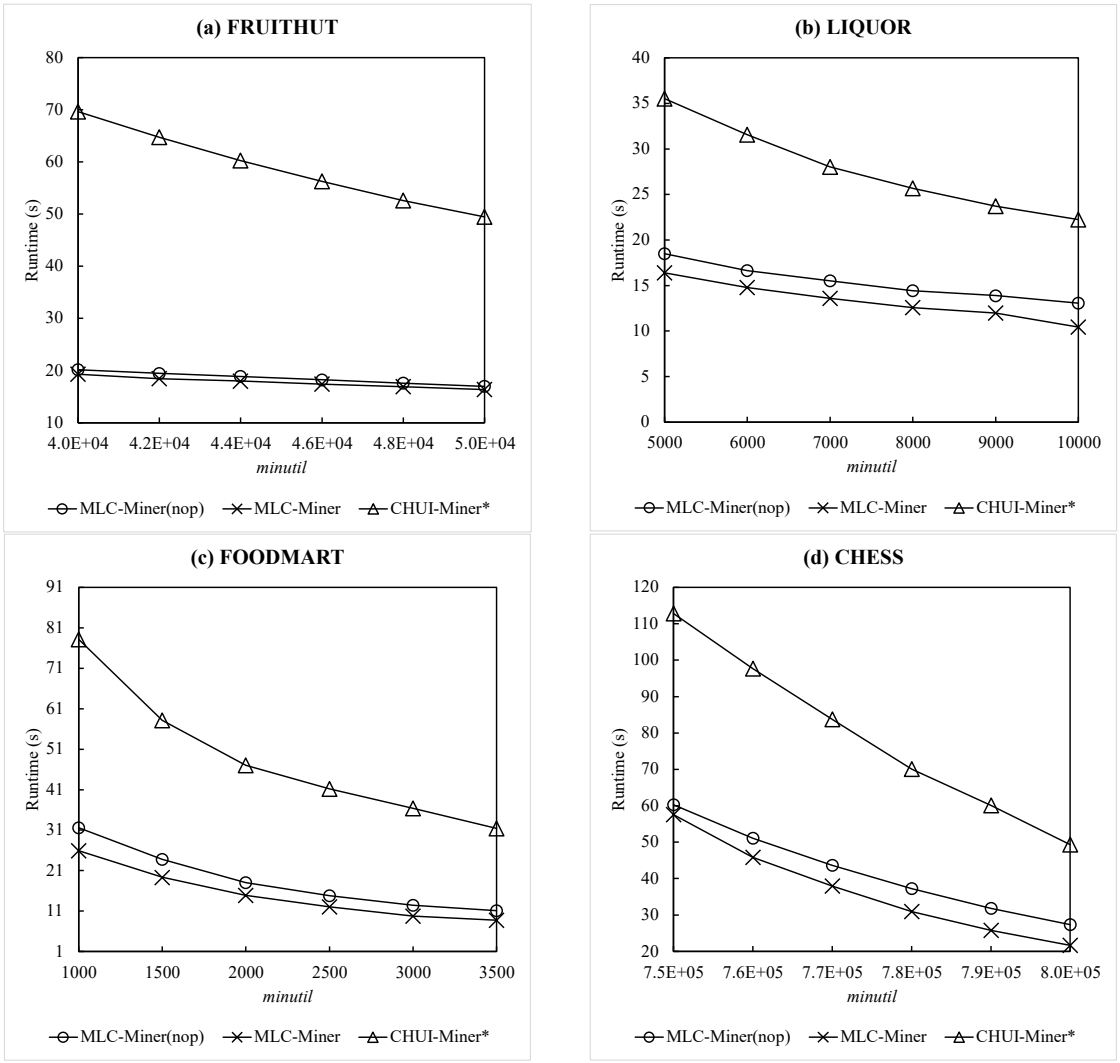
Cụ thể, với các CSDL dày như Chess hoặc SUSY, mức độ chênh lệch về thời gian thực thi giữa MLC-Miner và MLC-Miner(nop) là không lớn. Lý do chính là chiến lược EUCP không thực sự hiệu quả trên các CSDL dày. Kết quả ở Hình 3.8d và Hình 3.8f cho thấy thời gian thực thi của cả hai giải thuật là xấp xỉ với nhau. Đối với CSDL Chainstore, là một CSDL có số lượng hạng mục rất lớn, số lượng tập mục cần kiểm tra bao đóng của cả ba giải thuật cũng sẽ rất lớn và làm ảnh hưởng đến tốc độ thực thi của chúng. Tuy nhiên, giải thuật MLC-Miner và MLC-Miner(nop) với chiến lược RUCK, cả hai giải thuật đều có thời gian thực thi dưới 200 giây. Trong khi đó, CHUI-Miner* không thể hoàn tất quá trình khai thác trong giới hạn hai giờ. Vì vậy, CHUI-Miner* không xuất hiện trong Hình 3.8e.

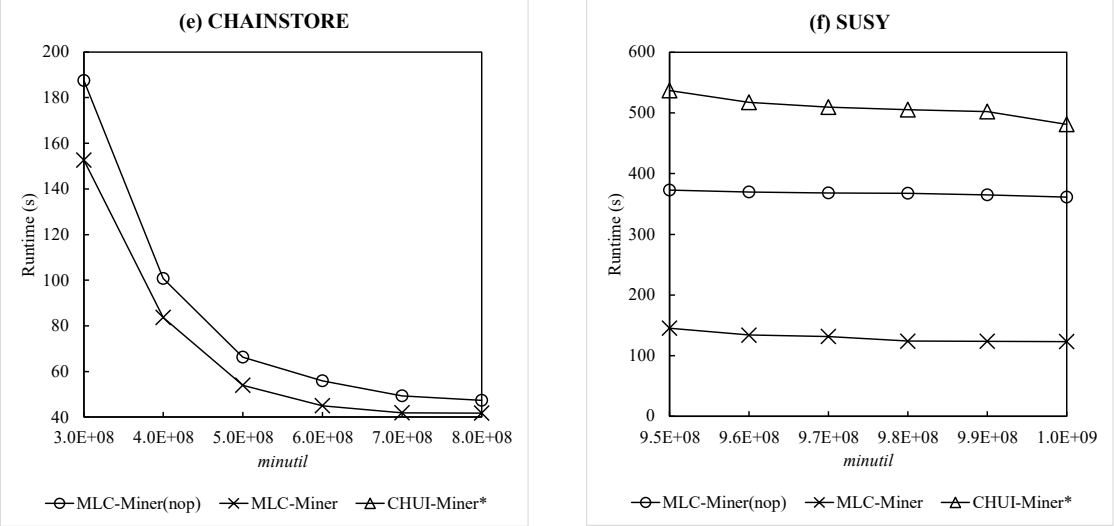
Trong CSDL này, hai giải thuật MLC-Miner và MLC-Miner(nop) có thời gian thực hiện xấp xỉ nhau. Nhưng MLC-Miner cho mức độ cải thiện về thời gian là tốt hơn MLC-Miner(nop) khi sử dụng chiến lược tỉa ứng viên EUCP. Điều này có thể giải thích được là do số lượng hạng mục chuyên biệt lẫn tổng quát của Chainstore rất lớn, vì vậy giải thuật cần phải thực hiện kiểm tra bao đóng trên nhiều tập mục ứng viên hơn. Vì MLC-Miner và MLC-Miner(nop) đều áp dụng cùng kỹ thuật RUCK, độ phức tạp về thời gian của việc kiểm tra này là tương đồng nhau. Tuy nhiên, do chiều sâu taxonomy của CSDL Chainstore lớn, dẫn đến các hạng mục tổng quát sẽ xuất hiện nhiều trong các giao dịch, nên hiệu quả tỉa ứng viên của EUCP là không cao.

Trong tất cả các CSDL thực nghiệm, CHUI-Miner* đều cho thời gian thực thi lớn hơn nhiều so với MLC-Miner(nop) và MLC-Miner, cao hơn đến gần 4 lần. Đặc biệt với CSDL Chainstore, CHUI-Miner* là không thể hoàn thành trong khoảng thời gian hai giờ. Đối với CSDL SUSY, CHUI-Miner* vẫn hoàn thành trên tất cả các ngưỡng $minutil$. Tuy nhiên, thời gian thực thi của giải thuật vẫn còn rất cao. Hơn nữa, mức

độ sử dụng bộ nhớ của giải thuật là cực kỳ cao. CHUI-Miner* yêu cầu lên đến 95% lượng bộ nhớ của hệ thống thực nghiệm để có thể thực hiện.

Bên cạnh đó, có thể nhận thấy được trong tất cả các thực nghiệm, số lượng hạng mục ảnh hưởng trực tiếp đến hiệu năng khai thác các tập mục hữu ích cao đa mức đóng vì chúng có liên quan đến quá trình kiểm tra bao đóng. Ngưỡng *minutil* càng thấp, số lượng các tập mục ứng viên càng lớn và càng cần nhiều thời gian hơn để kiểm tra. Vì vậy, bằng việc tích hợp chiến lược RUCK đã đề xuất, MLC-Miner và cả MLC-Miner(nop) đều cho thấy có khả năng làm việc được trên các CSDL lớn với hàng triệu giao dịch và hàng chục ngàn hạng mục.





Hình 3.8. So sánh thời gian thực hiện trên các CSDL thực nghiệm. [CT1]

3.5.4.5 Đánh giá về bộ nhớ sử dụng

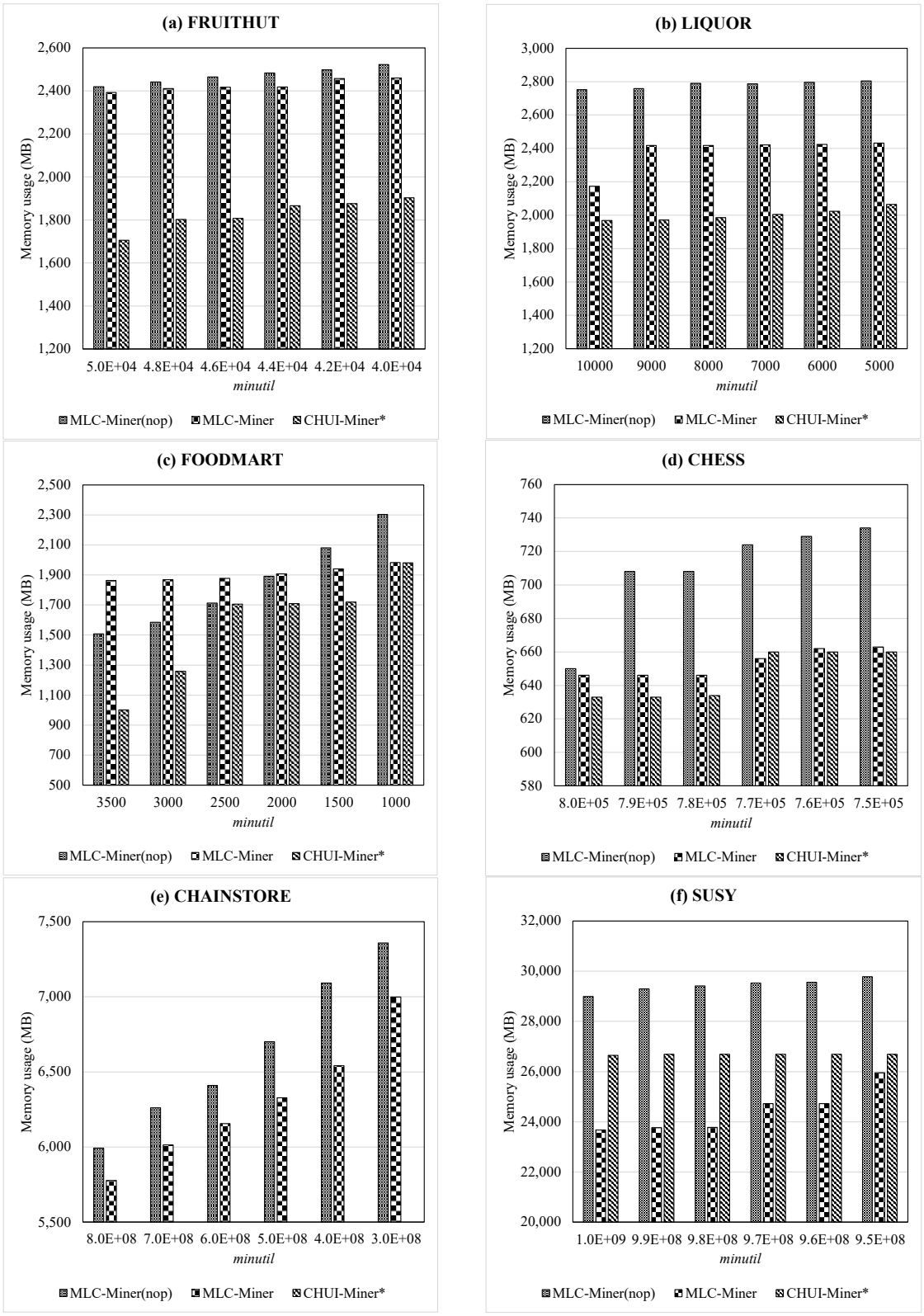
Lượng bộ nhớ sử dụng tối đa của các giải thuật MLC-Miner, MLC-Miner(nop) và CHUI-Miner* cũng được ghi nhận đồng thời trong quá trình đánh giá thời gian thực hiện. Các giá trị này cũng được trung bình hoá và thể hiện trong hình Hình 3.9.

Thực nghiệm cho thấy lượng bộ nhớ tối đa được cấp phát cho cả MLC-Miner(nop) và MLC-Miner đều thấp hơn nhiều so với CHUI-Miner*. Lưu ý rằng, tại CSDL Chainstore, vì CHUI-Miner* không thể hoàn thành trong giới hạn thời gian hai giờ, vì vậy Hình 3.9e không thể hiện thông tin của CHUI-Miner*.

Đối với các CSDL như Fruithut (Hình 3.9a), Liquor (Hình 3.9b), Foodmart (Hình 3.9c) và Chainstore (Hình 3.9e), chiến lược EUCP giúp MLC-Miner giảm thiểu đáng kể số lượng các tập ứng viên không tiềm năng. Vì vậy, đã kéo giảm lượng bộ nhớ cần thiết để lưu trữ các utility-list của chúng. Tuy nhiên, với MLC-Miner(nop) với chiến lược EUCP bị vô hiệu hoá, lượng bộ nhớ của giải thuật này là cao hơn hẳn so với MLC-Miner.

Đối với các CSDL dày như Chess (Hình 3.9d) và SUSY (Hình 3.9f), hiệu quả của EUCP là rất hạn chế. Vì vậy lượng bộ nhớ của cả MLC-Miner và MLC-Miner(nop) là xấp xỉ như nhau. Tuy nhiên, trong một số trường hợp, lượng bộ nhớ đã cấp phát

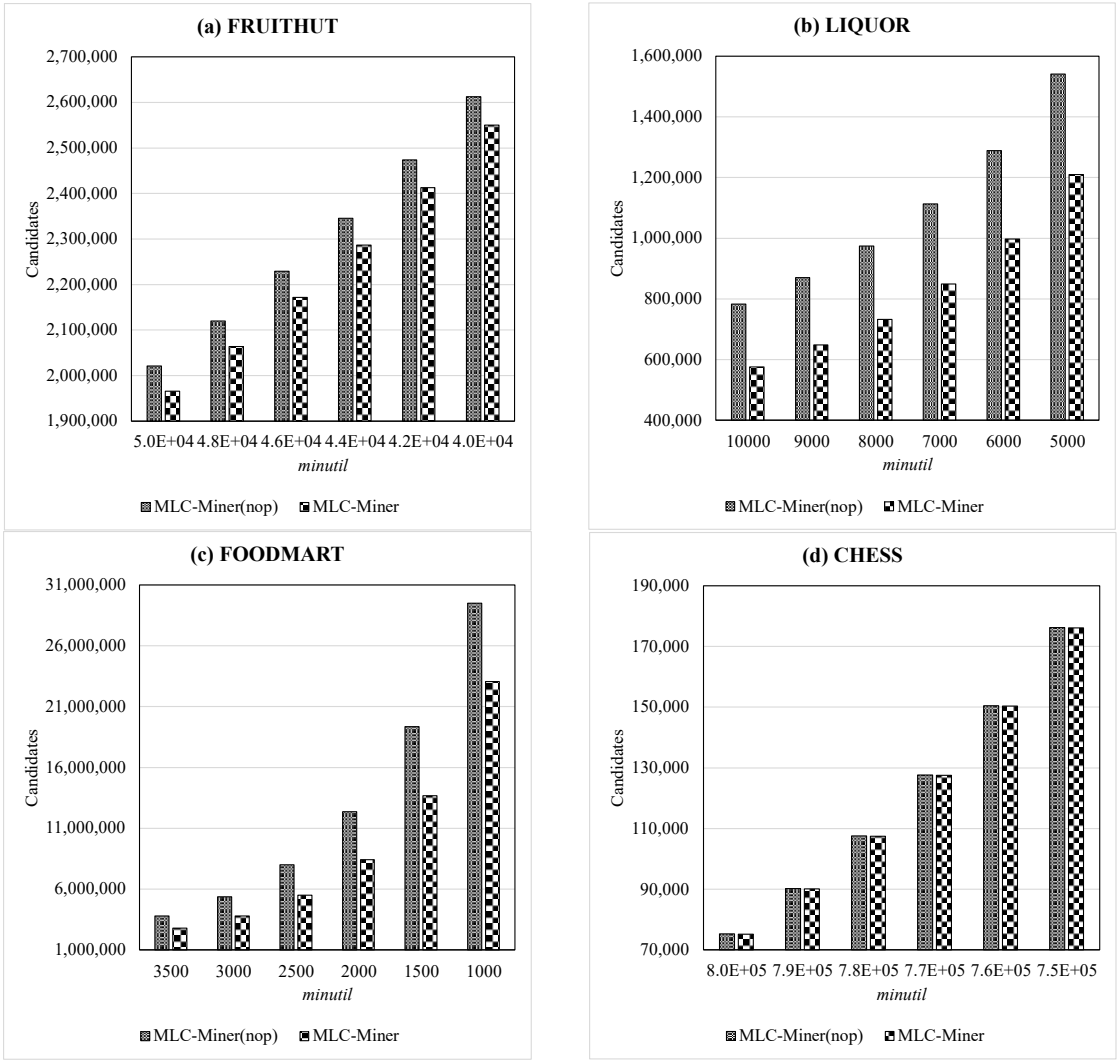
cho MLC-Miner lại cao hơn MLC-Miner(nop) do giải thuật cần phải có không gian để lưu trữ thêm cấu trúc EUCS.

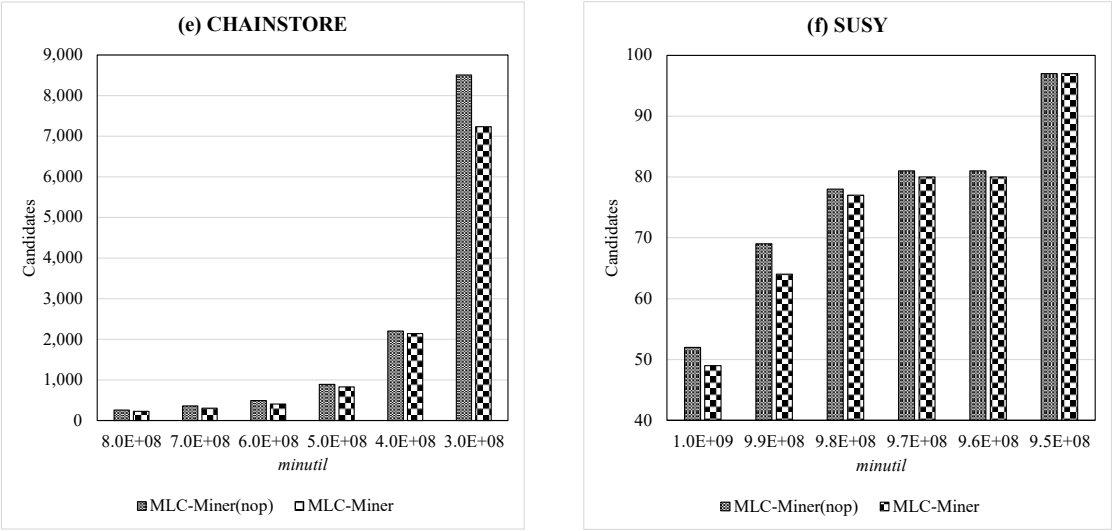


Hình 3.9. So sánh mức độ sử dụng bộ nhớ trên các CSDL thực nghiệm. [CT1]

Nhìn chung, lượng bộ nhớ lớn nhất mà MLC-Miner và MLC-Miner(nop) yêu cầu đều thấp hơn đáng kể so với CHUI-Miner*. Điều này có thể xác định là do CHUI-Miner* sử dụng cấu trúc dữ liệu utility-list truyền thống. Ngoài ra, do CHUI-Miner* không áp dụng bất kỳ chiến lược tỉa ứng viên hiệu quả nào ngoài việc sử dụng remaining utility, điều này cũng góp phần làm gia tăng lượng bộ nhớ cần thiết để lưu trữ các tập ứng viên không tiềm năng cùng với utility-list truyền thống của chúng. Điều này thể hiện qua CSDL SUSY (Hình 3.9f) với 5 triệu giao dịch.

3.5.4.6 So sánh hiệu quả tỉa ứng viên với EUCP





Hình 3.10. So sánh hiệu quả của EUCP trên các CSDL thực nghiệm. [CT1]

Trong phần thực nghiệm này, số lượng ứng viên không tiềm năng được tỉa với chiến lược EUCP cũng được so sánh giữa hai giải thuật là MLC-Miner và MLC-Miner(nop). Bên cạnh việc tỉa ứng viên với ngưỡng giới hạn trên remaining utility, MLC-Miner cũng áp dụng EUCP để tỉa ứng viên. Số lượng các tập ứng viên không tiềm năng được tỉa chủ yếu được đóng góp từ EUCP.

Hình 3.10 thể hiện số lượng tập mục ứng viên đã xét giữa MLC-Miner và MLC-Miner(nop) trên các CSDL thực nghiệm. Có thể thấy rằng trên hầu hết các CSDL như Fruithut (Hình 3.10a), Liquor (Hình 3.10b), Foodmart (Hình 3.10c) và Chainstore (Hình 3.10e), MLC-Miner có số lượng ứng viên cần xét ít hơn đáng kể so với MLC-Miner(nop). Số lượng ứng viên cần xét ít đi đồng nghĩa với việc giảm thiểu số lượng phép kiểm tra bao đóng, và cũng tương ứng với thời gian khai thác thấp hơn nhiều.

Đối với hai CSDL còn lại là Chess (Hình 3.10d) và SUSY (Hình 3.10f), số lượng ứng viên được xét giữa cả hai giải thuật MLC-Miner và MLC-Miner(nop) là tương đương nhau. Đặc biệt, đối với CSDL Chess, là CSDL có mật độ cao nhất trong sau CSDL đưa vào thực nghiệm, số lượng ứng viên được tỉa bớt bởi EUCP là rất ít.

Như đã trình bày trong phần đánh giá về thời gian khai thác cũng như mức độ sử dụng bộ nhớ ở trên, các CSDL có mật độ cao sẽ tác động đến hiệu quả tỉa ứng viên của

EUCP. Vì vậy, cũng làm giảm hiệu năng khai thác của MLC-Miner về mặt thời gian lẫn không gian so với MLC-Miner(nop).

3.6 Khai thác song song các tập mục hữu ích cao đa mức - đóng

3.6.1 Giải thuật PMLC-Miner

Giải thuật MLC-Miner đã giải quyết hiệu quả được bài toán khai thác tập mục hữu ích cao – đóng trên CSDL phân cấp, với thời gian và không gian được cải thiện tốt hơn hẳn so với giải thuật CHUI-Miner*. Tuy nhiên, giải thuật MLC-Miner vẫn là một giải thuật tuần tự và chưa tận dụng được khả năng tính toán của các vi xử lý đa nhân hiện nay. Vì vậy, phần nội dung nghiên cứu này sẽ tìm hiểu và triển khai mô hình xử lý song song đa nhân đã tìm hiểu ở trên vào giải thuật MLC-Miner với mong muốn cải thiện hơn nữa thời gian khai thác của giải thuật.

Với chiến lược song song hoá đã triển khai trên giải thuật MCML-Miner, cùng với không gian tìm kiếm của bài toán là gần như tương tự nhau, ta có thể áp dụng chiến lược song song hoá này vào giải thuật MLC-Miner để có thể cải thiện hiệu năng của bài toán.

Input: CSDL $\mathbb{D}$, taxonomy τ, ngưỡng $minutil$
Output: tập các tập mục hữu ích cao đóng, đa mức $\mathcal{R}$

1 $\mathcal{R} \leftarrow \emptyset$; $EUCS \leftarrow \emptyset$;
2 Quét $\mathbb{D}$ sử dụng τ để xây dựng tập $\mathbb{I}$ và $\mathbb{G}$ và xác định TWU của các hạng mục $i \in \mathbb{I}$.
3 Tính TWU cho các hạng mục tổng quát $g \in \mathbb{G}$.
4 Tỉa các hạng mục chuyên biệt: $\mathbb{I}^+ \leftarrow \{i \in \mathbb{I} \mid TWU(i) \geq minutil\}$.
5 Tỉa các hạng mục tổng quát: $\mathbb{G}^+ \leftarrow \{g \in \mathbb{G} \mid TWU(g) \geq minutil\}$.
6 Quét $\mathbb{D}$ một lần nữa để xây dựng EU-List, kết hợp xây dựng $EUCS$ cho các hạng mục thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$.
7 **FOR EACH** level $l \in \tau$ **DO PARALLEL**
8 $\mathbb{IG}^+ \leftarrow$ các hạng mục tiềm năng thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$ tại cấp l.
9 $\mathcal{R}_l \leftarrow$**Explore**$(\emptyset, \mathbb{IG}^+, \emptyset, minutil, EUCS)$;
10 **END**
11 **RETURN** $\mathcal{R} \leftarrow \cup_{l \in \tau} \mathcal{R}_l$.

Giải thuật 3.6 Mã giả giải thuật PMLC-Miner

Từ các nhận định trên, một giải thuật khai thác tập mục hữu ích cao đa mức – đóng từ CSDL phân cấp có tên gọi PMLC-Miner (Parallelized MLC-Miner) được đề xuất.

Giải thuật PMLC-Miner có các đặc điểm như sau:

- Là giải thuật một pha mở rộng dựa trên nền giải thuật MLC-Miner phục vụ cho việc khai thác tập mục hữu ích cao đa mức – đóng trên CSDL phân cấp.
- PMLC-Miner kế thừa các toàn bộ đặc trưng của MLC-Miner.
- Rút ngắn thời gian khai thác thông qua mô hình song song hóa đa nhân ở mỗi cấp của taxonomy.

Không gian tìm kiếm thể hiện tại Hình 3.4 cũng chính là không gian tìm kiếm của giải thuật PMLC-Miner. Mã giả của giải thuật PMLC-Miner được thể hiện tại Giải thuật 3.6.

Tương tự như mô hình song song hoá đã áp dụng tại giải thuật MCML-Miner, Giải thuật 3.6 cũng có cấu trúc tương đồng với MLC-Miner ở Giải thuật 3.4. Điểm khác biệt duy nhất giữa hai giải thuật này là từ dòng 7 đến dòng 11. Tại đoạn mã này, quá trình khai thác cũng sẽ được diễn ra song song, đồng thời sử dụng các nhân xử lý nhàn rỗi của CPU.

Các hàm, thủ tục còn lại được sử trong PMLC-Miner là tương tự hoàn toàn với giải thuật MLC-Miner (Giải thuật 3.2, Giải thuật 3.3, Giải thuật 3.4 và Giải thuật 3.5).

3.6.2 Độ phức tạp của giải thuật PMLC-Miner

Giải thuật song song PMLC-Miner, phiên bản song song hóa của MLC-Miner, áp dụng cùng một mô hình song song hóa tương tự MCML-Miner. Vì vậy độ phức tạp về mặt thời gian của giải thuật này cũng được xác định dựa trên phiên bản tuần tự là MLC-Miner mà đã được khảo sát với chi phí $O(\theta \times m \times w \times l)$.

Giả sử ρ là số nhân vi xử lý, trong trường hợp tốt nhất ($\rho \geq \theta$), độ phức tạp của giải thuật PMLC-Miner cũng sẽ được giảm về tiệm cận với độ phức tạp của giải thuật cơ sở là HUI-Miner. Điều này là nhờ vào quá trình song song hóa theo từng cấp của taxonomy đã được đề xuất ở mô hình của giải thuật MCML-Miner.

3.6.3 Thực nghiệm

Với giải thuật PMLC-Miner đã đề xuất ở trên, hiệu năng của giải thuật này được đánh giá thông qua các thực nghiệm trên nhiều CSDL phân cấp khác nhau.

Môi trường thực nghiệm của PMLC-Miner là hoàn toàn tương tự môi trường đã thực nghiệm giải thuật MLC-Miner: máy tính sử dụng vi xử lý Intel® Xeon® E5-2678 v3 64-bit với xung nhịp cơ sở là 2.5 Ghz. Dung lượng RAM là 128GB DDR4 ECC. Hệ điều hành được sử dụng là Microsoft Windows 10 Pro Workstation.

Giải thuật PMLC-Miner cũng được triển khai trên nền ngôn ngữ lập trình Java sử dụng môi trường phát triển Eclipse.

Để có tính khách quan trong thực nghiệm, giải thuật PMLC-Miner được áp dụng lại trên toàn bộ các CSDL phân cấp trong phần trình bày của giải thuật MLC-Miner (Bảng 3.7). Một số CSDL thực nghiệm có taxonomy được tổng hợp vì bản thân các CSDL này không có sẵn taxonomy. Các taxonomy này được cung cấp tại liên kết: https://github.com/dzutrinh/MLC-Miner/tree/main/taxonomy.

Giải thuật PMLC-Miner sẽ được đánh giá hiệu năng so với hai giải thuật là MLC-Miner, giải thuật mà PMLC-Miner kế thừa trực tiếp, CHUI-Miner* và CHUI-Miner**. Trong đó CHUI-Miner* là giải thuật CHUI-Miner được mở rộng để làm việc với CSDL phân cấp. Giải thuật CHUI-Miner** là phiên bản song song hóa của CHUI-Miner* sử dụng cùng mô hình song song hóa đã đề xuất tại mục 3.3, **Nội dung nghiên cứu 2**.

Tiêu chí đánh giá gồm hai yếu tố cơ bản trong các phép so sánh về hiệu năng: thời gian khai thác và lượng bộ nhớ tối đa được sử dụng trong quá trình khai thác. Các giải thuật lần lượt được cho thực hiện liên tục mười lần trên các CSDL thực nghiệm. Các số liệu về thời gian, bộ nhớ sử dụng được ghi nhận và trung bình hoá để thu được kết quả sau cùng.

3.6.3.1 Đánh giá về thời gian khai thác

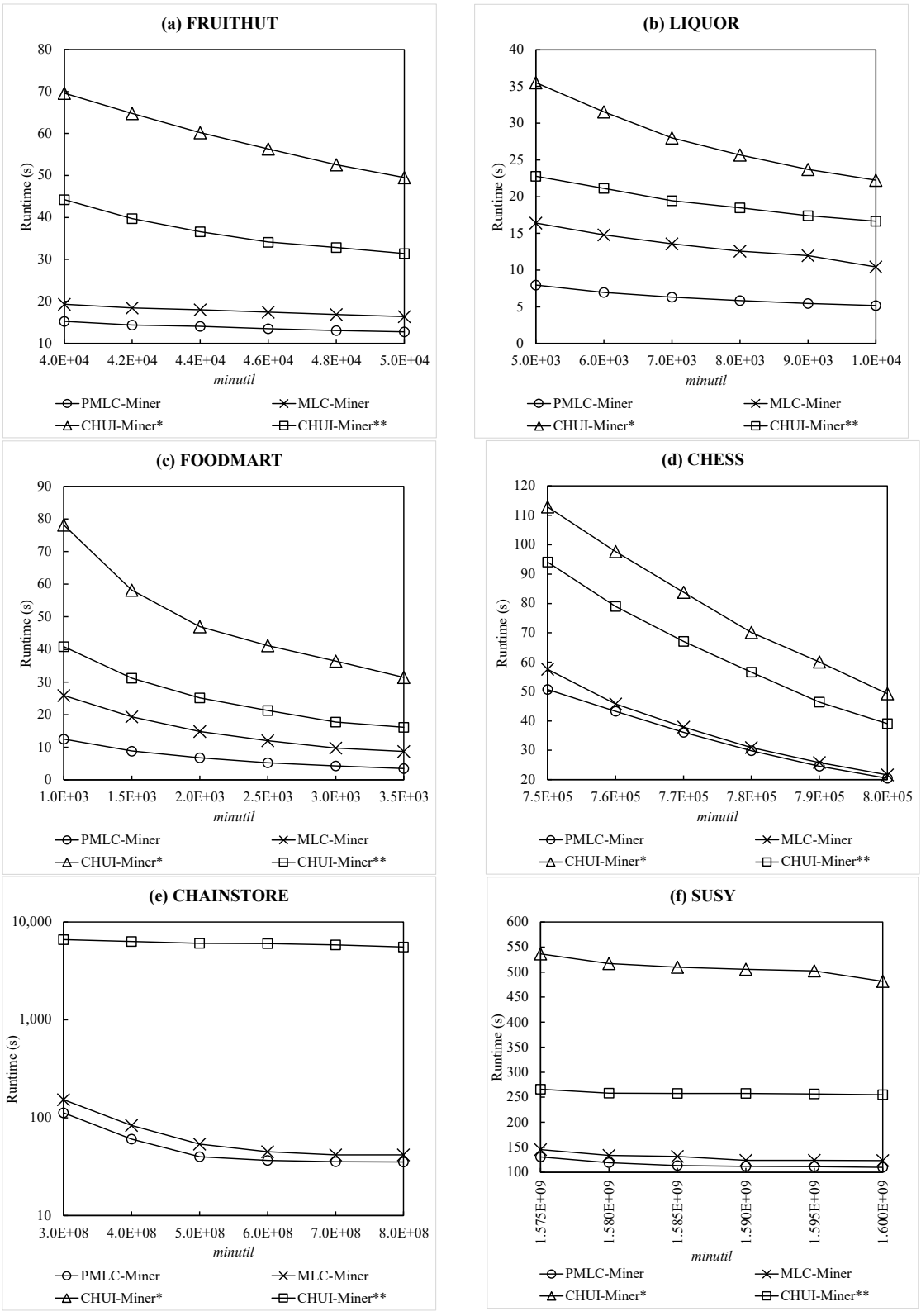
Các thống kê về thời gian thực hiện giữa PMLC-Miner, MLC-Miner, CHUI-Miner*, CHUI-Miner** được ghi nhận và thể hiện tại Hình 3.11. Ta có thể nhận thấy, trong tất cả các thực nghiệm, thời gian khai thác của PMLC-Miner lcó sự cải thiện đáng kể so với CHUI-Miner* và CHUI-Miner**. Tuy nhiên, nếu so với MLC-Miner thì mức độ cải thiện là không lớn. Lưu ý rằng, tương tự như trong phần thực nghiệm của giải thuật MLC-Miner, giải thuật CHUI-Miner* không hoàn thành trong khoảng thời gian hai giờ (7,200 giây) đối với CSDL Chainstore, vì vậy CHUI-Miner* không xuất hiện trong Hình 3.11e. CHUI-Miner** khi áp dụng mô hình song song hóa đã đề xuất cho thấy có sự cải thiện về thời gian thực hiện. Tuy nhiên, mức độ cải thiện này là không lớn. Điều này là do CHUI-Miner** và CHUI-Miner* không áp dụng các chiến lược hiệu quả để giảm độ phức tạp của giải thuật như MLC-Miner và PMLC-Miner.

Cụ thể, mức độ cải thiện về thời gian của PMLC-Miner so với CHUI-Miner* lên đến hơn 5 lần trên CSDL Foodmart (Hình 3.11c), trên 4 lần đối với các CSDL Fruithut (Hình 3.11a), Liquor (Hình 3.11b) và SUSY (Hình 3.11f). Mức độ cải thiện về thời gian thấp nhất là trên CSDL Chess, chỉ với 2.2 lần. Điều này có thể giải thích được là do CSDL này có kích thước nhỏ, và chỉ có 3 mức taxonomy.

Mức độ cải thiện về thời gian khai thác được thể hiện rõ rệt nhất giữa PMLC-Miner và CHUI-Miner** được thể hiện tại CSDL Chainstore (Hình 3.11e). Tỷ lệ cải thiện về thời gian khai thác lên đến hơn 110 lần. Chainstore là CSDL có mật độ cực thấp, hơn một triệu giao dịch và số lượng hạng mục chuyên biệt lẫn tổng quát lớn. Ngoài ra, Chainstore cũng có cấu trúc taxonomy sâu nhất trong tất cả các CSDL thực nghiệm với 9 cấp. Điều này thể hiện qua mức độ cải thiện lớn về thời gian của PMLC-Miner so với CHUI-Miner**. CHUI-Miner* không hoàn thành thực nghiệm trong thời gian 7,200 giây, vì vậy giải thuật này không xuất hiện ở Hình 3.11e.

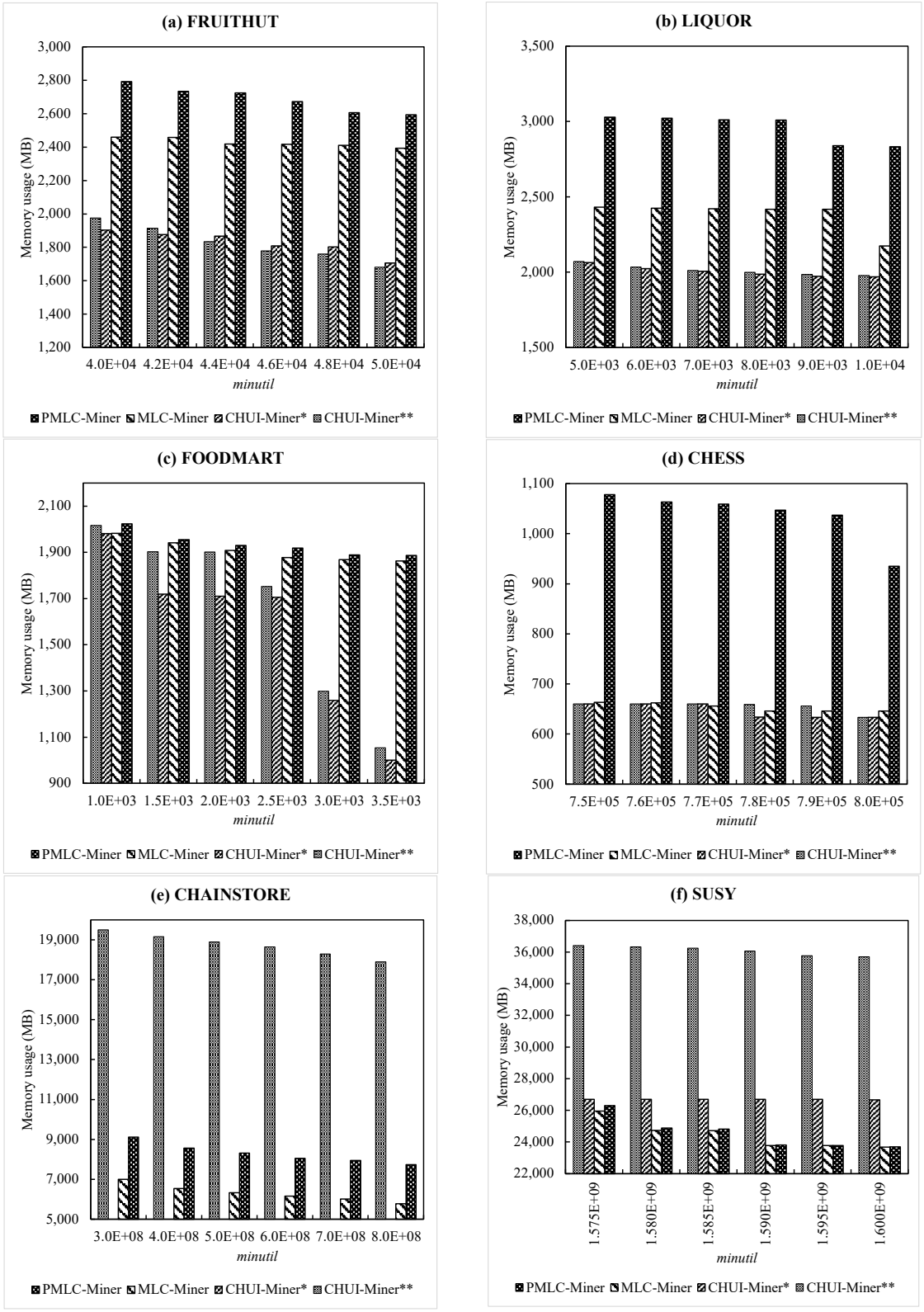
Khi so sánh về thời gian thực hiện giữa PMLC-Miner và MLC-Miner, ta cũng nhận thấy có sự cải thiện về thời gian. Tuy nhiên do MLC-Miner đã giải quyết bài toán khá hiệu quả, vì vậy việc song song của PMLC-Miner hoá không mang lại nhiều sự cải

thiện về mặt thời gian. PMLC-Miner có thời gian khai thác nhanh hơn MLC-Miner lên đến hai lần trong các CSDL thực nghiệm.



Hình 3.11. So sánh thời gian thực hiện của PMLC-Miner.

3.6.3.2 Đánh giá về mức độ sử dụng bộ nhớ tối đa



Hình 3.12. So sánh mức độ sử dụng bộ nhớ của PMLC-Miner.

Đồng thời với quá trình thực nghiệm về thời gian thực thi của các giải thuật, lượng bộ nhớ cực đại được cấp phát cho các giải thuật cũng được ghi nhận lại và thể hiện tại Hình 3.12.

Về mặt tổng thể, lượng bộ nhớ tối đa cần thiết của PMLC-Miner đều cao hơn hẳn so với MLC-Miner. Điều này là do quá trình song song hoá cần có thêm không gian lưu trữ để các tiểu trình (thread) có thể hoạt động đồng thời. Càng nhiều tiểu trình được cấp phát, lượng bộ nhớ được cấp phát cũng sẽ gia tăng theo.

Từ Hình 3.12, PMLC-Miner có mức độ sử dụng bộ nhớ thấp hơn CHUI-Miner* trên một số CSDL và lại cao hơn CHUI-Miner* trên một số CSDL khác. Cụ thể, trên CSDL Foodmart (Hình 3.12c) và CSDL Chess (Hình 3.12d), PMLC-Miner có lượng bộ nhớ cấp phát cao hơn MLC-Miner và cả CHUI-Miner*. Đối với CSDL Foodmart, mức độ chênh lệch trung bình giữa PMLC-Miner và CHUI-Miner* là xấp xỉ 190MB và với Chess là 470MB. Một con số không lớn so với tổng dung lượng bộ nhớ thông thường mà các máy tính phổ thông đang được trang bị.

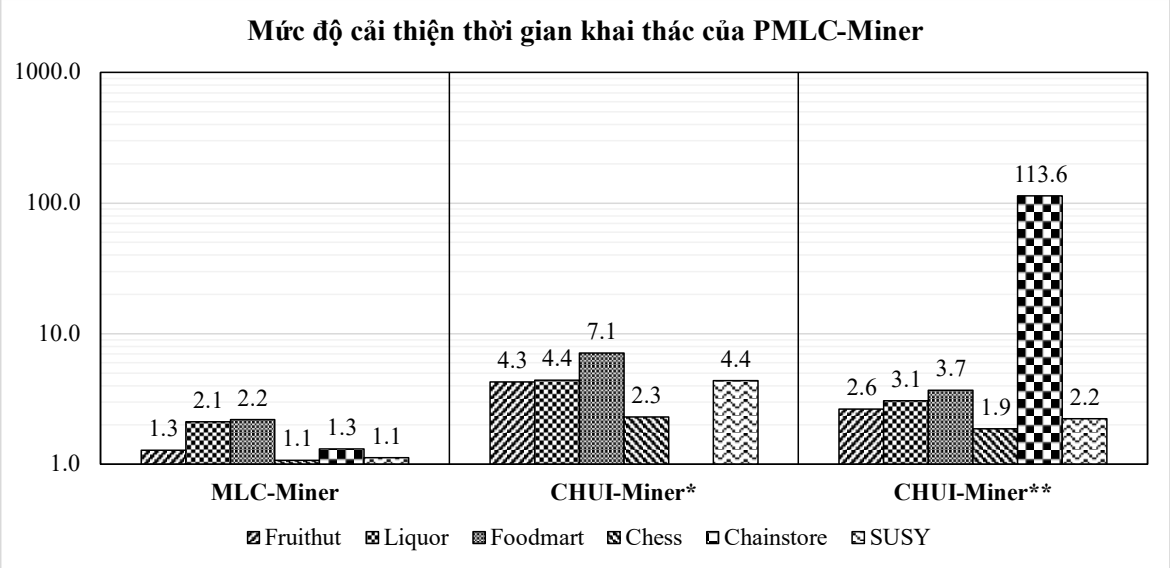
Đối với CSDL Chainstore, tương tự như các thực nghiệm ở giải thuật MLC-Miner, giải thuật CHUI-Miner* cũng không thể hoàn thành trong thời gian cho phép. Vì vậy CHUI-Miner* xem như không đáp ứng được trên các CSDL thuộc dạng này. Bên cạnh đó, PMLC-Miner cho mức độ cải thiện về thời gian không lớn so với MLC-Miner, chỉ 1.3 lần nhưng mức độ gia tăng về bộ nhớ so với MLC-Miner là xấp xỉ 2.0GB. Điều này thể hiện việc song song hoá của PMLC-Miner chưa thực sự tốt.

Hơn nữa, đối với CSDL SUSY, CHUI-Miner* cho thấy mức độ sử dụng bộ nhớ là cao nhất. Lý do chính là do cấu trúc dữ liệu utility-list truyền thống mà giải thuật này sử dụng không hiệu quả khi hoạt động trên các CSDL có số lượng giao dịch lớn như SUSY hoặc Chainstore. Ngoài ra, khi không gian bài toán được mở rộng với taxonomy, lượng bộ nhớ cần thiết để lưu trữ utility-list trên tất cả các mức của taxonomy cũng gia tăng đáng kể.

Trên hai CSDL lớn là Chainstore và SUSY, CHUI-Miner** cũng cho thấy mức độ tiêu thụ bộ nhớ của giải thuật là cực lớn vì cần phải lưu trữ các utility-list dài kết hợp với không gian để quản lý các luồng xử lý song song trong quá trình khai thác.

3.6.3.3 Mức độ cải thiện về hiệu năng của giải thuật PMLC-Miner

Để có cái nhìn tổng quát hơn về mức độ cải thiện hiệu năng của PMLC-Miner, Hình 3.13 tổng hợp tỷ lệ cải thiện thời gian khai thác trung bình giữa giải thuật này và ba giải thuật baseline là MLC-Miner, CHUI-Miner* và CHUI-Miner** trên tất cả các CSDL thực nghiệm.



Hình 3.13. Hiệu quả của PMLC-Miner so các giải thuật baseline.

Mức độ cải thiện về thời gian khai thác giữa PMLC-Miner và CHUI-Miner* là từ hơn hai lần đến cao nhất là gần bảy lần. Đối với PMLC-Miner và MLC-Miner, tỷ lệ này dao động trong khoảng từ 1.1 lần cho đến hơn hai lần. Đối với giải thuật song song hóa CHUI-Miner**, giải thuật PMLC-Miner cho thấy mức độ cải thiện về thời gian khai thác dao động từ gần hai lần lên đến trên 110 lần.

Qua đó cho thấy được mức độ hiệu quả của giải thuật MLC-Miner trong việc khai thác các tập mục hữu ích cao đa mức – đóng. Vì vậy phiên bản song song hoá của MLC-Miner là PMLC-Miner không cải thiện hiệu năng nhiều hơn được so với phiên bản tuần tự.

3.7 Kết chương

Chương này tập trung nghiên cứu bài toán khai thác song song các tập mục hữu ích cao đa mức, đa mức - đóng trên CSDL có phân cấp các hạng mục và đề xuất hàng loạt các phương pháp để giải quyết bài toán này. Các giải thuật được đề xuất lần lượt có tên gọi là MCML-Miner, MLC-Miner và PMLC-Miner. Trong đó, MCML-Miner là phiên bản mở rộng, song song hoá của MLHUI-Miner để phục vụ cho bài toán khai thác song song các tập mục hữu ích cao đa mức; MLC-Miner là phiên bản mở rộng và cải tiến của giải thuật CHUI-Miner để khai thác hiệu quả các tập mục hữu ích cao đa mức đóng; PMLC-Miner là phiên bản song song hoá của MLC-Miner, phục vụ cho bài toán khai thác các tập mục hữu ích cao đa mức – đóng từ CSDL phân cấp. Dưới đây là các ưu và nhược điểm của các giải thuật được trình bày.

Giải thuật MCML-Miner

- Ưu điểm
  - Là giải thuật một pha, phục vụ cho việc khai thác song song các tập mục hữu ích cao đa mức.
  - Hoạt động dựa trên taxonomy của CSDL giao dịch.
  - Song song hoá sử dụng cơ chế đa nhân, tận dụng các nhân của vi xử lý để phục vụ cho việc quét song song không gian tìm kiếm của bài toán.
  - Thời gian cải thiện đáng kể so với giải thuật tuần tự MLHUI-Miner.
- Nhược điểm
  - Mức độ sử dụng bộ nhớ là cao hơn so với MLHUI-Miner. Taxonomy càng lớn, mức độ sử dụng bộ nhớ càng cao.
  - Không tận dụng được tất cả các nhân nhàn rỗi của vi xử lý nếu như chiều cao của taxonomy nhỏ hơn số lượng nhân của vi xử lý.
  - Quá trình xử lý song song chỉ diễn ra trong pha quét không gian tìm kiếm.

Giải thuật MLC-Miner

- Ưu điểm
  - Có khả năng khai thác tất cả các tập mục hữu ích cao đa mức - đóng trong CSDL giao dịch có sự phân cấp các hạng mục.

- Có thể sử dụng hoặc không sử dụng taxonomy để biểu diễn cấu trúc phân cấp các hạng mục.
- Ứng dụng kỹ thuật tỉa ứng viên EUCP để loại bỏ sớm các tập mục không tiềm năng khỏi quá trình xét.
- Áp dụng kỹ thuật tối ưu hoá trong việc kiểm tra tính đóng các tập mục, chiến lược RUCK, để cải thiện hiệu năng giải thuật.
- Chỉ cần thực hiện hai lần quét CSDL trong toàn bộ quá trình khai thác.
- Mức độ sử dụng bộ nhớ thấp hơn đáng kể so với CHUI-Miner*.

- Nhược điểm
  - Thời gian thực hiện của giải thuật vẫn còn cao trên các CSDL có số lượng hạng mục lớn.

Giải thuật PMLC-Miner

- Ưu điểm
  - Giải thuật song song một pha khai thác các tập mục hữu ích cao đa mức – đóng từ CSDL phân cấp.
  - Song song hoá sử dụng cơ chế đa nhân. Tận dụng các nhân của vi xử lý để phục vụ cho việc quét song song các không gian tìm kiếm của bài toán.
  - Cải thiện thời gian thực hiện so với giải thuật tuần tự ban đầu là CHUI-Miner* và MLC-Miner. Đặc biệt có sự cải thiện đáng kể so với CHUI-Miner*.
  - Mức độ cải thiện thời gian khai thác đối với giải thuật song song hóa cùng lớp là CHUI-Miner** lên đến trên 110 lần.

- Nhược điểm
  - Do áp dụng mô hình song song hoá tương tự như MCML-Miner, giải thuật PMLC-Miner cũng gặp phải tất cả các nhược điểm của MCML-Miner.
  - Hiệu năng khai thác cải thiện so với MLC-Miner cũng không cao, do mức độ hiệu quả sẵn có của MLC-Miner.

Nội dung nghiên cứu 2 của luận án đã hoàn thành bằng việc đề xuất một mô hình song song hoá như đã trình bày xuyên suốt trong toàn bộ CHƯƠNG 3. Mô hình này

được áp dụng vào hai giải thuật là MCML-Miner và PMLC-Miner và đều cho thấy có sự cải thiện về hiệu quả khai thác các tập mục hữu ích cao đa mức.

Tuy nhiên, cũng từ các nhận định nói trên, ta có thể thấy được quá trình xử lý song song của mô hình đề xuất chỉ diễn ra duy nhất trong pha quét không gian tìm kiếm của giải thuật. Khi áp dụng lên các CSDL có kích thước lớn, ngoài quá trình quét không gian tìm kiếm để xác định các tập mục hữu ích cao, vốn có thời gian khai thác lớn, các giai đoạn hỗ trợ cho quá trình khai thác như quét CSDL để xác định TWU, tỉa sớm các hạng mục không tiềm năng xây dựng cấu trúc EUCS, v.v… cũng mất rất nhiều thời gian nhưng lại chưa được xem xét áp dụng mô hình song song hoá.

Bên cạnh đó, mô hình song song hoá này cũng chưa tận dụng triệt để các nhân xử lý nhàn rỗi của các vi xử lý có nhiều nhân và còn phụ thuộc chính vào chiều sâu của taxonomy. Nếu áp dụng mô hình song song hóa này trên các CSDL thông thường, các CSDL không chứa taxonomy, thì quá trình song song hoá sẽ bị triệt tiêu.

Hướng phát triển tiếp theo của luận án sẽ tập trung vào các công việc chính như sau:

- Mở rộng quá trình song song hóa ra trên các giai đoạn khác nhau của quá giải thuật khai thác thay vì chỉ tập trung duy nhất vào quá trình quét không gian tìm kiếm của bài toán.

- Cải thiện sâu hơn nữa mô hình song song hoá này. Từ đó, tận dụng tốt hơn, triệt để hơn sức mạnh tính toán song song của các vi xử lý đa nhân phục vụ cho bài toán khai thác các tập mục hữu ích cao hiệu quả hơn.

Đây cũng chính là mục tiêu của **Nội dung nghiên cứu 3** và sẽ được khảo sát, triển khai trong CHƯƠNG 4.

CHƯƠNG 4 CẢI THIỆN HIỆU QUẢ MÔ HÌNH KHAI THÁC SONG SONG CÁC TẬP MỤC HỮU ÍCH CAO ĐA MỨC

4.1 Giới thiệu

Với các nội dung thuộc phạm vi **Nội dung nghiên cứu 2** đã trình bày trong CHƯƠNG 3, bài toán khai tác tập mục hữu ích cao đa mức từ CSDL có sự phân cấp hạng mục được đề xuất vào năm 2017. Giải thuật đầu tiên phục vụ cho hướng nghiên cứu này là MLHUI-Miner. Một số giải thuật nhằm cải thiện hiệu quả hiệu năng khai thác của bài toán này cũng đã được đề xuất. Tuy nhiên, tất cả các giải thuật này đều được phát triển ở dạng các giải thuật tuần tự. Cho dù được triển khai trên các hệ thống tính toán có nhiều vi xử lý, các giải thuật này vẫn chỉ sử dụng duy nhất một vi xử lý để thực hiện công việc mà chúng đã được thiết kế. Do đó, các giải thuật này không có khả năng tận dụng được năng lực tính toán của các vi xử lý đa nhân.

Bài toán khai thác tập mục hữu ích cao đa mức có đặc điểm là không gian tìm kiếm được mở rộng hơn nhiều lần so với bài toán khai thác tập mục hữu ích cao truyền thống khi được kết hợp với thông tin taxonomy. Tuy nhiên, để khảo sát không gian tìm kiếm rộng lớn của bài toán, các giải thuật khai thác vẫn chỉ hoạt động ở dạng tuần tự và không hiệu quả để triển khai trong thực tế.

Mục 3.3 thuộc **Nội dung nghiên cứu 2** tại CHƯƠNG 3 đã đề xuất một mô hình song song hoá dựa trên taxonomy để tiến hành khai thác song song. Mô hình này cũng được áp dụng vào hai giải thuật là MCML-Miner và PMLC-Miner để khai thác lần lượt các tập mục hữu ích cao đa mức và đa mức – đóng. Tuy nhiên, mô hình này chỉ tập trung triển khai song song hoá quá trình khai thác mà thôi. Hơn nữa, mô hình cũng phụ thuộc hoàn toàn vào chiều sâu của cấu trúc taxonomy.

Nội dung nghiên cứu này tập trung vào việc nghiên cứu, mở rộng và cải thiện sâu hơn nữa mô hình song song hoá đã trình bày để có thể có thể tận dụng triệt để các vi xử lý đa nhân phổ biến hiện nay.

4.2 Cải thiện hiệu quả của mô hình khai thác song song tập mục hữu ích cao đa mức

4.2.1 Giải thuật MCML+

Như đã trình bày, nhược điểm lớn nhất của giải thuật MCML-Miner là chỉ khai thác không gian tìm kiếm của từng cấp trong taxonomy một cách độc lập và song song trên từng nhân của CPU. Trong các trường hợp thông thường sử dụng các CPU với 2 đến 4 nhân, MCML-Miner cho hiệu quả khai thác tốt. Nếu CPU có số nhân xử lý nhiều hơn chiều sâu tối đa của taxonomy, khi đó chiến lược này trở nên kém hiệu quả vì không sử dụng tất cả các nhân CPU còn rãnh rỗi. Tại thời điểm thực hiện luận án, số lượng nhân xử lý trong một CPU đã lên đến 12 nhân trên một đế, và giá thành cho các CPU này cũng thấp hơn so với các thế hệ trước đó. Trong trường hợp xấu nhất, nếu taxonomy τ là rỗng, kỹ thuật song song hoá bị triệt tiêu hoàn toàn và giải thuật quay trở về bài toán khai thác HUI truyền thống, dẫn đến việc lãng phí tài nguyên tính toán. Vì vậy, mục tiêu của phần nội dung này là nghiên cứu và đề ra các tiếp cận mới hơn nhằm giải quyết vấn đề mà MCML-Miner đang gặp phải, từ đó tận dụng tốt hơn năng lực xử lý của các CPU hiện đại ngày nay. Ngoài ra, MCML-Miner cũng chỉ được song song hoá tại giai đoạn khai thác, trong thực tế, ta có thể triển khai việc song song hoá ở nhiều giai đoạn khác nhau của giải thuật, không chỉ ở quá trình khai thác chính, để tăng hơn nữa hiệu quả khai thác tập mục hữu ích cao đa mức.

Vì vậy, mục đích chính của giải thuật MCML+ là triển khai quá trình song song hoá trên các giai đoạn khác nhau của giải thuật MCML-Miner ban đầu, ngoài giai đoạn khai thác song song hoá đã có.

Đặc điểm của giải thuật MCML+ gồm:

- Giải thuật một pha.
- Là phiên bản cải tiến của giải thuật MCML-Miner.
- Khai thác dựa trên taxonomy [40].
- Song song hoá dựa trên mô hình xử lý đa nhân.

- Áp dụng chiến lược song song hoá đa nhân ở các giai đoạn khác nhau cải thiện hiệu năng của quá trình khai thác.

Giải thuật MCML-Miner ngoài quá trình khai thác chính có thể song song hoá, một số giai đoạn khác của giải thuật cũng mất rất nhiều thời gian để thực hiện, đặc biệt trên các CSDL lớn. Các giai đoạn này gồm:

- Tỉa hoàn toàn các hạng mục không tiềm năng khỏi CSDL.
- Xây dựng cấu trúc EUCS trước khi thực hiện quá trình khai thác, nhằm áp dụng chiến lược EUCP.

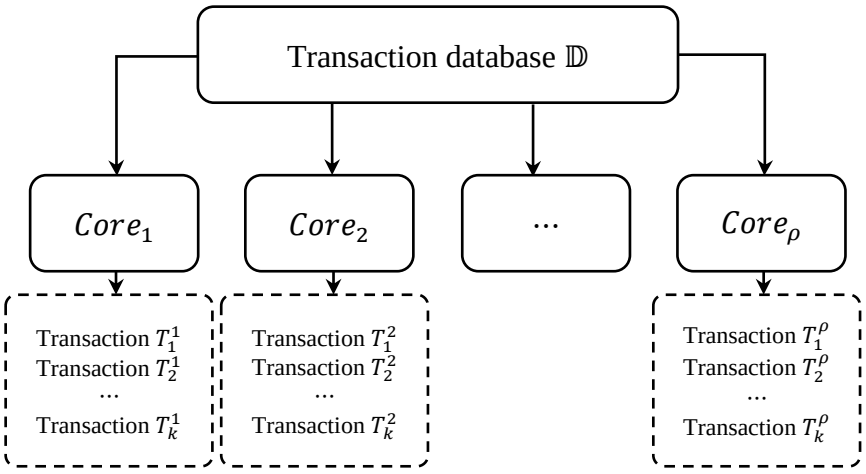
Với các nhận định trên, các kỹ thuật song song hoá sau đây được đề xuất để cải thiện thời gian thực hiện của toàn bộ giải thuật.

4.2.1.1 Kỹ thuật tỉa song song các ứng viên không tiềm năng

Thực hiện hiện việc quét CSDL để lược bỏ các hạng mục không tiềm năng và sắp xếp lại thứ tự các hạng mục trong từng giao dịch là một trong các thao tác chính của các giải thuật khai thác tập mục hữu ích cao gần đây. Trong bước này, tất cả các hạng mục có $TWU < minutil$ đều được xem là không tiềm năng trở thành HUI hoặc tham gia vào các tập mục HUI. Vì vậy, chúng sẽ được loại bỏ khỏi không gian tìm kiếm. Tất cả các giải thuật khai thác HUI truyền thống như HUI-Miner [67], FHM [16], EFIM [58], iMEFIM [17], ML-HUI-Miner [40], MLHMiner [73] hoặc ngay cả phiên bản song song của CLH-Miner là pCLH-Miner [49] cũng đều triển khai bước này một cách tuần tự. Khi tiến hành quét CSDL để thực hiện công việc này, hầu hết các giải thuật HUIM đều thực hiện các công việc sau trên từng giao dịch đơn lẻ:

- Quét qua giao dịch để loại bỏ các hạng mục không tiềm năng [67], [58].
- Sắp xếp các hạng mục trong giao dịch dựa trên thứ tự tăng của TWU [67], [58].
- Đánh giá lại giá trị TU của mỗi giao dịch sau khi đã loại bỏ các hạng mục không tiềm năng.

Theo các tiếp cận truyền thống, quá trình này chỉ diễn ra theo cách tuần tự và tại một thời điểm, chỉ một giao dịch được xử lý. Điều này thể hiện rõ việc lãng phí tài nguyên tính toán và tác động đáng kể lên thời gian khai thác, đặc biệt trên các CSDL có kích thước lớn có số lượng giao dịch lên đến hàng triệu. Tuy nhiên, ta có thể triển khai việc xử lý song song để áp dụng vào bước này nhằm giảm đi thời gian cần thiết cho việc tỉa các ứng viên không tiềm năng (Hình 4.1). Giả sử CSDL $\mathbb{D}$ có tổng cộng m giao dịch, CPU có tổng cộng ρ nhân xử lý, ký hiệu $Core_i$ $(1 \leq i \leq \rho)$ đại diện cho nhân thứ i trong ρ nhân. Từ đó, ta có thể ấn định $\rho - 1$ nhân xử lý đầu tiên để thao tác trên $k = \left\lceil \frac{m}{\rho} \right\rceil$ giao dịch. Các giao dịch còn lại (nếu có), là $m \; mod \; \rho$, sẽ được gán cho nhân xử lý cuối cùng, nhân thứ ρ. Trong trường hợp $m \; mod \; \rho = 0$, khi đó m giao dịch sẽ được phân phối đồng đều cho tất cả ρ nhân. Đây là tiếp cận đầu tiên trong việc cải thiện mô hình song song hoá của giải thuật MCML-Miner.



Hình 4.1. Mô hình song song hoá quá trình tỉa ứng viên [CT2]

Sau khi được gán cho từng nhân CPU để thao tác, giá trị TU của mỗi giao dịch cần phải được tính lại vì chúng không còn chứa các hạng mục không tiềm năng nữa. Hơn nữa, giá trị TU của các giao dịch là khác nhau ứng với từng cấp trong taxonomy.

4.2.1.2 Song song hoá việc tạo EUCS

Kỹ thuật xử lý song song cũng có thể được áp dụng để xây dựng cấu trúc EUCS. Để thực hiện được điều này, EUCS cần phải được đặt tại vị trí toàn cục của giải thuật để tất cả các nhân xử lý có thể dùng chung cấu trúc này một cách đồng bộ. Mỗi khi một

giao dịch T_c được quét, ứng với một cặp tập mục $X, Y \subseteq T_c$, giá trị TU của T_c sẽ được cập nhật vào EUCS. Đây là tiếp cận thứ 2 trong việc cải tiến mô hình song song hoá của MCML-Miner.

4.2.1.3 Tổng hợp

Đây là giải thuật song song hoá thứ nhất được đề xuất để cải thiện hiệu quả của giải thuật MCML-Miner. Tiếp cận này được đặt tên là MCML+ và có khả năng tận dụng tất cả các nhân CPU nhàn rỗi vào việc *tỉa song song các ứng viên không tiềm năng khỏi CSDL* $\mathbb{D}$ và *xây dựng song song cấu trúc EUCS*.

Giải thuật MCML+ bao gồm hai giai đoạn song song hoá chính như sau:

- Giai đoạn song song hoá thứ nhất áp dụng để tỉa các ứng viên không tiềm năng khỏi $\mathbb{D}$ và xây dựng song song cấu trúc EUCS.
- Giai đoạn song song hoá thứ hai chính là phần song song của giải thuật MCML-Miner, áp dụng để khai thác MLHUI đồng thời theo từng cấp taxonomy.

4.2.2 Giải thuật MCML++

Tiếp cận được đề xuất với giải thuật MCML+ đạt được mục đích cải thiện thời gian khai thác, đặc biệt trên các CSDL lớn. Tuy nhiên khi bắt đầu tiến hành quá trình khai thác chính thì MCML+ lại quay trở lại giải thuật MCML-Miner ban đầu, do đó vẫn chưa đạt được hiệu quả như mong muốn. Để cải thiện hơn nữa hiệu năng khai thác, nhược điểm này cần phải được đánh giá và xử lý triệt để: tất cả các nhân CPU cần phải được đưa vào sử dụng cho quá trình quét không gian bài toán, thay vì chỉ tận dụng một vài nhân để quét từng cấp của taxonomy.

Một phương pháp để giải quyết vấn đề của MCML-Miner là phân phối không gian tìm kiếm của từng hạng mục đơn cho các nhân CPU. Tiếp cận này tương tự như các giải thuật trước đó đã áp dụng như pEFIM [47], MCH-Miner [48], v.v… Chiến lược điều phối được sử dụng trong các giải thuật này khá đơn giản: "*Đến trước, Phục vụ trước*". Ưu điểm của tiếp cận này là giải phóng CPU khỏi bị nghẽn cổ chai khi chỉ tận dụng một số lượng ít các nhân xử lý và để trống các nhân còn lại như đối với

MCML-Miner. Tuy nhiên, việc ấn định các nhánh trong không gian tìm kiếm một cách ngẫu nhiên cho các nhân CPU mà không xem xét đến đặc điểm của chúng cũng không đạt được hiệu quả như mong muốn. Các tác vụ để quét không gian tìm kiếm lớn thường có thời gian thực hiện dài, đôi khi lại được đẩy lên thực thi trước các tác vụ có thời gian thực hiện ngắn. Nếu hàng loạt các tác vụ có thời gian thực thi lâu được đẩy lên trước các tác vụ có thời gian thực thi nhanh, tổng thời gian chờ trong hàng đợi sẽ tăng. Vì thế, luận án đề xuất một heuristic như sau: các không gian tìm kiếm có kích thước nhỏ (có số lượng nhỏ mở rộng cho các tập mục) nên được đẩy lên thực hiện trước, rồi đến các không gian có kích thước lớn hơn. Chiến lược điều phối này còn được biết đến với tên gọi *Ưu tiên Tác vụ Ngắn nhất trước* (Shortest Jobs First - SJF). Các không gian tìm kiếm nhỏ sẽ mất ít thời gian khám phá hơn là các không gian tìm kiếm lớn, và ngược lại.

Định nghĩa 4.1. (Trình tự xử lý)

Cho $\mathcal{A}$ và $\mathcal{B}$ là hai tập mục cần được xét mở rộng trong quá trình khai thác. Không gian tìm kiếm của $\mathcal{A}$ được nói là nhỏ hơn không gian tìm kiếm của $\mathcal{B}$ nếu $|E(\mathcal{A})| < |E(\mathcal{B})|$, trong đó $E(\mathcal{A})$ và $E(\mathcal{B})$ là tập các hạng mục có thể dùng để mở rộng lần lượt tập mục $\mathcal{A}$ và $\mathcal{B}$.

Bằng việc áp dụng kỹ thuật dưới dạng heuristic này, các hạng mục có $TWU \geq minutil$ sẽ được sắp xếp lại trước khi gán cho từng nhân CPU để tiến hành xử lý. Chiến lược này bảo đảm rằng các không gian có kích thước tiềm năng nhỏ (chứa ít các hạng mục có thể dùng để mở rộng) sẽ được ưu tiên xử lý trước các không gian lớn hơn, thay vì các không gian tìm kiếm được lựa chọn xử lý một cách ngẫu nhiên theo quy tắc "*Đến trước, phục vụ trước*". Ta có thể hiện thực thứ tự xử lý này bằng cách sử dụng một cấu trúc hàng đợi ưu tiên.

Giải thuật MCML++ được đề xuất tích hợp tất cả các kỹ thuật song song hoá đã được trình bày. Giai đoạn khai thác song song phụ thuộc vào cấp taxonomy của cả giải thuật MCML-Miner và MCML+ được thay thế hoàn toàn với việc phân phối và

quétsong song các không gian tìm kiếm cho các nhân CPU. Các đặc điểm của giải thuật MCML++ có thể được tóm tắt như sau:

- Là giải thuật một pha để khai thác tất cả các tập mục hữu ích cao đa mức có sự phân cấp hạng mục dưới dạng taxonomy.
- Tích hợp mô hình xử lý song song đa nhân để cải thiện hiệu năng khai thác:
  - Tỉa song song các ứng viên không tiềm năng sử dụng tất cả các nhân CPU.
  - Song song hoá quá trình tạo EUCS.
  - Loại bỏ hạn chế về quá trình song song hoá dựa trên cấp của MCML-Miner bằng cách "phẳng hoá" tất cả các hạng mục tiềm năng. Mỗi hạng mục này sẽ lần lượt được tất cả các nhân CPU xử lý một cách độc lập và đồng thời. Khai thác tối đa năng lực xử lý của CPU mà không lệ thuộc vào chiều sâu của taxonomy.
  - Áp dụng chiến lược điều phối SJF vào quá trình quét song song không gian tìm kiếm của bài toán.

Input: CSDL $\mathbb{D}$, taxonomy τ, ngưỡng $minutil$

Output: tập các tập mục hữu ích cao đóng, đa mức $\mathcal{R}$

1 $\mathcal{R} \leftarrow \emptyset$; $EUCS \leftarrow \emptyset$;
2 Quét $\mathbb{D}$ sử dụng τ để xây dựng tập $\mathbb{I}$ và $\mathbb{G}$ và xác định TWU của các hạng mục $i \in \mathbb{I}$.
3 Tính TWU cho các hạng mục tổng quát $g \in \mathbb{G}$.
4 Song song hóa, tỉa các hạng mục: $\mathbb{I}^+ \leftarrow \{i \in \mathbb{I} \mid TWU(i) \geq minutil\}$.
5 Song song hóa, tỉa các hạng mục: $\mathbb{G}^+ \leftarrow \{g \in \mathbb{G} \mid TWU(g) \geq minutil\}$.
6 Song song hóa, quét $\mathbb{D}$ để xây dựng EU-List, xây dựng $EUCS$ cho các hạng mục thuộc $\mathbb{I}^+ \cup \mathbb{G}^+$ sử dụng taxonomy τ.
7 Sắp thứ tự các hạng mục tiềm năng trong sử dụng chiến lược SJF.
8 Đẩy các hạng mục $\mathbb{I}^+ \cup \mathbb{G}^+$ vào hàng đợi ưu tiên dựa trên thứ tự SJF.
9 **FOR EACH** $i \in \mathbb{I}^+ \cup \mathbb{G}^+$ **DO PARALLEL**
10 $\mathbb{IG}_i \leftarrow$ các hạng mục tiềm năng có thể mở rộng i.
11 $\mathcal{R} \leftarrow \mathcal{R} \cup$ **Explore**$(\emptyset, \mathbb{IG}_i, \emptyset, minutil, EUCS)$;
12 **END**
13 **RETURN** $\mathcal{R}$.

Giải thuật 4.1. Giải thuật MCML++

Trong đó hai tiếp cận đầu tiên đã được áp dụng trong giải thuật MCML+. Mã giả của giải thuật MCML++ được thể hiện tại Giải thuật 4.1. Trong đó, quá trình song song hóa được áp dụng trên nhiều giai đoạn khác nhau của giải thuật (các dòng 4, 5, 7, 8). Quá trình khai thác song song hóa phụ thuộc vào taxonomy của MCML-Miner được thay thế hoàn toàn từ dòng 9 đến dòng 10. Thủ tục **Explore** để quét không gian tìm kiếm của bài toán là không đổi so với giải thuật MCML-Miner. Tuy nhiên, với MCML++, giải thuật này được áp dụng cho từng hạng mục tiềm năng và được thực hiện một cách song song.

Bảng 4.1. Mức độ song song hoá giữa các giải thuật

Giải thuật	**Mức độ song song hoá**		
	Tỉa TWU	***Phụ thuộc cấp taxonomy***	***Độc lập cấp taxonomy/Điều phối***
MLHUI-Miner	Không	Không	Không
MCML-Miner	Không	**Có**	Không
PMLC-Miner	Không	**Có**	Không
MCML+	**Có**	**Có**	Không
MCML++	**Có**	Không	**Có**

Với việc đề xuất giải thuật MCML++, quá trình khai thác song song phụ thuộc vào cấp taxonomy diễn ra tại giải thuật MCML-Miner và MCML+ đã được loại bỏ hoàn toàn. MCML++ tận dụng tất cả các nhân CPU hiện có vào quá trình khai thác để cải thiện hiệu năng tổng thể. Bảng 4.1 tổng kết mức độ song song hoá giữa hai giải thuật đã đề xuất với giải thuật MCML-Miner ban đầu. Có thể thấy rằng MCML-Miner chỉ có một giai đoạn song song hoá phụ thuộc cấp taxonomy. MCML+ bổ sung thêm một giai đoạn song song hoá tại bước tỉa hạng mục không tiềm năng và xây dựng EUCS. Tuy nhiên, giải thuật MCML+ vẫn tỏ ra chưa hiệu quả khi vẫn duy trì cơ chế song

song hoá của MCML-Miner. Việc đề xuất MCML++ để khắc phục nhược điểm này thông qua hai tiếp cận mới để khai thác triệt để năng lực xử lý của CPU và loại bỏ hoàn toàn quá trình khai thác dựa trên cấp của taxonomy.

4.3 Thực nghiệm

4.3.1 Môi trường và CSDL thực nghiệm

Việc đánh giá hiệu năng của MCML+ và MCML++ cũng được thực hiện trên cùng một hệ thống đã thực nghiệm giải thuật MCML-Miner để bảo đảm tính khách quan của các giải thuật này.

Bảng 4.2. CSDL thực nghiệm giải thuật MCML+/MCML++

Database	$\|\mathbb{D}\|$	$\|\mathbb{I}\|$	$\|\mathbb{G}\|$	$\|\tau\|$	T_{MAX}	T_{AVG}	**Mật độ**
Connect	67,577	129	39	4	43	43.0	$3.33E-01$
Chainstore	1,112,949	46,086	9,674	9	170	7.2	$1.57E-04$
Chainstore4X	4,451,796	46,086	9,674	9	170	7.2	$1.57E-04$
SUSY	5,000,000	190	63	4	19	19.0	$1.00E-01$

Đối với CSDL thực nghiệm, cả ba giải thuật này sẽ chỉ được thực nghiệm trên các CSDL được cho tại Bảng 4.2. Các CSDL thực nghiệm này được lựa chọn sử dụng trong việc đánh giá hai giải thuật MCML+ và MCML++ vì mang các đặc điểm tiêu biểu như: kích thước CSDL; số lượng hạng mục chuyên biệt; số lượng hạng mục tổng quát; mật độ phân bố các hạng mục trong từng giao dịch.

Trong đó Connect mặc dù là CSDL có kích thước nhỏ, nhưng đây là CSDL có mật độ cao với độ dài trung bình của giao dịch là lớn nhất trong toàn bộ các CSDL được thực nghiệm.

CSDL Chainstore4X là phiên bản được nhân bản bốn lần dựa trên CSDL gốc là Chainstore. Trong bốn CSDL thực nghiệm, SUSY có số lượng giao dịch là lớn nhất, lên đến năm triệu giao dịch.

Cho đến nay, chỉ có duy nhất một giải thuật MCML-Miner đã được đề xuất để phục vụ cho bài toán khai thác tập mục hữu ích cao đa mức, cùng lớp bài toán đang nghiên cứu và khảo sát. Vì vậy, giải thuật MCML-Miner được đưa vào so sánh về mặt hiệu năng song song hóa với MCML+ và MCML++. Giải thuật tuần tự MLHUI-Miner ban đầu cũng tham gia vào quá trình so sánh để ghi nhận mức độ cải thiện về hiệu năng khi áp dụng mô hình song song hóa được đề xuất.

Hai tiêu chí đánh giá về mặt hiệu năng được so sánh là thời gian thực hiện và lượng bộ nhớ sử dụng.

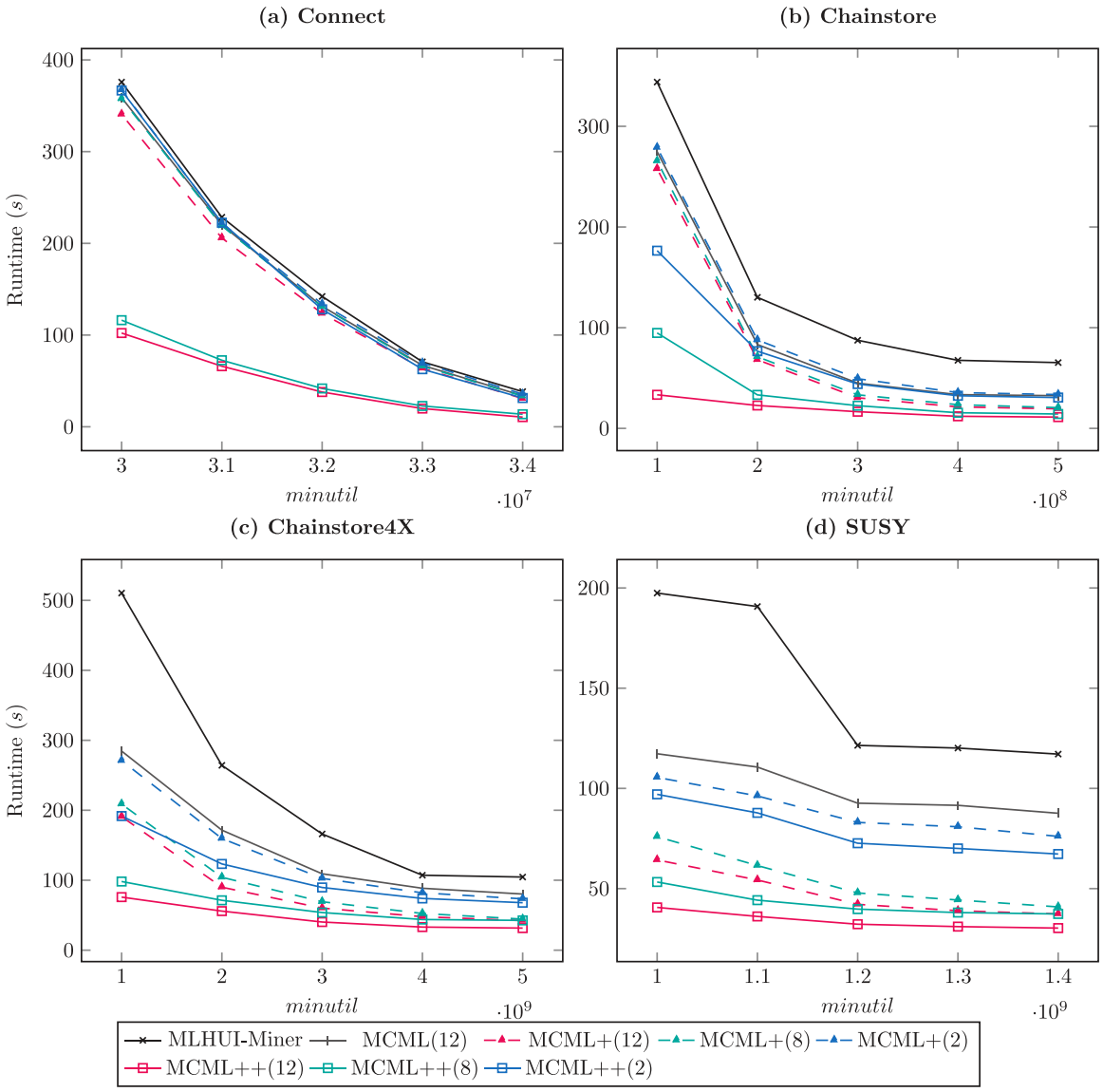
Giải thuật MCML-Miner sẽ được phân phối tất cả số nhân CPU hiện có của hệ thống, $\rho = 12$, trong tất cả các phép thử. Tuy nhiên, số nhân CPU thực tế sử dụng của MCML-Miner là phụ thuộc vào chiều sâu taxonomy ứng với mỗi CSDL thực nghiệm được sử dụng. Ví dụ, nếu sử dụng CSDL Chainstore hoặc Chainstore4X với chiều sâu taxonomy là chín, MCML-Miner sẽ chỉ sử dụng chín nhân CPU cho dù hệ thống có đến mười hai nhân.

Để đơn giản trong trình bày, giải thuật MCML-Miner sẽ được ký hiệu là MCML(12). Giải thuật MCML+ và MCML++ sẽ được thực nghiệm với $\rho = 2, 8$ và 12 nhân. Chúng lần lượt được ký hiệu là MCML+(2), MCML+(8), MCML+(12) đối với MCML+, và MCML++(2), MCML++(8), MCML++(12) đối với MCML++.

Đối với thực nghiệm về thời gian thực hiện và bộ nhớ sử dụng, $minutil$ thực nghiệm bắt đầu ở ngưỡng cao và được giảm dần trên từng CSDL để ghi nhận số liệu. Đối với thực nghiệm về khả năng mở rộng, ngưỡng $minutil$ được cố định tại giá trị thấp nhất trong phép thử về thời gian thực hiện, sau đó kích thước CSDL thực nghiệm sẽ được tăng lên theo tỷ lệ 25%, 50%, 75% và 100% của kích thước CSDL ban đầu. Hơn nữa, sự chênh lệch về hiệu năng giữa các chiến lược điều phối cũng được đánh giá. Cụ thể sẽ là ba chiến lược điều phối sau: SJF, SJF nghịch (RSJF), và "*Đến trước, Phục vụ trước*" (NS).

4.3.2 *Đánh giá về thời gian thực hiện*

Về mặt thời gian thực hiện, trong phép thực nghiệm này, mặc dù được phân phối đến 12 nhân, giải thuật MCML-Miner, ký hiệu MCML(12), vẫn chỉ tận dụng được bốn nhân đối với CSDL Connect và SUSY, chín nhân đối với Chainstore và Chainstore4X. Kết quả ghi nhận được thể hiện tại Hình 4.2. Các đường đứt nét thể hiện thời gian thực hiện của MCML+ và các đường liên tục thể hiện thời gian thực hiện của MCML++. Đường liên tục với chấm tròn thể hiện thời gian thực hiện của MCML-Miner.



Hình 4.2. Thời gian thực hiện của các giải thuật [CT2]

Qua tất cả các CSDL thực nghiệm, MCML++ luôn cho thời gian thực hiện tốt hơn hẳn so với MCML+ và MCML-Miner, ngay cả khi MCML-Miner và MCML+ được cấp phát số nhân CPU nhiều hơn cả MCML++. Đối với CSDL Connect (Hình 4.2a), tốc độ cải thiện lên đến 3.5 lần so với MCML+ và MCML-Miner.

Connect là CSDL nhỏ nhất trong các CSDL thực nghiệm, chỉ với hơn 67K giao dịch. Quá trình tỉa song song các hạng mục không tiềm năng không tỏ ra có hiệu quả trong trường hợp này. Thời gian thực hiện còn lại phụ thuộc chính vào quá trình khai thác song song.

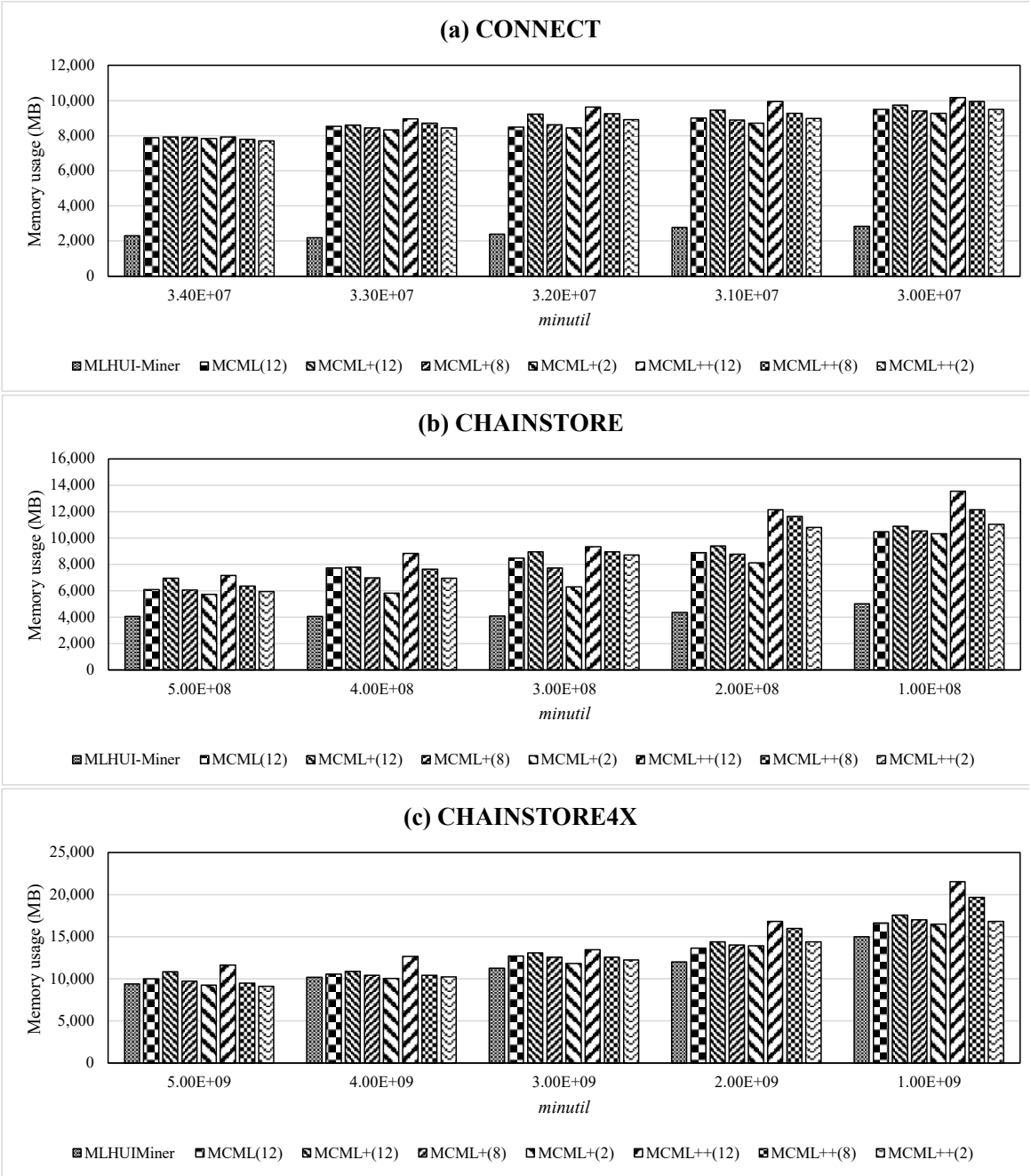
Đối với CSDL Chainstore (Hình 4.2b), tốc độ đạt được của MCML++ cao hơn hai lần so với MCML+ khi sử dụng cùng số nhân và lên đến hơn 8 lần so với MCML-Miner. Lần lượt sử dụng 8 và 12 nhân CPU, MCML++ nhanh hơn MCML+ từ 1.3 đến 1.4 lần. Khi được cấp phát chỉ hai nhân CPU, thời gian cải thiện của MCML++ so với MCML+ và MCML-Miner chỉ là 1.1 lần.

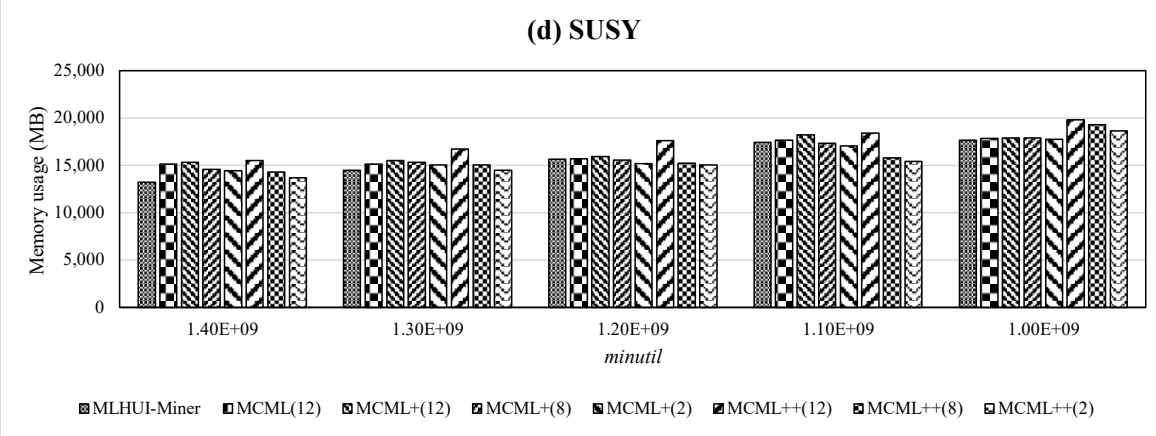
Đối với CSDL Chainstore4X (Hình 4.2c) và SUSY (Hình 4.2d), tốc độ đạt được của MCML++ (12) cao hơn gần 4 lần so với MCML(12). Ta cũng nhận thấy được MCML++(12) cũng có mức độ cải thiện tương tự so với MCML+(12). Do đây là các CSDL lớn, các tiếp cận được MCML++ sử dụng thể hiện mức độ hiệu quả ngay khi chỉ sử dụng có 2 nhân CPU khi so với MCML(12).

Từ các kết quả so sánh nhằm đánh giá về thời gian khai thác, có thể nhận thấy mô hình song song hóa được đề xuất trong nội dung nghiên cứu này mang lại hiệu quả rõ rệt so với mô hình song song hóa mà giải thuật MCML-Miner sử dụng. MCML+ và MCML++ luôn cho thời gian khai thác thấp hơn MCML-Miner, cũng là một giải thuật song song, đặc biệt trên các CSDL có số lượng giao dịch lớn như Chainstore, Chainstore4X và SUSY.

Một cách tổng quát, giải thuật MCML++ có thời gian khai thác là thấp nhất trong cả bốn CSDL thực nghiệm do MCML++ được áp dụng đầy đủ mô hình song song hóa được đề xuất trong **Nội dung nghiên cứu 3**.

4.3.3 Đánh giá về mức độ sử dụng bộ nhớ





Hình 4.3. Lượng bộ nhớ sử dụng của các giải thuật [CT2]

Đối với mức độ sử dụng bộ nhớ của các giải thuật, kết quả thực nghiệm được thể hiện tại Hình 4.3. Có thể thấy, khi ngưỡng $minutil$ được hạ xuống, lượng bộ nhớ sử dụng cũng tăng theo. MCML++ cho thấy lượng bộ nhớ cần thiết để hoạt động là cao hơn so với MCM+ và MCML-Miner. Càng sử dụng nhiều nhân CPU cho quá trình song song, càng nhiều bộ nhớ phải được cấp phát. Điều này thể hiện rõ khi áp dụng lên các CSDL thưa và chứa nhiều hạng mục như Chainstore (Hình 4.3b). Tuy nhiên trên các CSDL có mật độ cao như Connect (Hình 4.3a), Chainstore4X (Hình 4.3c) và SUSY (Hình 4.3d), mức độ chênh lệch dao động từ 1.0GB đến dưới 5.0GB. Lượng bộ nhớ tối đa cấp phát cho MCML++(12), MCML+(12) và MCML(12) ngưỡng $minutil = 1.1E + 9$ thấp nhất quan sát được ở SUSY lần lượt là 18.3GB, 17.4GB và 17.9GB (Hình 4.3d).

Điểm đáng lưu ý là SUSY (Hình 4.3d) là CSDL có kích thước khá lớn với 5 triệu dòng dữ liệu giao dịch. Ngoài ra, CSDL dữ liệu này còn được bổ sung thêm một taxonomy có 4 cấp với 63 hạng mục tổng quát. Không gian lưu trữ cần thiết cho toàn bộ CSDL phân cấp này vì thế cũng gia tăng theo.

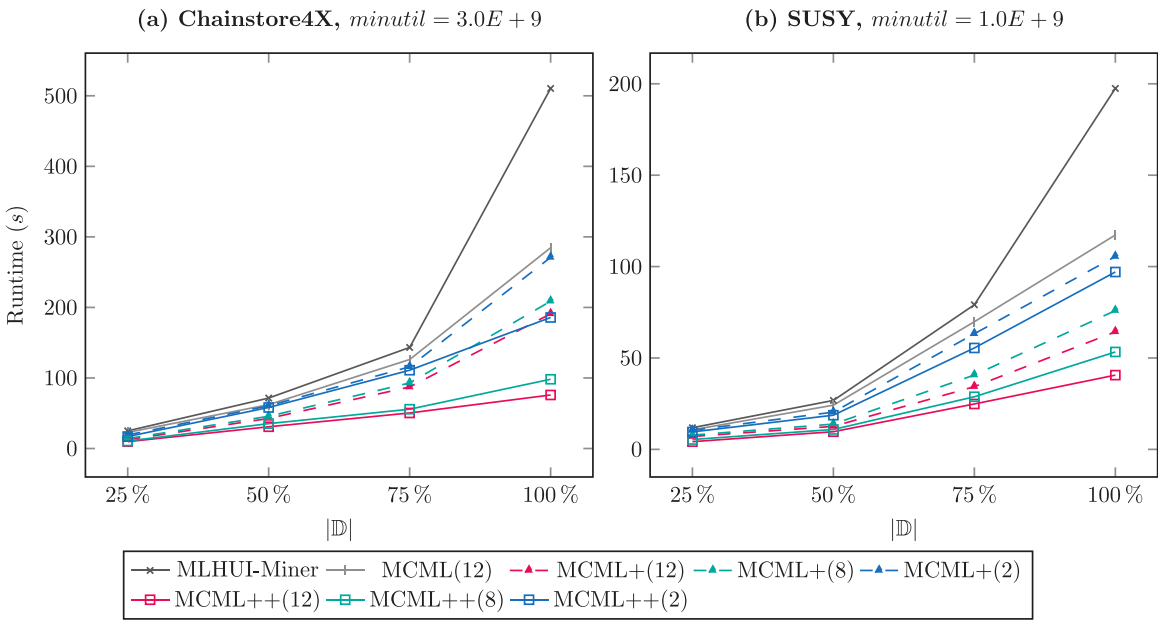
Lượng bộ nhớ cần thiết khi khai thác không có sử dụng các chiến lược song song hoá ở $minutil = 1.0E + 9$ đối với giải thuật MLHUI-Miner là xấp xỉ 18GB. Việc áp các chiến lược song song hoá đã nêu với 12 nhân chỉ làm tăng lượng bộ nhớ sử dụng lên xấp xỉ 2.0GB để đạt được mức độ cải thiện hiệu năng khai thác lên gần gấp 5 lần.

Ngoài ra, cho đến nay, qua khảo sát có rất ít các giải thuật khai thác HUIM có thể làm việc với CSDL lớn như SUSY, chưa kể đến việc tích hợp taxonomy để phục vụ cho bài toán khai thác tập mục hữu ích cao đa mức. Qua đó, cho thấy giải thuật đề xuất là MCML+ và MCML++ có thể làm việc với các CSDL lớn hơn nữa cho đến khi chạm ngưỡng giới hạn bộ nhớ của hệ thống thực nghiệm.

Trung bình, giải thuật MCML++ cho thấy có mức sử dụng bộ nhớ cao hơn MCML-Miner từ 1.1 đến 1.2 lần. Tuy nhiên, thời gian cải thiện giữa MCML-Miner và MCML++ lên đến trên 8.0 lần.

4.3.4 Đánh giá về khả năng thích nghi với việc mở rộng CSDL

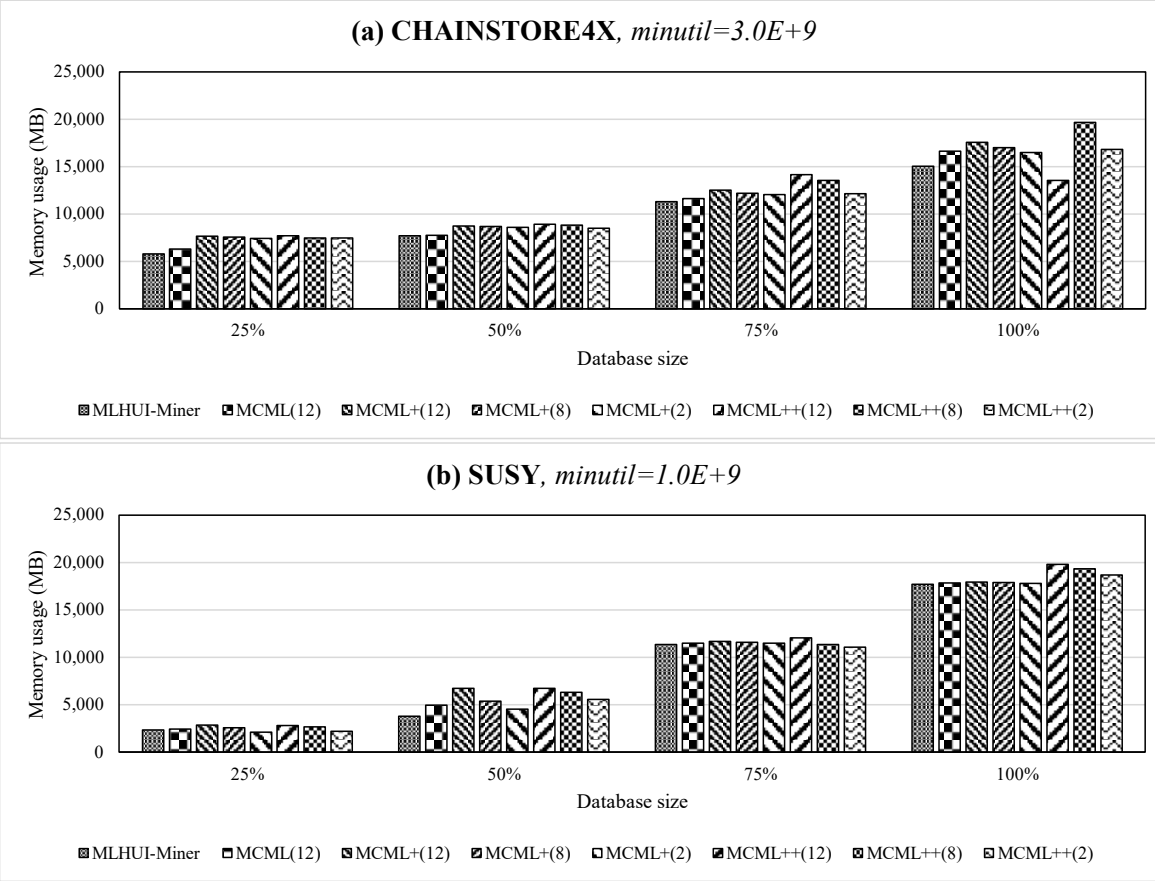
Đối với khả năng mở rộng, giải thuật MCML+ và MCML++ cũng được đánh giá trên hai CSDL lớn là Chainstore4X và SUSY. Ngưỡng $minutil$ sẽ được cố định $minutil = 3.0E + 9$ đối với Chainstore4X và $minutil = 1.1E + 9$ đối với SUSY. CSDL $\mathbb{D}$ sẽ được gia tăng kích thức theo các tỷ lệ từ 25% đến 100% kích thước ban đầu. Thời gian thực hiện được cho tại Hình 4.4 và lượng bộ nhớ sử dụng được cho tại Hình 4.5.



Hình 4.4. Thời gian thực hiện khi mở rộng CSDL [CT2]

Đối với thời gian thực hiện, khi kích thước CSDL gia tăng, thời gian thực hiện của các giải thuật dần phân hoá rõ rệt và cũng có xu hướng tương tự như trong phép thực nghiệm về thời gian thực hiện. MCML++(2), MCML++(8) và MCML++(12) vẫn có hiệu năng vượt trội trong các phép thực nghiệm này. Chỉ với hai nhân CPU được cấp phát, MCML++ cho hiệu quả tốt hơn hẳn so với MCML-Miner. Hơn nữa, thời gian thực hiện gia tăng gần như tuyến tính khi kích thước CSDL tăng, đặc biệt khi được cấp phát nhiều nhân CPU. Điều này thể hiện rõ ở mốc tỷ lệ 100% trên cả hai CSDL thực nghiệm.

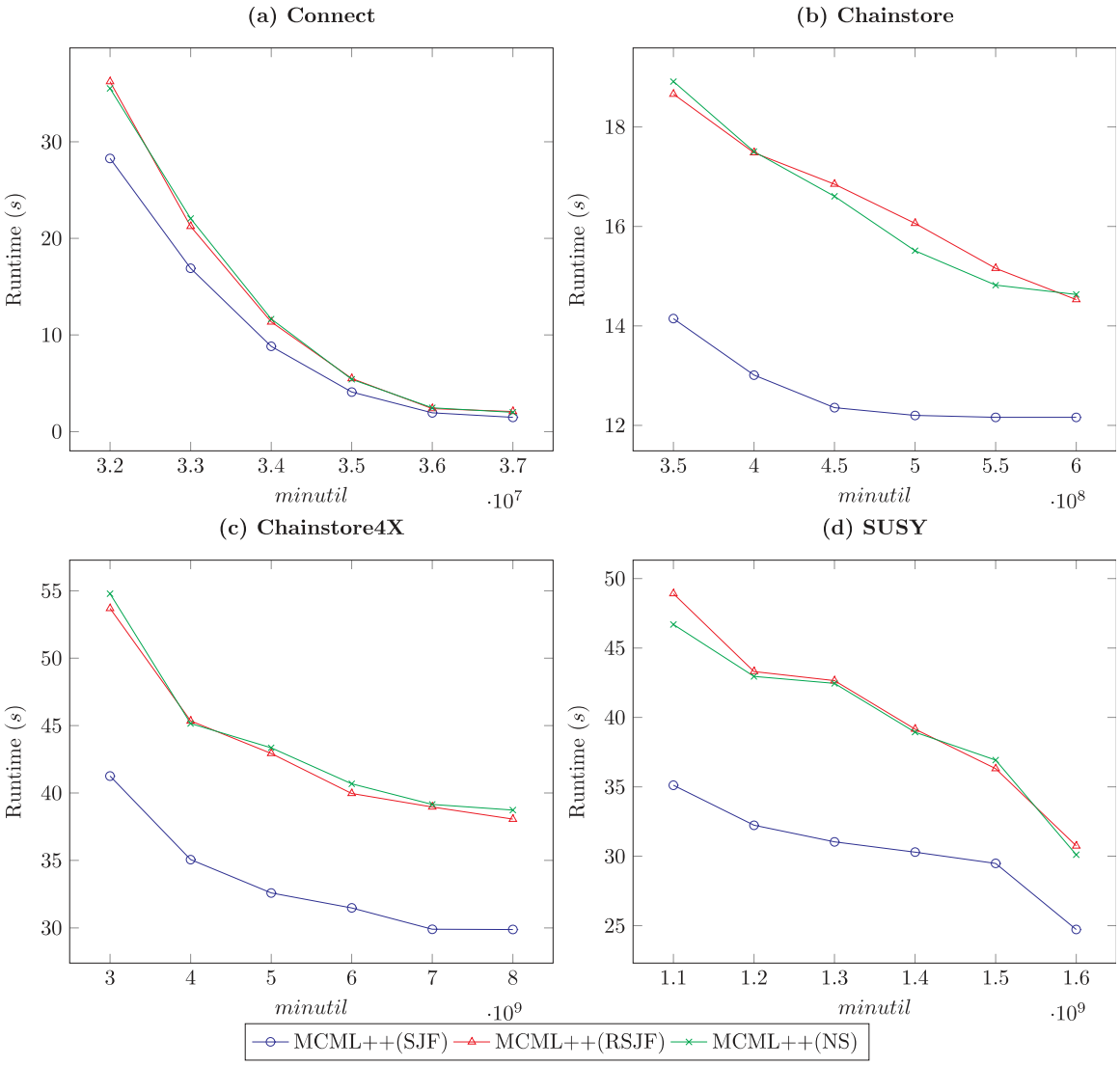
Đối với lượng bộ nhớ sử dụng, khi kích thước CSDL gia tăng, lượng bộ nhớ cần thiết để hoạt động cũng tăng theo. Về tổng thể, MCML++ cho thấy lượng bộ nhớ cần thiết cao hơn không đáng kể so với MCML+ và MCML-Miner. Tương tự, càng được cấp phát nhiều nhân, lượng bộ nhớ sử dụng cũng tăng theo để duy trì độ ổn định của giải thuật song song.



Hình 4.5. Lượng bộ nhớ sử dụng khi mở rộng CSDL [CT2]

4.3.5 *So sánh giữa các chiến lược điều phối*

Để đánh giá mức độ hiệu quả của chiến lược điều phối đã đề xuất là SJF, giải thuật MCML++ cũng được đánh giá trên cả bốn CSDL nói trên. Số nhân CPU được cố định tại $\rho = 12$.



Hình 4.6. Các chiến lược điều phối được sử dụng [CT2]

Trong thực nghiệm này, thứ tự của hàng đợi ưu tiên sử dụng trong Chiến lược 1 được ấn định như sau: chiều tăng dần của không gian tìm kiếm song song (SJF), ký hiệu MCML++(SJF), chiều không tăng của không gian tìm kiếm song song (Reversed Shortest Jobs First), ký hiệu MCML++(RSJF). Ngoài ra, chiến lược điều phối đơn giản First Come, First Served cũng được đánh giá và được ký hiệu là MCML++(NS).

Khi đó, hàng đợi ưu tiên Q trở thành hàng đợi thông thường để vô hiệu hoá việc điều phối. Kết quả được thể hiện tại Hình 4.6.

Có thể quan sát được rằng MCML++(SJF) mang lại hiệu quả vượt trội so với MCML++(RSJF) và MCML++(NS) trong tất cả các thử nghiệm.

Sự khác biệt về hiệu năng là đáng kể đối với các CSDL lớn như *Chainstore* (Hình 4.6b), *Chainstore4X* (Hình 4.6c) và *SUSY* (Hình 4.6d). MCML++(RSJF) và MCML++(NS) có hiệu quả gần như tương tự nhau. Điều này là do MCML++(RSJF) sử dụng chiến lược điều phối có cấu trúc tương tự như một cây liệt kê tập hợp [58], và có xu hướng lệch về bên trái. Đối với cơ chế điều phối này, không gian lớn được ưu tiên xử lý trước và sẽ chiếm nhiều thời gian CPU trong khi các không gian bé hơn phải chờ đến lượt được xử lý trong hàng đợi. Đối với MCML++(NS), đây là chiến lược điều phối đơn giản *đến trước, phục vụ trước* và phụ thuộc hoàn toàn vào hệ thống sẽ cho phép CPU xử lý không gian tìm kiếm nào. Điều này diễn ra ngẫu nhiên và cũng được thể hiện trên Hình 4.6, hiệu quả của chiến lược này là không ổn định khi so với MCML++(RSJF). Nhìn chung, MCML++(SJF) đã thể hiện được tính hiệu quả của chiến lược điều phối SJF được áp dụng.

4.4 Kết chương

Nội dung nghiên cứu này tập trung vào bài toán khai thác song song tập mục hữu ích cao đa mức trên CSDL có phân cấp các hạng mục và đề xuất các phương pháp để giải quyết hiệu quả bài toán này.

Hai giải thuật MCML+ và MCML++ đã được đề xuất và là các mở rộng của giải thuật MCML-Miner, kết hợp với các chiến lược xử lý song song hiệu quả để tận dụng tối đa năng lực xử lý của các vi xử lý đa nhân.

Các giai đoạn như tỉa các hạng mục không tiềm năng, xây dựng EUCS và cả quá trình quét không gian bài toán cũng được song song hoá hoàn toàn. Ngoài ra, một chiến lược điều phối SJF cũng được áp dụng để kiểm soát quá trình thực hiện các tác vụ khai thác song song.

Hàng loạt các thực nghiệm đã được thực hiện để đánh giá hiệu quả của các phương pháp đề xuất so với giải thuật ban đầu trên nhiều CSDL. Kết quả cho thấy MCML+ và MCML++ đều có hiệu quả tốt hơn đáng kể so với MCML-Miner. Chỉ với hai nhân xử lý được cấp phát trên các CSDL lớn, MCML++ vẫn có hiệu quả vượt trội so với MCML-Miner với nhiều nhân xử lý hơn. Phép thử khả năng mở rộng cũng cho thấy kết quả tương tự.

Về mức độ sử dụng bộ nhớ, MCML+ và MCML++ cho thấy cần nhiều bộ nhớ hơn so với MCML-Miner. Tuy nhiên, qua thực nghiệm cho thấy sự chênh lệch này chỉ từ 1.1 đến 1.2 lần nhưng mang lại sự cải thiện về thời gian lên đến trên 8 lần so với giải thuật MCML-Miner.

Thực nghiệm cho thấy MCML+ và MCML++ thực sự hiệu quả trên các CSDL có số lượng giao dịch lớn, từ một đến năm triệu giao dịch. Khi đó, các nhân vi xử lý sẽ được tận dụng triệt để vào các giai đoạn khác nhau quá trình khai thác.

Với các nghiên cứu nhằm nâng cao hiệu quả của quá trình khai thác song song các tập mục hữu ích cao đa mức nêu trên, các phương pháp đã đề xuất này vẫn có khả năng mở rộng trong tương lai.

- Trước tiên, việc phân chia không gian tìm kiếm của các giải thuật trên đều mất cân bằng trong đa số trường hợp, các tác vụ khai thác sẽ kết thúc ở các thời điểm khác nhau. Vì vậy ta có thể tìm hiểu cách để phân chia tải các công việc một cách cân bằng hơn thông qua các chiến lược cân bằng tải.
- Kế đến, các giải thuật đã đề xuất chỉ hoạt động trên một hệ thống tính toán đơn, có năng lực xử lý và bộ nhớ giới hạn. Vì vậy, giải pháp tiếp theo là mở rộng giải thuật để có thể hoạt động trên các CSDL có kích thước lớn.

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Thảo luận

5.1.1 So sánh với các nghiên cứu trước đây

Trong quá trình thực hiện luận án, dựa trên các câu hỏi nghiên cứu đã khảo sát thông qua 3 nội dung nghiên cứu đã mô tả tại phần MỞ ĐẦU. Luận án đã đề xuất hai mô hình xử lý song song cùng với các chiến lược và kỹ thuật giúp cải thiện hiệu năng của bài toán khai thác tập mục hữu ích cao trên các loại CSDL như định dạng CSDL có sự biến động về độ hữu ích, CSDL có sự phân cấp các hạng mục.

Các giải thuật như iEFIM-Closed [CT4] thuộc **Nội dung nghiên cứu 1**, MCML-Miner [CT3], PMLC-Miner thuộc **Nội dung nghiên cứu 2**, MCML+, và MCML++ [CT2] thuộc **Nội dung nghiên cứu 3** đã được đề xuất và so sánh với các giải thuật khác trong cùng lĩnh vực khai thác tập mục hữu ích cao. Kết quả thực nghiệm cho thấy rằng các giải thuật mới này có những ưu điểm vượt trội so với các giải thuật trước đây.

Các giải thuật truyền thống thường không tận dụng được khả năng xử lý song song của các vi xử lý đa nhân, dẫn đến thời gian thực hiện dài và hiệu năng không cao khi áp dụng trên các cơ sở dữ liệu lớn. Trong khi đó, các giải thuật mới được đề xuất trong luận án đã khắc phục được hạn chế này, đặc biệt là trên các CSDL có cấu trúc phân cấp phức tạp. Điều này được thể hiện rõ rệt tại các **Nội dung nghiên cứu 2** và **Nội dung nghiên cứu 3** thông qua bốn giải thuật song song hóa đã được đề xuất, dựa trên hai mô hình xử lý song song trong CHƯƠNG 3 và CHƯƠNG 4.

Thông qua **Nội dung nghiên cứu 1**, luận án đã đề xuất một giải thuật nhằm hướng đến việc thu gọn tập kết quả của quá trình khai thác thông qua biểu diễn đóng của tập mục hữu ích cao. Giải thuật iEFIM-Closed kết hợp với các chiến lược hiệu quả nhằm giảm chi phí quét CSDL, tỉa ứng viên cho thấy mức độ cải thiện đáng kể về hiệu năng khai thác tập mục hữu ích cao đóng. Tuy nhiên, việc áp dụng mô hình xử lý song song lên giải thuật này là không khả thi do xuất hiện các ràng buộc dữ liệu. Bên cạnh

đó, iEFIM-Closed chỉ hoạt động trên định dạng CSDL giao dịch thông thường và không xem xét đến sự phân cấp của các hạng mục. Đây cũng là mục tiêu của **Nội dung nghiên cứu 2** và **Nội dung nghiên cứu 3**, tiếp tục khảo sát và giải quyết các tồn tại của **Nội dung nghiên cứu 1**.

Đối với **Nội dung nghiên cứu 2**, so sánh với các giải thuật như tuần tự như CLS-Miner, CHUI-Miner, FHM, và các phương pháp khác, các giải thuật MCML-Miner [CT3] và PMLC-Miner không chỉ cải thiện thời gian thực hiện mà còn giảm thiểu lượng bộ nhớ cần thiết cho quá trình khai thác trên một số CSDL. Điều này chứng tỏ rằng việc áp dụng mô hình xử lý song song và các chiến lược tối ưu hóa không gian tìm kiếm đã mang lại hiệu quả cao trong việc khai thác tập mục hữu ích cao.

Tương tự với **Nội dung nghiên cứu 3**, MCML+ [CT2] và MCML++ [CT2] mang lại sự cải thiện vượt trội về hiệu năng khai thác khi mở rộng quá trình song song hóa trên các giai đoạn khác nhau của giải thuật. Đồng thời, quá trình khai thác cũng được cải tiến để không còn phụ thuộc vào cấu trúc taxonomy, từ đó cho mức độ tận dụng sức mạnh xử lý song song tốt hơn đáng kể.

5.1.2 Phân tích hiệu quả của các giải thuật đề xuất

Kết quả thực nghiệm đã chứng minh rằng các giải thuật mới có hiệu suất tốt hơn hẳn so với các giải thuật tuần tự trước đó và cả các phiên bản đã được song song hóa, đặc biệt là khi áp dụng trên các CSDL lớn và phức tạp như SUSY. Các giải thuật song song hoá như MCML-Miner [CT3], PMLC-Miner, MCML+ [CT2] và MCML++ [CT2] cho thấy sự cải thiện đáng kể về thời gian thực hiện, khi so sánh với các giải thuật không áp dụng mô hình xử lý song song.

Tuy nhiên, một số thực nghiệm trên CSDL nhỏ như Chess hoặc Connect cho thấy rằng các giải thuật mới không phải lúc nào cũng vượt trội. Mặc dù có thời gian thực hiện tốt, nhưng do kích thước nhỏ và mật độ cao CSDL Chess hoặc Connect, sự cải thiện về hiệu suất không rõ rệt so với các giải thuật truyền thống. Điều này gợi ý rằng việc chọn lựa mô hình và chiến lược khai thác phù hợp có thể phụ thuộc vào đặc điểm của từng CSDL cụ thể.

Với các kết quả đánh giá dựa trên thực nghiệm, có thể thấy được luận án đã giải quyết một cách hiệu quả các nội dung nghiên cứu đã đề ra, đặc biệt là với các nội dung liên quan đến việc áp dụng mô hình xử lý song song vào bài toán khai thác tập mục hữu ích cao.

5.1.3 Hạn chế của nghiên cứu

Mặc dù các giải thuật mới đề xuất đã cho thấy nhiều ưu điểm vượt trội, nhưng vẫn còn một số hạn chế cần được thảo luận. Đầu tiên, các giải thuật hiện tại chủ yếu được phát triển và tối ưu hóa cho các CSDL có kích thước lớn nhưng chưa thực sự tối ưu cho các CSDL nhỏ với mật độ cao như Chess hoặc Connect. Điều này cho thấy rằng mô hình song song hóa không phải lúc nào cũng là lựa chọn tốt nhất cho mọi loại CSDL.

Thứ hai, các mô hình xử lý song song trong nghiên cứu chỉ được triển khai trên các hệ thống đơn máy tính, chưa thực nghiệm trên môi trường phân tán. Điều này giới hạn khả năng áp dụng của các giải thuật trong bối cảnh xử lý dữ liệu lớn (big data) khi kích thước CSDL vượt quá khả năng xử lý của một máy tính đơn lẻ.

Thứ ba, việc triển khai các giải thuật bằng ngôn ngữ lập trình Java có thể ảnh hưởng đến hiệu suất do các giới hạn của máy ảo Java (JVM) trong việc quản lý bộ nhớ và tối ưu hóa thời gian thực thi. Các thực nghiệm trong nghiên cứu này cho thấy rằng việc sử dụng Java là hợp lý, nhưng để đạt được hiệu suất tốt hơn, việc triển khai bằng các ngôn ngữ lập trình khác có thể là một hướng đi cần xem xét.

5.1.4 Vấn đề phát sinh và cách giải quyết

Trong quá trình thực hiện nghiên cứu, một số vấn đề đã phát sinh, đòi hỏi phải có các điều chỉnh trong quá trình triển khai. Một trong những vấn đề chính là sự không cân bằng trong phân phối tải giữa các nhân xử lý khi áp dụng mô hình song song hóa. Việc này đã được giải quyết thông qua các chiến lược phân chia tải mới, giúp tối ưu hóa việc sử dụng các nhân xử lý, nhưng vẫn cần có những nghiên cứu sâu hơn để cải thiện thêm.

Ngoài ra, trong quá trình xử lý các CSDL lớn, vấn đề về bộ nhớ cũng đã trở thành một thách thức. Để giải quyết, các kỹ thuật quản lý bộ nhớ động và tối ưu hóa việc sử dụng bộ nhớ đã được áp dụng. Điều này đã giúp giảm thiểu mức sử dụng bộ nhớ, nhưng vẫn còn nhiều cơ hội để cải thiện, đặc biệt là khi xử lý các CSDL siêu lớn.

5.1.5 Ý nghĩa của nghiên cứu

Những kết quả đạt được từ nghiên cứu không chỉ mang ý nghĩa về mặt học thuật mà còn có giá trị thực tiễn cao. Việc phát triển các giải thuật hiệu quả hơn trong khai thác tập mục hữu ích cao có thể đóng góp quan trọng vào việc phân tích và xử lý dữ liệu trong các lĩnh vực như thương mại điện tử, tài chính, và y tế.

Đặc biệt, các kết quả này đã mở ra một hướng đi mới trong việc áp dụng mô hình xử lý song song vào khai thác dữ liệu, tận dụng sức mạnh của các vi xử lý đa nhân để cải thiện hiệu suất. Điều này không chỉ giúp giảm thời gian xử lý mà còn tạo điều kiện cho việc xử lý các CSDL lớn hơn, phức tạp hơn, đáp ứng nhu cầu ngày càng tăng của các hệ thống phân tích dữ liệu hiện đại.

5.2 Kết luận

Trong suốt quá trình nghiên cứu và triển khai, nghiên cứu sinh đã đạt được nhiều kết quả đáng kể thông qua việc đề xuất và phát triển hàng loạt các giải thuật khai thác tập mục hữu ích cao (HUIM), tập mục hữu ích cao tổng quát, và kết hợp các bài toán này với mô hình xử lý song song sử dụng kiến trúc tính toán đa nhân. Đây là một hướng đi mới mẻ và có tiềm năng lớn trong việc khai thác và ứng dụng các hệ thống tính toán hiện đại.

Một trong những mục tiêu chính của luận án là phát triển các giải thuật hiệu quả, tận dụng tối đa khả năng tính toán của các vi xử lý đa nhân. Các hệ thống tính toán đa nhân ngày càng trở nên phổ biến và chi phí thấp, nhưng tiềm năng của chúng chưa được khai thác triệt để trong lĩnh vực khai thác dữ liệu. Qua các nghiên cứu và thực nghiệm được thực hiện, các giải thuật mới đã cho thấy sự cải thiện hiệu năng rõ rệt, đặc biệt là khi áp dụng mô hình xử lý song song đa nhân.

Cụ thể, các giải thuật như MCML-Miner, PMLC-Miner, MCML+, và MCML++ đã được phát triển với mục tiêu tối ưu hóa quá trình khai thác tập mục hữu ích cao trong các cơ sở dữ liệu[1] (CSDL) phân cấp[2], áp dụng chiến lược xử lý song song trên các vi xử lý đa nhân. Thực nghiệm đã chứng minh rằng các giải thuật này cải thiện đáng kể về mặt thời gian khai thác, sử dụng bộ nhớ, và khả năng mở rộng so với các giải thuật không áp dụng mô hình xử lý song song.

Trong quá trình thực hiện, tất cả các giải thuật đều được triển khai bằng ngôn ngữ lập trình Java, và để đảm bảo tính khách quan, thư viện lập trình Java được sử dụng để ghi nhận các thông số về bộ nhớ và thời gian thực hiện. Các thực nghiệm được lặp lại nhiều lần để đảm bảo độ chính xác và tính thống nhất của kết quả.

Các kết quả này đã được trình bày thông qua các chuyên đề nghiên cứu đã báo cáo. Các giải thuật đã đề xuất được liệt kê tại Bảng 5.1.

Bảng 5.1. Các giải thuật khai thác đã được đề xuất

STT	Giải thuật	Công trình	Đóng góp		
			Nội dung 1	Nội dung 2	Nội dung 3
1	**iEFIM-Closed**	CT4	✓		
2	**MCML-Miner**	CT3		✓	✓
3	**MLC-Miner**	CT1	✓	✓	
4	**PMLC-Miner**		✓	✓	✓
5	**MCML+**	CT2		✓	✓
6	**MCML++**	CT2		✓	✓

[1] Nguồn: Thư viện khai thác dữ liệu mã nguồn mở SPMF.

[2] Nguồn: https://github.com/dzutrinh/MLC-Miner/tree/main/taxonomy

Một điểm đặc biệt đáng chú ý là CSDL Chess, mặc dù có kích thước nhỏ, nhưng lại là một thách thức lớn do mật độ cao và chiều dài giao dịch lớn. Tuy nhiên, như đã đề cập, Chess không phải là một CSDL giao dịch thực sự mà là một tập dữ liệu tổng hợp nhằm mục đích đánh giá hiệu năng của các giải thuật.

Ngoài Chess, các CSDL lớn như Chainstore và SUSY cũng được sử dụng để đánh giá hiệu năng. SUSY là một CSDL rất lớn, mà ít có giải thuật HUIM nào có thể xử lý. Các thực nghiệm cho thấy các giải thuật mới đề xuất có khả năng xử lý hiệu quả ngay cả trên các CSDL lớn với cấu trúc phân cấp phức tạp.

Về mặt phương pháp luận, các kỹ thuật tối ưu hóa không gian tìm kiếm, mô hình song song hóa quá trình khai thác, và các chiến lược điều phối đã được phát triển và chứng minh bằng cả mô hình toán học và thực nghiệm. Các giải thuật này không chỉ cải thiện thời gian khai thác mà còn mở ra hướng đi mới cho việc tận dụng hiệu quả khả năng xử lý của các vi xử lý đa nhân trong các hệ thống tính toán hiện đại.

Những đóng góp của luận án không chỉ có ý nghĩa trong việc giải quyết các vấn đề hiện tại mà còn tạo ra nền tảng vững chắc cho các nghiên cứu và ứng dụng trong tương lai. Những phương pháp và giải thuật này có thể được sử dụng làm nền tảng để phát triển các hệ thống khai thác dữ liệu tiên tiến, hiệu quả hơn trong các lĩnh vực khác nhau, từ thương mại điện tử đến phân tích dữ liệu lớn.

5.3 Hướng phát triển trong tương lai

Mặc dù các giải thuật đề xuất đã đạt được những kết quả đáng khích lệ, nhưng vẫn còn nhiều hướng phát triển có thể được xem xét trong tương lai để cải thiện hơn nữa hiệu quả và ứng dụng của các giải thuật này.

5.3.1 Nghiên cứu và phát triển chiến lược cân bằng tải

Một vấn đề lớn khi triển khai các giải thuật khai thác theo chiều sâu là không gian tìm kiếm thường không đồng đều, dẫn đến các tác vụ khai thác có thời gian thực hiện khác nhau. Điều này tạo ra sự không cân bằng trong việc phân phối tải giữa các nhân xử lý, làm giảm hiệu quả tổng thể của quá trình khai thác song song. Để giải quyết

vấn đề này, cần nghiên cứu và phát triển các chiến lược cân bằng tải mới, nhằm đảm bảo các nhân xử lý được sử dụng hiệu quả hơn, giảm thiểu thời gian chờ đợi và tối ưu hóa hiệu suất của toàn bộ hệ thống.

5.3.2 Mở rộng mô hình xử lý song song trên môi trường phân tán

Các mô hình xử lý song song hiện tại chủ yếu tập trung vào một hệ thống tính toán đơn, trong khi xu hướng công nghệ ngày nay đòi hỏi khả năng xử lý và lưu trữ ngày càng lớn. Với sự phát triển của các hệ thống "big data," việc mở rộng các giải thuật để hoạt động trên môi trường phân tán là một hướng đi cần thiết. Trong đó, các giải thuật cần được thiết kế sao cho có thể khai thác dữ liệu đồng thời trên nhiều máy tính trong mạng lưới phân tán, tận dụng sức mạnh tính toán của các hệ thống máy tính hiện đại có số lượng nhân tính toán rất lớn. Điều này cũng đòi hỏi việc phát triển các kỹ thuật mới để phân tán dữ liệu và phối hợp giữa các máy tính trong môi trường phân tán một cách hiệu quả.

5.3.3 Tích hợp các phương pháp học sâu vào khai thác dữ liệu

Một hướng phát triển khác có tiềm năng là kết hợp các giải thuật khai thác tập mục hữu ích cao với các phương pháp học sâu (deep learning). Các mô hình học sâu hiện nay đã chứng minh khả năng vượt trội trong việc phân tích và nhận dạng mẫu từ các tập dữ liệu lớn và phức tạp. Việc tích hợp các phương pháp này có thể giúp cải thiện khả năng khai thác, nhận dạng và dự đoán các mẫu hữu ích trong các CSDL lớn và phức tạp hơn. Đặc biệt, các mô hình học sâu có thể hỗ trợ trong việc tự động hóa quá trình khai thác, giúp nhận diện các mẫu ẩn khó phát hiện mà các phương pháp truyền thống khó tiếp cận.

5.3.4 Ứng dụng trong các lĩnh vực thực tiễn

Mở rộng ứng dụng của các giải thuật đã phát triển trong các lĩnh vực thực tiễn như thương mại điện tử, tài chính, chăm sóc sức khỏe, và an ninh mạng cũng là một hướng phát triển đáng chú ý. Trong mỗi lĩnh vực, các giải thuật này có thể được tùy chỉnh và tối ưu hóa để đáp ứng các yêu cầu cụ thể, chẳng hạn như phát hiện gian lận trong

tài chính, dự đoán nhu cầu sản phẩm trong thương mại điện tử, hoặc phân tích dữ liệu y tế để hỗ trợ chẩn đoán và điều trị.

5.3.5 Tăng cường tính khả mở của giải thuật

Một yếu tố quan trọng khác cần xem xét là khả năng mở rộng của các giải thuật khi đối mặt với các CSDL ngày càng lớn. Việc phát triển các giải thuật với khả năng mở rộng tốt, đồng thời tối ưu hóa việc sử dụng tài nguyên tính toán và bộ nhớ, là một thách thức lớn nhưng cần thiết. Điều này không chỉ giúp các giải thuật có thể áp dụng rộng rãi hơn mà còn nâng cao hiệu suất trong các môi trường tính toán đa dạng, từ các hệ thống máy tính đơn giản đến các hệ thống tính toán hiệu năng cao.

5.3.6 Phân tích và cải thiện các tiêu chí đánh giá

Cuối cùng, cần tiếp tục phân tích và cải thiện các tiêu chí đánh giá hiệu quả của các giải thuật, chẳng hạn như thời gian thực hiện, lượng bộ nhớ sử dụng, và số lượng ứng viên phải xét. Việc đánh giá một cách toàn diện và khách quan các giải thuật không chỉ giúp cải thiện chất lượng của các nghiên cứu mà còn tạo điều kiện cho việc so sánh và đánh giá giữa các giải thuật khác nhau một cách chính xác.

Nhìn chung, thông qua 3 nội dung nghiên cứu chính, luận án đã đạt được nhiều kết quả đáng kể trong việc phát triển các giải thuật khai thác tập mục hữu ích cao và ứng dụng mô hình tính toán song song trên vi xử lý đa nhân. Những đóng góp của luận án không chỉ giúp cải thiện hiệu suất khai thác dữ liệu mà còn mở ra nhiều hướng nghiên cứu mới, có tiềm năng lớn trong tương lai. Việc tiếp tục nghiên cứu và phát triển các giải thuật này, cùng với việc mở rộng ứng dụng trong các lĩnh vực thực tiễn, sẽ góp phần quan trọng vào sự phát triển của lĩnh vực khai thác dữ liệu và các ngành liên quan.

DANH MỤC CÔNG TRÌNH ĐÃ CÔNG BỐ CỦA HỌC VIÊN

1. Các công trình là tác giả chính

[CT1]. **T.D.D. Nguyen**, N.T. Tung, L.T.T. Nguyen, T.T. Pham, B. Vo, MLC-miner: Efficiently discovering multi-level closed high utility patterns from quantitative hierarchical transaction databases, Expert Syst Appl 254 (2024) 124383.
https://doi.org/10.1016/j.eswa.2024.124383

[CT2]. **T.D.D. Nguyen**, N.T. Tung, T. Pham, L.T.T. Nguyen, Parallel approaches to extract multi-level high utility itemsets from hierarchical transaction databases, Knowl Based Syst 276 (2023) 110733.
https://doi.org/10.1016/j.knosys.2023.110733.

[CT3]. **T.D.D. Nguyen**, L.T.T. Nguyen, A. Kozierkiewicz, T. Pham, B. Vo, An Efficient Approach for Mining High-Utility Itemsets from Multiple Abstraction Levels, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer International Publishing, Cham, 2021: pp. 92–103.
https://doi.org/10.1007/978-3-030-73280-6_8.

[CT4]. **T.D.D. Nguyen**, L.T.T. Nguyen, L. Vu, B. Vo, W. Pedrycz, Efficient algorithms for mining closed high utility itemsets in dynamic profit databases, Expert Syst Appl 186 (2021) 115741.
https://doi.org/10.1016/j.eswa.2021.115741

2. Các công trình đồng tác giả

[CT5]. T.M. Le, **T.D.D. Nguyen**, L.T.T. Nguyen, A. Kozierkiewicz, N.T. Tung, Extracting Top-k High Utility Patterns from Multi-level Transaction Databases, in: Intelligent Information and Database Systems, Springer Nature Singapore, 2023: pp. 296–306.
https://doi.org/10.1007/978-981-99-5834-4_24

TÀI LIỆU THAM KHẢO

[1] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, ACM SIGMOD Record 22 (1993) 207–216.

[2] H. Nguyen, T. Le, M. Nguyen, P. Fournier-Viger, V.S. Tseng, B. Vo, Mining frequent weighted utility itemsets in hierarchical quantitative databases, Knowl Based Syst (2021) 107709. https://doi.org/https://doi.org/10.1016/j.knosys.2021.107709.

[3] B. Vo, B. Le, Fast algorithm for mining generalized association rules, International Journal of Database Theory and Application 2 (2009) 1–12.

[4] R. Srikant, R. Agrawal, Mining generalized association rules, Future Generation Computer Systems 13 (1997) 161–180.

[5] B. Vo, H. Nguyen, T.B. Ho, B. Le, Parallel method for mining high utility itemsets from vertically partitioned distributed databases, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009: pp. 251–260.

[6] H. Bui, B. Vo, H. Nguyen, WUN-Miner: A new method for mining frequent Weighted Utility itemsets, in: 2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016, 2017: pp. 1365–1370.

[7] L.T.T. Nguyen, T. Trinh, N.T. Nguyen, B. Vo, A method for mining top-rank- k frequent closed itemsets, Journal of Intelligent and Fuzzy Systems 32 (2017) 1297–1305. https://doi.org/10.3233/JIFS-169128.

[8] B. Vo, T. Le, F. Coenen, T.P. Hong, Mining frequent itemsets using the N-list and subsume concepts, International Journal of Machine Learning and Cybernetics 7 (2016) 253–265. https://doi.org/10.1007/s13042-014-0252-2.

[9] B. Vo, T.P. Hong, B. Le, DBV-Miner: A Dynamic Bit-Vector approach for fast mining frequent closed itemsets, Expert Syst Appl 39 (2012) 7196–7206.

[10] M. Ledmi, S. Zidat, A. Hamdi-Cherif, GrAFCI+ A fast generator-based algorithm for mining frequent closed itemsets, Knowl Inf Syst 63 (2021) 1873–1908. https://doi.org/10.1007/s10115-021-01575-3.

[11] S. Raj, D. Ramesh, M. Sreenu, K.K. Sethi, EAFIM: efficient apriori-based frequent itemset mining algorithm on Spark for big transactional data, Knowl Inf Syst 62 (2020) 3565–3583. https://doi.org/10.1007/s10115-020-01464-1.

[12] K.-W. Chon, E. Yi, M.-S. Kim, SGMiner: A fast and scalable GPU-based frequent pattern miner on SSDs, IEEE Access 10 (2022) 62502–62519. https://doi.org/10.1109/ACCESS.2022.3179592.

[13] H. Yao, H.J. Hamilton, G.J. Butz, A foundational approach to mining itemset utilities from databases, in: SIAM International Conference on Data Mining, 2004: pp. 482–486.

[14] Y. Liu, W.K. Liao, A. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, in: 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, Springer-Verlag, 2005: pp. 689–695.

[15] B. Le, H. Nguyen, B. Vo, An efficient strategy for mining high utility itemsets, International Journal of Intelligent Information and Database Systems 5 (2011) 164–176.

[16] P. Fournier-Viger, C.W. Wu, S. Zida, V.S. Tseng, FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning, in: International Symposium on Methodologies for Intelligent Systems, 2014: pp. 83–92.

[17] L.T.T. Nguyen, P. Nguyen, T.D.D. Nguyen, B. Vo, P. Fournier-Viger, V.S. Tseng, Mining high-utility itemsets in dynamic profit databases, Knowl Based Syst 175 (2019) 130–144.

[18] P. Fournier-Viger, S. Zida, J.C.W. Lin, C.W. Wu, V.S. Tseng, EFIM-closed: Fast and memory efficient discovery of closed high-utility itemsets, in: Machine Learning and Data Mining in Pattern Recognition, Springer, 2016: pp. 199–213.

[19] L.T.T. Nguyen, V. V. Vu, M.T.H. Lam, T.T.M. Duong, L.T. Manh, T.T.T. Nguyen, B. Vo, H. Fujita, An efficient method for mining high utility closed itemsets, Inf Sci (N Y) 495 (2019) 78–99.

[20] V.S. Tseng, C.W. Wu, B.E. Shie, P.S. Yu, UP-Growth: An efficient algorithm for high utility itemset mining, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010: pp. 253–262.

[21] V.S. Tseng, B.E. Shie, C.W. Wu, P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, IEEE Trans Knowl Data Eng 25 (2013) 1772–1786.

[22] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules in Large Databases, in: 20th International Conference on Very Large Data Bases (VLDB '94), Morgan Kaufmann Publishers Inc., 1994: pp. 487–499.

[23] L.T.T. Nguyen, D.B. Vu, T.D.D. Nguyen, B. Vo, Mining maximal high utility itemsets on dynamic profit databases, Cybern Syst 51 (2020) 140–160.

[24] T.D.D. Nguyen, Q.B. Vu, L.T.T. Nguyen, Efficient algorithms for mining maximal high-utility itemsets, in: 2019 6th NAFOSTED Conference on Information and Computer Science, NICS 2019, Hanoi, 2019: pp. 428–433.

[25] H. Duong, T. Hoang, T. Tran, T. Truong, B. Le, P. Fournier-Viger, Efficient algorithms for mining closed and maximal high utility itemsets, Knowl Based Syst 257 (2022) 109921. https://doi.org/https://doi.org/10.1016/j.knosys.2022.109921.

[26] B.E. Shie, P.S. Yu, V.S. Tseng, Efficient algorithms for mining maximal high utility itemsets from data streams with different models, Expert Syst Appl 39 (2012) 12947–12960.

[27] C.W. Wu, P. Fournier-Viger, J.Y. Gu, V.S. Tseng, Mining closed+ high utility itemsets without candidate generation, in: TAAI 2015 - 2015 Conference on Technologies and Applications of Artificial Intelligence, IEEE, 2016: pp. 187–194.

[28] T.-L. Dam, H. Ramampiaro, K. Nørvåg, Q.-H. Duong, Towards efficiently mining closed high utility itemsets from incremental databases, Knowl Based Syst 165 (2019) 13–29. https://doi.org/https://doi.org/10.1016/j.knosys.2018.11.019.

[29] T. Mai, L.T.T. Nguyen, An efficient approach for mining closed high utility itemsets and generators, Journal of Information and Telecommunication 1 (2017) 193–207.

[30] B. Vo, L.T.T. Nguyen, N. Bui, T.D.D. Nguyen, V.N. Huynh, T.P. Hong, An efficient method for mining closed potential high-utility itemsets, IEEE Access 8 (2020) 31813–31822.

[31] T.L. Dam, K. Li, P. Fournier-Viger, Q.H. Duong, CLS-Miner: efficient and effective closed high-utility itemset mining, Front Comput Sci 13 (2019) 357–381.

[32] J. Wang, J. Han, C. Li, Frequent Closed Sequence Mining without Candidate Maintenance, IEEE Trans Knowl Data Eng 19 (2007) 1042–1056. https://doi.org/10.1109/TKDE.2007.1043.

[33] K. Gouda, M.J. Zaki, GenMax: An efficient algorithm for mining maximal frequent itemsets, Data Min Knowl Discov 11 (2005) 223–242. https://doi.org/10.1007/S10618-005-0002-X/METRICS.

[34] S.Z. Ali, Y. Rathore, A effective and efficient algorithm for cross level frequent pattern mining, in: 2014 International Conference on Advances in Engineering and Technology Research, ICAETR 2014, 2014: pp. 1–6.

[35] E. Baralis, L. Cagliero, T. Cerquitelli, P. Garza, Generalized association rule mining with constraints, Inf Sci (N Y) 194 (2012) 68–84.

[36] E. Baralis, L. Cagliero, T. Cerquitelli, V. D'Elia, P. Garza, Expressive generalized itemsets, Inf Sci (N Y) 278 (2014) 327–343.

[37] J. Han, Y. Fu, Mining multiple-level association rules in large databases, IEEE Trans Knowl Data Eng 11 (1999) 798–805.

[38] Y.-C. Lee, T.-P. Hong, C.-H. Chen, Mining Generalized Association Rules with Quantitative Data under Multiple Support Constraints, in: J.-S. Pan, S.-M. Chen, N.T. Nguyen (Eds.), Computational Collective Intelligence. Technologies and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010: pp. 224–231.

[39] B. Liu, W. Hsu, Y. Ma, Mining Association Rules with Multiple Minimum Supports, in: 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, 1999: pp. 337–341.

[40] M.-C. Tseng, W.-Y. Lin, Efficient mining of generalized association rules with non-uniform minimum support, Data Knowl Eng 62 (2007) 41–64.

[41] L. Cagliero, S. Chiusano, P. Garza, G. Ricupero, Discovering high-utility itemsets at multiple abstraction levels, in: M. Kirikova, K. Nørvåg, G.A. Papadopoulos, J. Gamper, R. Wrembel, J. Darmont, S. Rizzi (Eds.), European Conference on Advances in Databases and Information Systems, Springer International Publishing, Cham, 2017: pp. 224–234.

[42] S. Alias, N.M. Norwawi, pSPADE: Mining sequential pattern using personalized support threshold value, in: International Symposium on Information Technology 2008, ITSim, 2008: pp. 1–8.

[43] S. Cong, J. Han, D. Padua, Parallel mining of closed sequential patterns, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005: pp. 562–567.

[44] T. Zhu, S. Bai, A parallel mining algorithm for closed sequential patterns, in: 21st International Conference on Advanced Information Networking and Applications Workshops/Symposia, AINAW'07, 2007: pp. 392–395.

[45] Y. Chen, A. An, Approximate parallel high utility itemset mining, Big Data Research 6 (2016) 26–42.

[46] K.K. Sethi, D. Ramesh, D.R. Edla, P-FHM+: parallel high utility itemset mining algorithm for big data processing, in: Procedia Comput Sci, 2018: pp. 918–927.

[47] H. Jie, W. Dehua, Design and Implementation of Automatic Selection of the Most Efficient Itemset Algorithm Based on Spark, Sci Program 2022 (2022) 3362203. https://doi.org/10.1155/2022/3362203.

[48] T.D.D. Nguyen, L.T.T. Nguyen, B. Vo, A parallel algorithm for mining high utility itemsets, in: International Conference on Information Systems Architecture and Technology, Springer Verlag, 2019: pp. 286–295. https://doi.org/10.1007/978-3-319-99996-8_26.

[49] B. Vo, L.T.T. Nguyen, T.D.D. Nguyen, P. Fournier-Viger, U. Yun, A Multi-Core Approach to Efficiently Mining High-Utility Itemsets in Dynamic Profit Databases, IEEE Access 8 (2020) 85890–85899.

[50] N.T. Tung, L.T.T. Nguyen, T.D.D. Nguyen, A. Kozierkiewicz, Cross-level high-utility itemset mining using multi-core processing, in: 13th International Conference on Computational Collective Intelligence, 2021: pp. 467–479.

[51] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: A frequent-pattern tree approach, Data Min Knowl Discov 8 (2004) 53–87.

[52] M.J. Zaki, Scalable algorithms for association mining, IEEE Trans Knowl Data Eng 12 (2000) 372–390.

[53] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, D. Yang, H-mine: hyper-structure mining of frequent patterns in large databases, in: 2001 IEEE International Conference on Data Mining, 2001: pp. 441–448. https://doi.org/10.1109/ICDM.2001.989550.

[54] T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets, in: FIMI, 2004.

[55] H. Yao, H.J. Hamilton, Mining itemset utilities from transaction databases, Data Knowl Eng 59 (2006) 603–626.

[56] B. Le, H. Nguyen, T.A. Cao, B. Vo, A novel algorithm for mining high utility itemsets, in: First Asian Conference on Intelligent Information and Database Systems, 2009: pp. 13–17.

[57] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, in: ACM International Conference Proceeding Series, 2012: pp. 55–64.

[58] S. Zida, P. Fournier-Viger, J.C.W. Lin, C.W. Wu, V.S. Tseng, EFIM: a fast and memory efficient algorithm for high-utility itemset mining, Knowl Inf Syst 51 (2017) 595–625.

[59] S. Krishnamoorthy, HMiner: Efficiently mining high utility itemsets, Expert Syst Appl 90 (2017) 168–183.

[60] C.F. Ahmed, S.K. Tanbeer, B.S. Jeong, Y.K. Lee, Efficient tree structures for high utility pattern mining in incremental databases, IEEE Trans Knowl Data Eng 21 (2009) 1708–1721.

[61] V.S. Tseng, B.E. Shie, C.W. Wu, P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, IEEE Trans Knowl Data Eng 25 (2013) 1772–1786. https://doi.org/10.1109/TKDE.2012.59.

[62] M.J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, New algorithms for fast discovery of association rules, in: D. Heckerman, H. Mannila, D. Pregibon (Eds.), 3rd International Conference on Knowledge Discovery and Data Mining (KDD-97), {AAAI} Press, 1997: pp. 283–286. http://www.aaai.org/Library/KDD/1997/kdd97-060.php.

[63] M.J. Zaki, K. Gouda, Fast vertical mining using diffsets, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, 2003: pp. 326–335. https://doi.org/10.1145/956750.956788.

[64] J. Liu, K. Wang, B.C.M. Fung, Direct discovery of high utility itemsets without candidate generation, in: IEEE International Conference on Data Mining, ICDM, 2012: pp. 984–989.

[65] S. Krishnamoorthy, Pruning strategies for mining high utility itemsets, Expert Syst Appl 42 (2015) 2371–2381.

[66] H. Ryang, U. Yun, K.H. Ryu, Fast algorithm for high utility pattern mining with the sum of item quantities, Intelligent Data Analysis 20 (2016) 395–415.

[67] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, in: ACM International Conference, ACM, 2012: pp. 55–64.

[68] Q.H. Duong, P. Fournier-Viger, H. Ramampiaro, K. Nørvåg, T.L. Dam, Efficient high utility itemset mining using buffered utility-lists, Applied Intelligence 48 (2018) 1859–1877.

[69] J.F. Qu, P. Fournier-Viger, M. Liu, B. Hang, C. Hu, Mining High Utility Itemsets Using Prefix Trees and Utility Vectors, IEEE Trans Knowl Data Eng 35 (2023) 10224–10236. https://doi.org/10.1109/TKDE.2023.3256126.

[70] V.S. Tseng, C.W. Wu, P. Fournier-Viger, P.S. Yu, Efficient algorithms for mining the concise and lossless representation of high utility itemsets, IEEE Trans Knowl Data Eng 27 (2015) 726–739.

[71] C. Lucchese, S. Orlando, R. Perego, DCI Closed: A Fast and Memory Efficient Algorithm to Mine Frequent Closed Itemsets, in: FIMI, 2004.

[72] C. Sivamathi, S. Vijayarani, Multi-level utility mining: Retrieval of high utility itemsets in a transaction database, Computers & Electrical Engineering 76 (2019) 268–282. https://doi.org/https://doi.org/10.1016/j.compeleceng.2019.04.003.

[73] N.T. Tung, L.T.T. Nguyen, T.D.D. Nguyen, B. Vo, An efficient method for mining multi-level high utility Itemsets, Applied Intelligence 52 (2022) 5475–5496.

[74] P. Fournier-Viger, Y. Yang, J.C.-W. Lin, J.M. Luna, S. Ventura, Y. Wang, J.C.-W. Lin, J.M. Luna, S. Ventura, Mining cross-level high utility itemsets, in: 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2020: p. 12.

[75] M. Nouioua, Y. Wang, P. Fournier-Viger, J.C.-W. Lin, J.M.-T. Wu, TKC: Mining top-k cross-level high utility itemsets, in: 2020 International Conference on Data Mining Workshops (ICDMW), 2020: pp. 673–682.

[76] N.T. Tung, L.T.T. Nguyen, T.D.D. Nguyen, P. Fourier-Viger, N.T. Nguyen, B. Vo, Efficient mining of cross-level high-utility itemsets in taxonomy quantitative databases, Inf Sci (N Y) 587 (2022) 41–62.

[77] P. Fournier-Viger, J.C.W. Lin, Q.H. Duong, T.L. Dam, FHM+: faster high-utility itemset mining using length upper-bound reduction, in: H. Fujita, M. Ali, A. Selamat, J. Sasaki, M. Kurematsu (Eds.), Trends in Applied Knowledge-Based Systems and Data Science, Springer International Publishing, Cham, 2016: pp. 115–127.

[78] H. Ryang, U. Yun, Top-k high utility pattern mining with effective threshold raising strategies, Knowl Based Syst 76 (2015) 109–126.

[79] V.S. Tseng, C.W. Wu, P. Fournier-Viger, P.S. Yu, Efficient algorithms for mining top-K high utility itemsets, IEEE Trans Knowl Data Eng 28 (2016) 54–67.

[80] S. Krishnamoorthy, Mining top-k high utility itemsets with effective threshold raising strategies, Expert Syst Appl 117 (2019) 148–165.

[81] T.M. Le, T.D.D. Nguyen, L.T.T. Nguyen, A. Kozierkiewicz, N.T. Tung, Extracting Top-k High Utility Patterns from Multi-level Transaction Databases, in: Intelligent Information and Database Systems, Springer Nature Singapore, 2023: pp. 296–306.

[82] V. V. Vu, M.T.H. Lam, T.T.M. Duong, L.T. Manh, T.T.T. Nguyen, L. V. Nguyen, U. Yun, V. Snasel, B. Vo, FTKHUIM: A Fast and Efficient Method for Mining Top-K High-Utility Itemsets, IEEE Access 11 (2023) 104789–104805. https://doi.org/10.1109/ACCESS.2023.3314984.

[83] S. Kannimuthu, K. Premalatha, Discovery of high utility itemsets using genetic algorithm with ranked mutation, Applied Artificial Intelligence 28 (2014) 337–359. https://doi.org/10.1080/08839514.2014.891839.

[84] J.C.-W. Lin, L. Yang, P. Fournier-Viger, J.M.-T. Wu, T.-P. Hong, L.S.-L. Wang, J. Zhan, Mining high-utility itemsets based on particle swarm optimization, Eng Appl Artif Intell 55 (2016) 320–330. https://doi.org/https://doi.org/10.1016/j.engappai.2016.07.006.

[85] W. Song, C. Huang, Mining high utility itemsets using bio-inspired algorithms: A diverse optimal value framework, IEEE Access 6 (2018) 19568–19582. https://doi.org/10.1109/ACCESS.2018.2819162.

[86] W. Song, J. Nan, Mining high utility itemsets using ant colony optimization, Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery (2021).

[87] N.T. Tung, T.D.D. Nguyen, L.T.T. Nguyen, Q.T. Tho, A. Mai, A Nature-Inspired Method to Mine Top-k Multi-Level High-Utility Itemsets, Cybern Syst (2023) 1–22. https://www.tandfonline.com/doi/full/10.1080/01969722.2023.2296244 (accessed January 6, 2024).

[88] P. Fournier-Viger, J.C.W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, H.T. Lam, The SPMF open-source data mining library version 2, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2016: pp. 36–40. https://doi.org/10.1007/978-3-319-46131-1_8.

[89] T. Hamrouni, S. Ben Yahia, E.M. Nguifo, Generalization of association rules through disjunction, Ann Math Artif Intell 59 (2010) 201–222.

[90] B. Vo, B. Le, Fast Algorithm for Mining Generalized Association Rules, Int J. of Database Theory 2 (2009) 19–21.

[91] H. Zhou, S. Mabu, K. Shimada, K. Hirasawa, MBFP generalized association rule mining and classification in traffic volume prediction, IEEJ Transactions on Electrical and Electronic Engineering 6 (2011) 457–467.

[92] P. Fournier-Viger, Y. Yang, J.C.-W. Lin, J.M. Luna, S. Ventura, Y. Wang, J.C.-W. Lin, J.M. Luna, S. Ventura, Mining Cross-Level High Utility Itemsets, in: 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2020: p. 12.

[93] M.J. Flynn, Some Computer Organizations and Their Effectiveness, IEEE Transactions on Computers C–21 (1972) 948–960.

[94] A. Duran, E. Ayguadé, R.M. Badia, J. Labarta, L. Martinell, X. Martorell, J. Planas, OmpSs: A proposal for programming heterogeneos multi-core architecture, Parallel Process Lett 21 (2011) 173–193. https://doi.org/https://doi.org/10.1142/S0129626411000151.

[95] K. Hasan, J. Antonio, S. Radhakrishnan, A New Composite CPU/Memory Model for Predicting Efficiency of Multi-core Processing, in: IEEE High Performance Computer Architecture (HPCA-14), 2014.