ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
Khoa Công Nghệ Thông Tin

KHAI THÁC TOP-K SỰ KIỆN ĐỒNG XUẤT HIỆN VỚI BITTABLE

CBHD: TS.NGUYỄN NGỌC THẢO

HVTH: NGUYỄN DUY CHINH

TP. HỒ CHÍ MINH, 12/2017

Mục tiêu

- Nghiên cứu các cách tiếp cận trong khai thác top-k sự kiện đồng xuất hiện trên cơ sở dữ liệu giao tác.

- Áp dụng và đề xuất cải tiến phương pháp nhằm nâng cao hơn nữa hiệu suất của quá trình khai thác.

Nội dung trình bày

1. Giới thiệu
2. Khai thác top-k sự kiện đồng xuất hiện
3. Phương pháp đề xuất
4. Kết quả thực nghiệm
5. Kết luận và hướng phát triển

1. Giới thiệu

Lĩnh vực ứng dụng:

- Phân tích thói quen mua sắm của khách hàng
- Hành vi sử dụng Web
- Hệ thống đưa ra gợi ý
- ….

Đây là bài toán mới lần đầu tiên được đề xuất năm 2015. Cho đến nay chưa có nhiều công trình nghiên cứu liên quan.

2. Khai thác top-k sự kiện đồng xuất hiện

Cho một cơ sở dữ liệu giao tác DB, một itemset P và k là số lượng sự kiện đồng xuất hiện với P phổ biến nhất. Bài toán khai thác top-k sự kiện đồng xuất hiện của P xác định k sự kiện xuất hiện phổ biến nhất cùng với P trong DB.

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f

Ví dụ cho $P = \{a, c\}$, những item đồng xuất hiện với P gồm có: b, d, e, f. Số lần đồng xuất hiện của b với P được ký hiệu là $CO(P, b)$. Khi đó, $CO(P, b) = 2$, $CO(P, d) = 1$, $CO(P, e) = 1$, $CO(P, f) = 2$. Nếu $k = 2$, thì kết quả top-2 sự kiện đồng xuất hiện đối với P là b, f.

(Ví dụ trên sẽ được áp dụng cho các slide minh họa sau)

2. Khai thác top-k sự kiện đồng xuất hiện Ứng Dụng

Bài toán có thể được ứng dụng trong các siêu thị hoặc các hệ thống dự đoán hành vi…

Bài toán được phát biểu lần đầu tiên bởi Zhi-Hong Deng, năm 2015, ông đã đề ra 3 thuật toán cơ bản là NT, NTI và PT

2. Khai thác top-k sự kiện đồng xuất hiện
A. Thuật toán NT

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f

Đầu vào: $P = \{a, c\}; k = 2$

Duyệt tất cả giao tác để đếm số lần đồng xuất hiện của các item

- $CO(P, b) = 2$
- $CO(P, d) = 1$
- $CO(P, e) = 1$
- $CO(P, f) = 2$

Xếp danh sách các item theo thứ tự giảm dần của $CO(P, i)$:

- $\{b, f, d, e\}$

Lấy top-2 cho ra kết quả: $\{b, f\}$

2. Khai thác top-k sự kiện đồng xuất hiện
B. Thuật toán NTI

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f

TID_set(a)= {1,2,4}

TID_set(b)= {1,2,5}

TID_set(c)={1,2,4,5}

TID_set(d)={2,5}

TID_set(e)= {3,4}

TID_set(f)= {2,3,4,5}

TID_set(g)= {3}

$P = \{a, c\}$

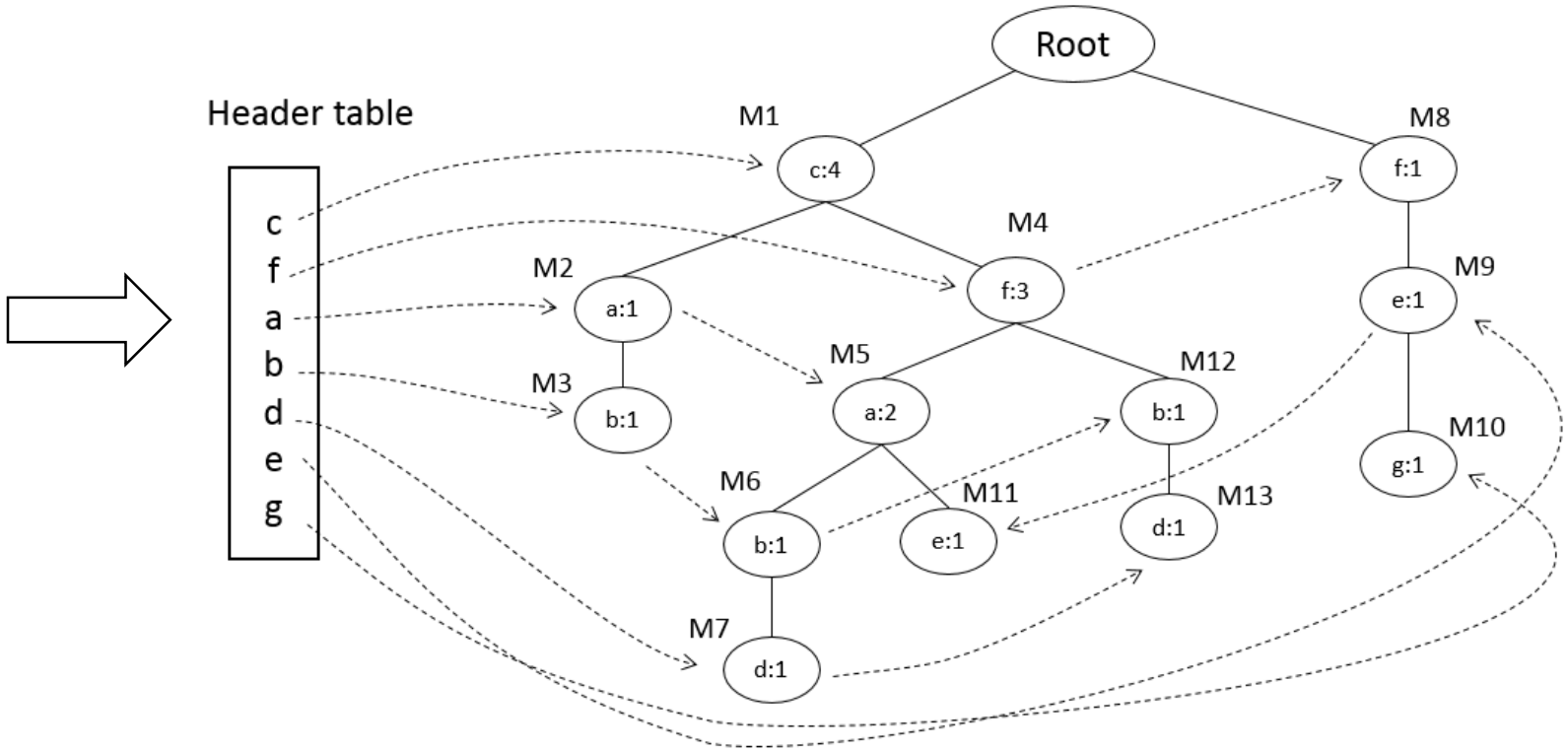
TID_set(a, c)= {1,2,4}

TID	Items
1	a, b, c
2	a, b, c, d, f
4	a, c, e, f

Chạy lại thuật toán NT

2. Khai thác top-k sự kiện đồng xuất hiện
C. Thuật toán PT

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f



2. Khai thác top-k sự kiện đồng xuất hiện C. Thuật toán PT(tt)

- Cây Pi-Tree có nhiều điểm tương đồng với cây FP-Tree nhưng có 2 điểm khác so với FP-Tree
  - Các đỉnh trong cây Pi-Tree không phân biệt sự kiện mà nó đại diện có phải là sự kiện phổ biến hay không
  - Các đỉnh trên cây Pi-Tree có liên kết đến đỉnh cha.
- Để tính số lần đồng xuất hiện của các item trải qua 3 giai đoạn:
  - Xác định các nhánh chứa mẫu truy vấn itemset P
  - Up count
  - Down count

3. Phương pháp đề xuất

Dùng cấu trúc BitTable để nén dữ liệu xử lý. Sử dụng các phép toán AND/OR trên kiểu dữ liệu bit để tính toán nhanh hơn.

1. Thuật toán BT(**B**it**T**able based algorithm)
2. Thuật toán BTI(**B**it**T**able base algorithm with **I**nverted list index)
3. Thuật toán BTIV(**B**it**T**able based algorithm with **I**nverted list index in **V**ertical)

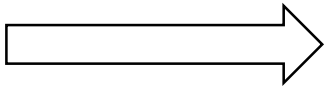
3. Phương pháp đề xuất

- Cấu trúc BitTable là cấu trúc dữ liệu đặc biệt được giới thiệu lần đầu tiên trong một công trình khoa học có tên **"BitTableFI: An efficient mining frequent itemsets algorithm"** [3] của hai tác giả Jie Dong và Min Han, năm 2006.

	a	b	c	d	e	f	g
1	1	1	1	0	0	0	0
2	1	1	1	1	0	1	0
3	0	0	0	0	1	1	1
4	1	0	1	0	0	1	0
5	0	1	1	1	0	1	0

3. Phương pháp đề xuất
A. Biểu diễn dữ liệu

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f



	a	b	c	d	e	f	g
1	*1*	*1*	*1*	*0*	*0*	*0*	*0*
2	*1*	*1*	*1*	*1*	*0*	*1*	*0*
3	*0*	*0*	*0*	*0*	*1*	*1*	*1*
4	*1*	*0*	*1*	*0*	*0*	*1*	*0*
5	*0*	*1*	*1*	*1*	*0*	*1*	*0*

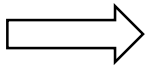
Biểu diễn BitTable theo chiều ngang

	1	2	3	4	5
a	*1*	*1*	*0*	*1*	*0*
b	*1*	*1*	*0*	*0*	*1*
c	*1*	*1*	*0*	*1*	*1*
d	*0*	*1*	*0*	*0*	*1*
e	*0*	*0*	*1*	*0*	*0*
f	*0*	*1*	*1*	*1*	*1*
g	*0*	*0*	*1*	*0*	*0*

Biểu diễn BitTable theo chiều dọc

3. Phương pháp đề xuất
B. Thuật toán BT

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f



	a	b	c	d	e	f	g
1	*1*	*1*	*1*	*0*	*0*	*0*	*0*
2	*1*	*1*	*1*	*1*	*0*	*1*	*0*
3	*0*	*0*	*0*	*0*	*1*	*1*	*1*
4	*1*	*0*	*1*	*0*	*0*	*1*	*0*
5	*0*	*1*	*1*	*1*	*0*	*1*	*0*

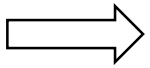
$AND\ 1010000 = 1010000$?

Kiểm tra $P = \{a, c\} \sqsubseteq T$?

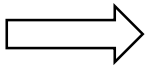
- Thuật toán BT tương tự NT, duyệt tất cả các bit để đếm số lần đồng xuất hiện của các Item. Dùng phép AND trên bit để xác định giao tác có chứa P hay không.

3. Phương pháp đề xuất
C. Thuật toán BTI

TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f



TID	Items
1	a, b, c
2	a, b, c, d, f
4	a, c, e, f

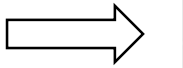


Chạy lại thuật toán BT

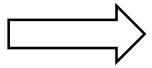
- Thuật toán BTI khắc phục nhược điểm của BTI bằng cách rút tỉa cơ sở dữ liệu đầu vào trước khi chuyển sang BitTable để xử lý.

3. Phương pháp đề xuất
D. Thuật toán BTIV

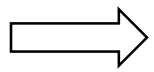
TID	Items
1	a, b, c
2	a, b, c, d, f
3	e, f, g
4	a, c, e, f
5	b, c, d, f



TID	Items
1	a, b, c
2	a, b, c, d, f
4	a, c, e, f



	1	2	4
a	*1*	*1*	*1*
b	*1*	*1*	*0*
c	*1*	*1*	*1*
d	*0*	*1*	*0*
e	*0*	*0*	*1*
f	*0*	*1*	*1*



	1	2	4	
b	*1*	*1*	*0*	$\rightarrow CO(P, b) = 2$
d	*0*	*1*	*0*	$\rightarrow CO(P, d) = 1$
e	*0*	*0*	*1*	$\rightarrow CO(P, e) = 1$
f	*0*	*1*	*1*	$\rightarrow CO(P, f) = 2$

- Thuật toán BTIV kết hợp ưu điểm của BTI với việc chuyển dữ liệu dạng văn bảng sang BitTable dạng dọc để xử lý.

4. Kết quả thực nghiệm
A. Tập dữ liệu

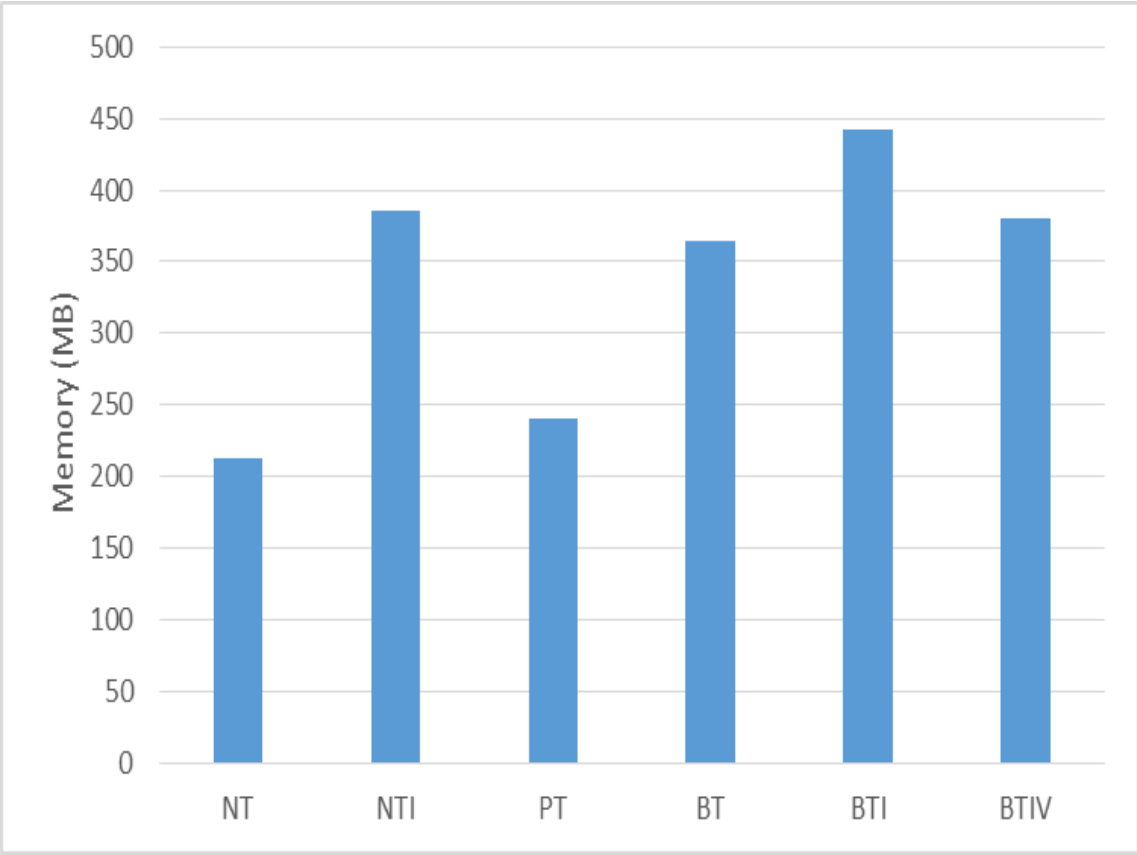
- Bộ dữ liệu thực (http://fimi.ua.ac.be/data)
  - So sánh với bài báo gốc

CSDL	Số Lượng Giao Tác	Số Sự Kiện Khác Nhau	Chiều Dài Trung Bình Mỗi Giao Tác	Tỉ Trọng
Connect	67,557	129	43	524
Accidents	340,183	468	34	727

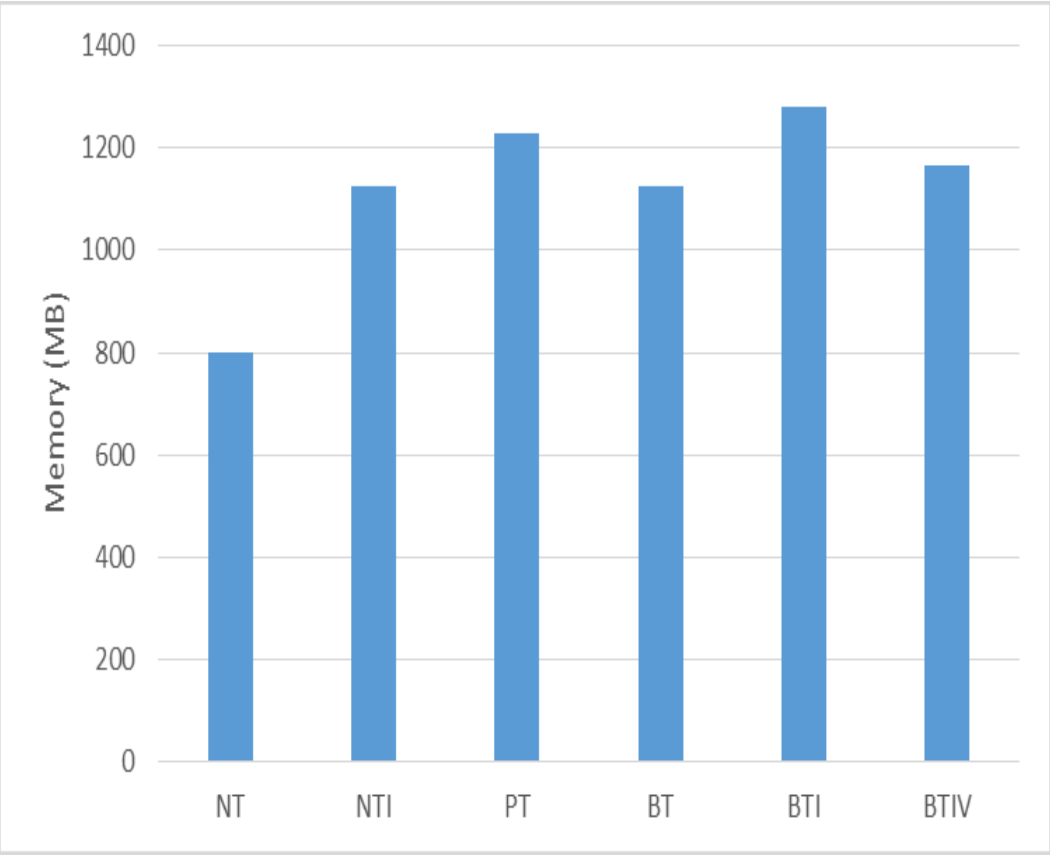
- Bộ dữ liệu tổng hợp (được phát sinh từ công cụ có tên là SPMF)
  - Đánh giáo thời gian thực thi và khả năng mở rộng

CSDL	Số Lượng Giao Tác	Số Sự Kiện Khác Nhau	Chiều Dài Trung Bình Mỗi Giao Tác	Tỉ Trọng
Syn_data1	1,000,000	198	20	5,050
Syn_data2	1,000,000	678	20	1,475

4. Kết quả thực nghiệm
B. Bộ nhớ sử dụng

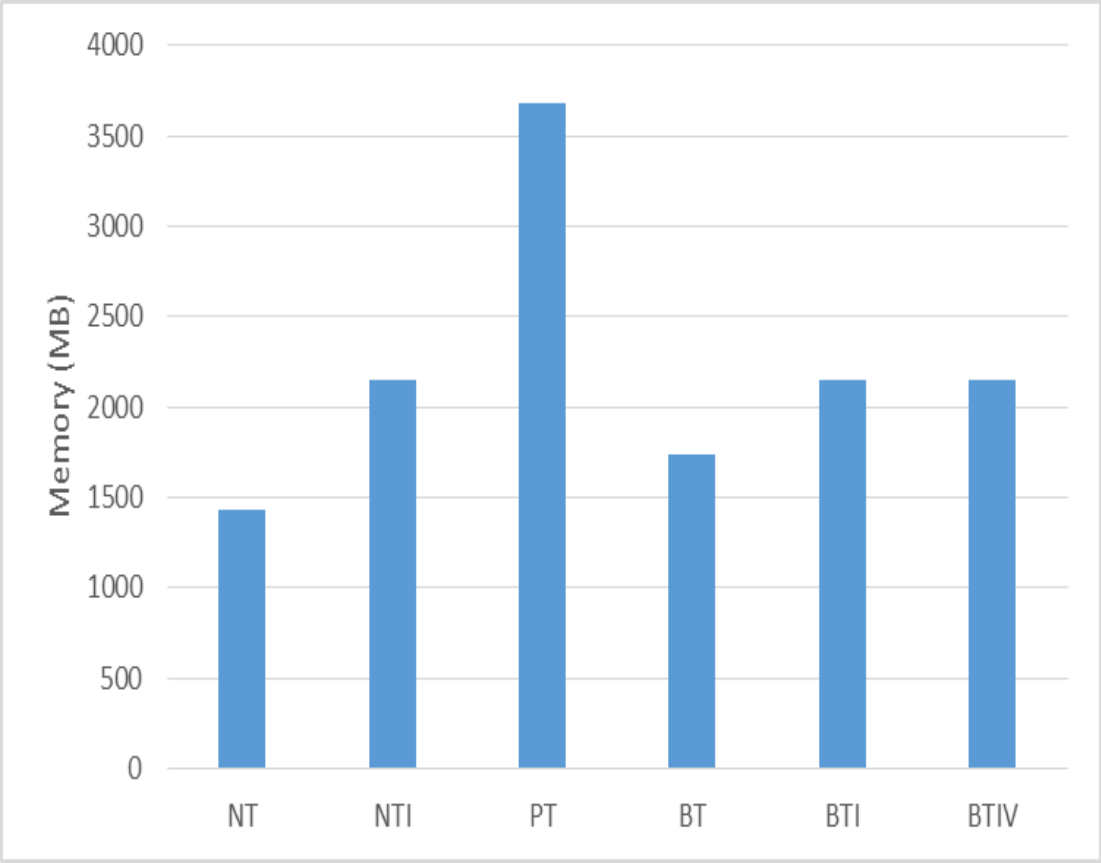


Tập Connect

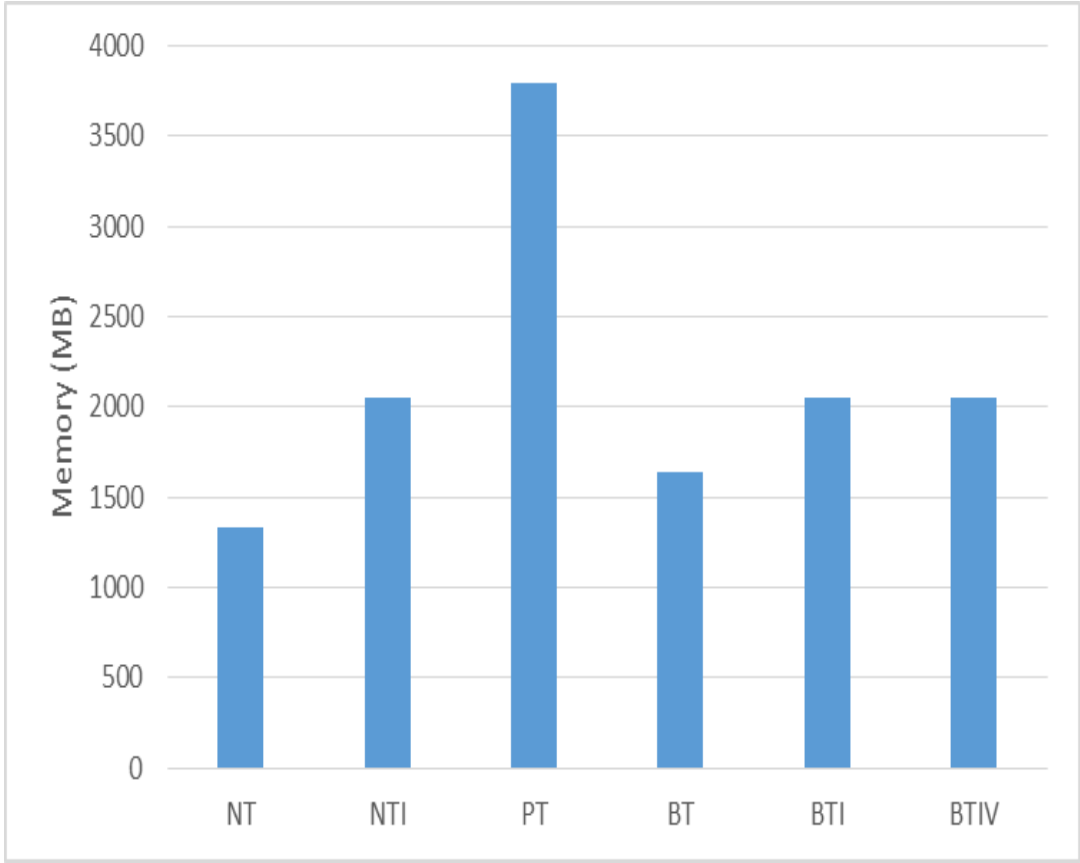


Tập Accidents

4. Kết quả thực nghiệm
B. Bộ nhớ sử dụng(tt)



Tập Syn_data1

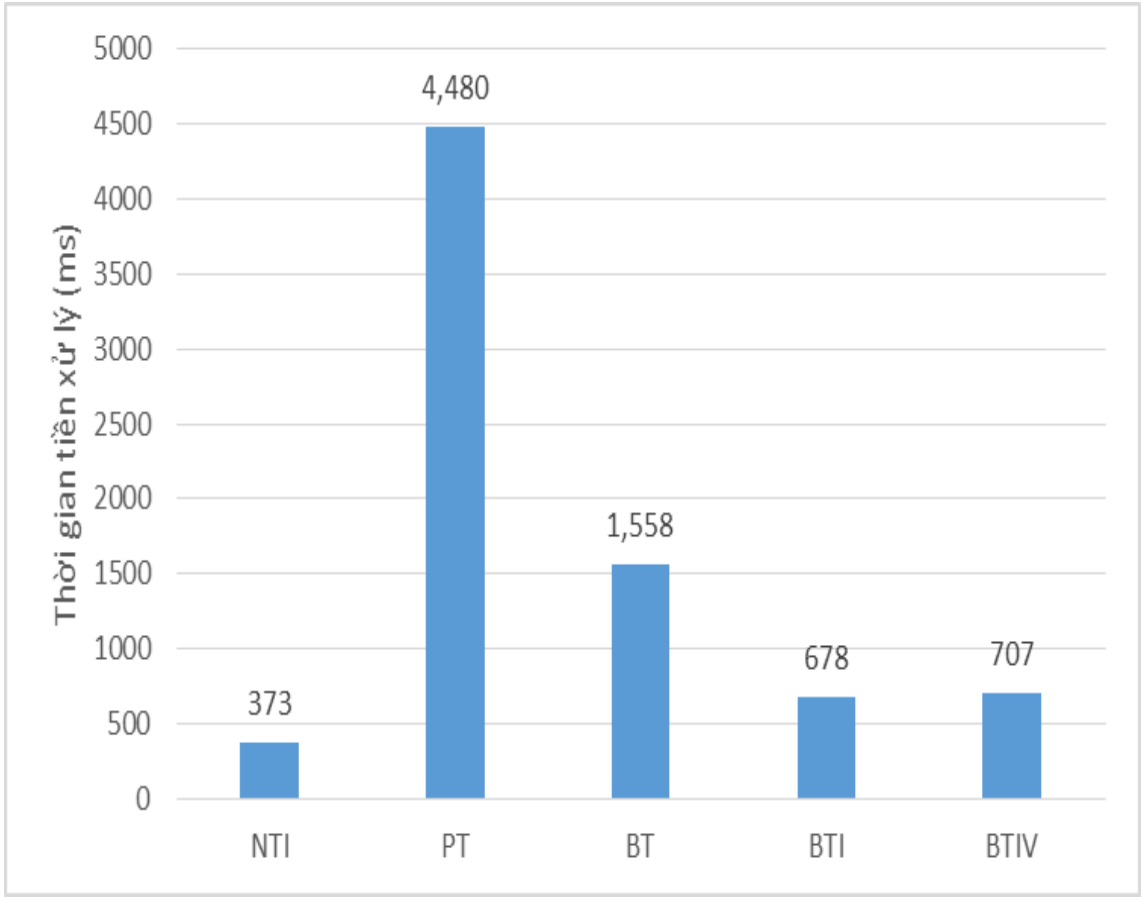


Tập Syn_data2

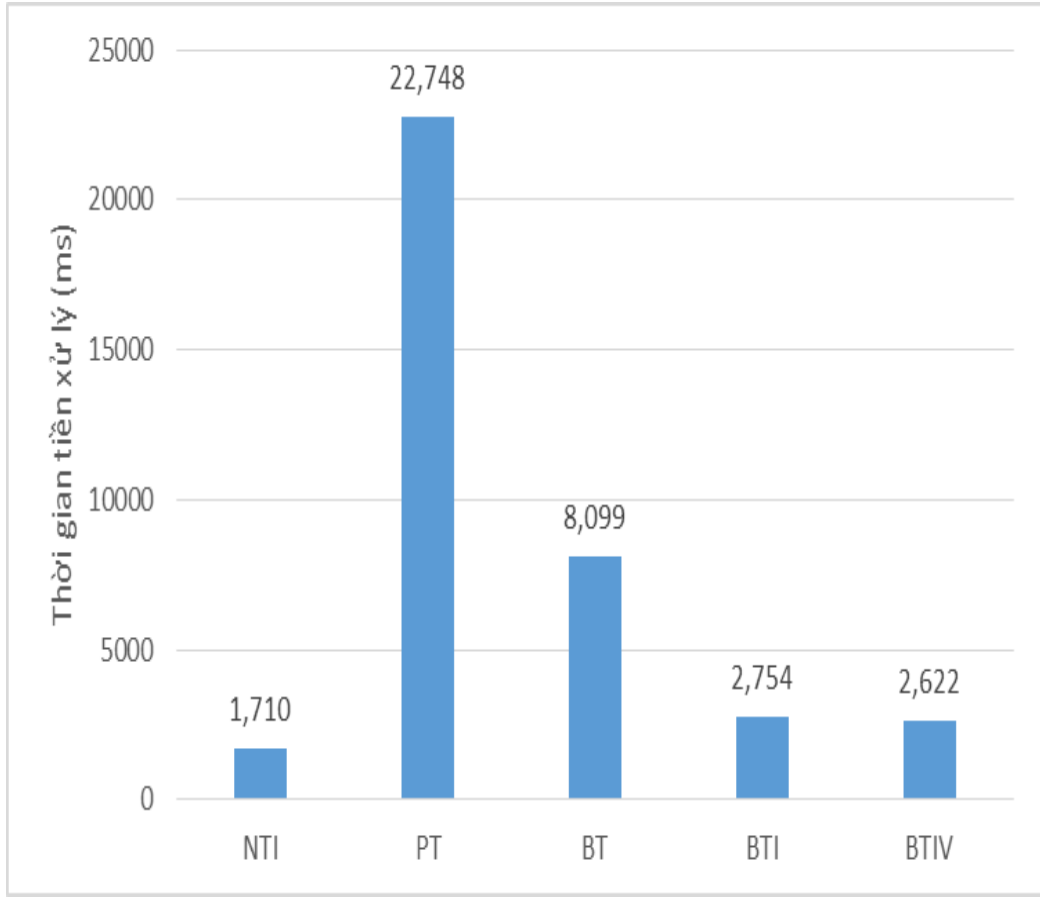
4. Kết quả thực nghiệm
B. Bộ nhớ sử dụng(tt)

Bộ nhớ sử dụng của ba thuật toán đề xuất không chênh lệch nhiều nếu so sánh với thuật toán NTI. Bộ nhớ sử dụng của thuật toán PT tăng lên đáng kể nếu so sánh trên tập Connect với những tập khác. Nguyên nhân là do thuật toán PT không rút tỉa cơ sở dữ liệu đầu vào trước khi xây dựng cây Pi-Tree và số đỉnh phát sinh trên tập Connect là 359,291, trên tập Accidents là 4,243,241, trên tập Syn_data1 là 17,021,247 và trên tập Syn_data2 là 18,152,498.

4. Kết quả thực nghiệm
C. Thời gian tiền xử lý

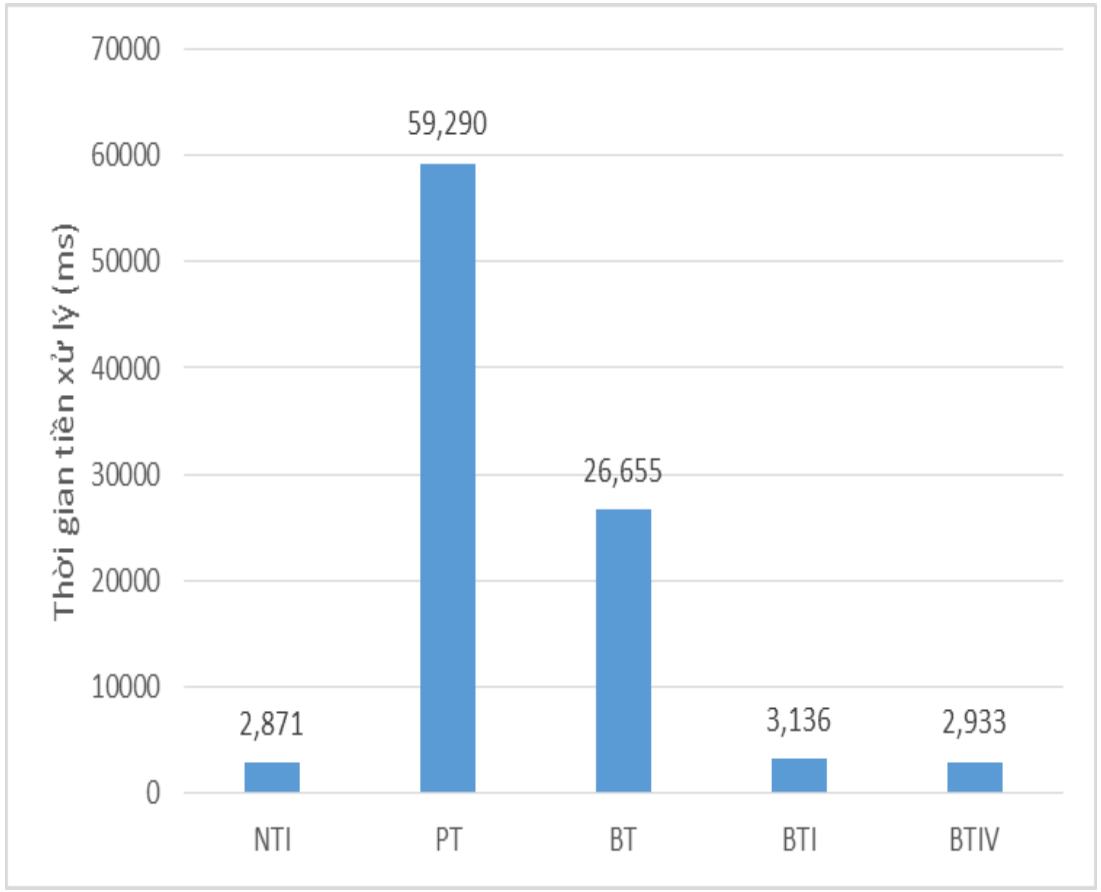


Tập Connect

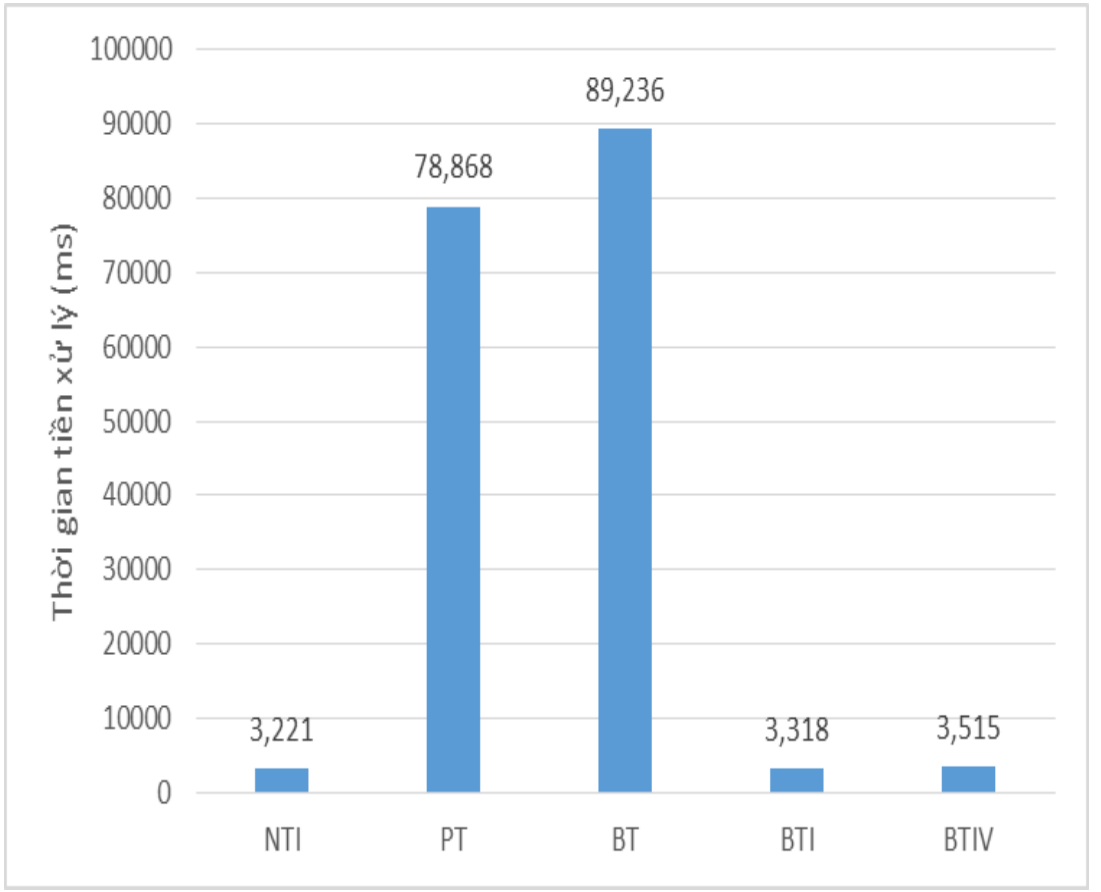


Tập Accidents

4. Kết quả thực nghiệm
C. Thời gian tiền xử lý(tt)



Tập Syn_data1

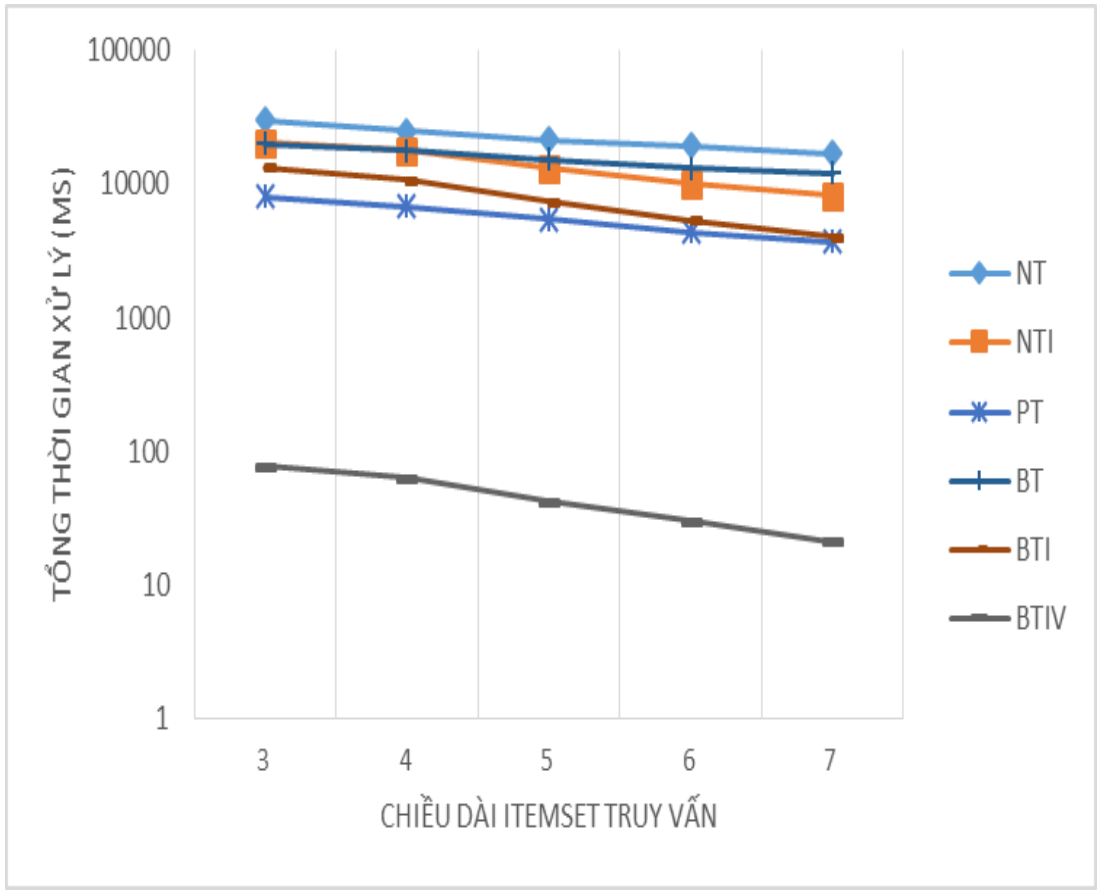


Tập Syn_data2

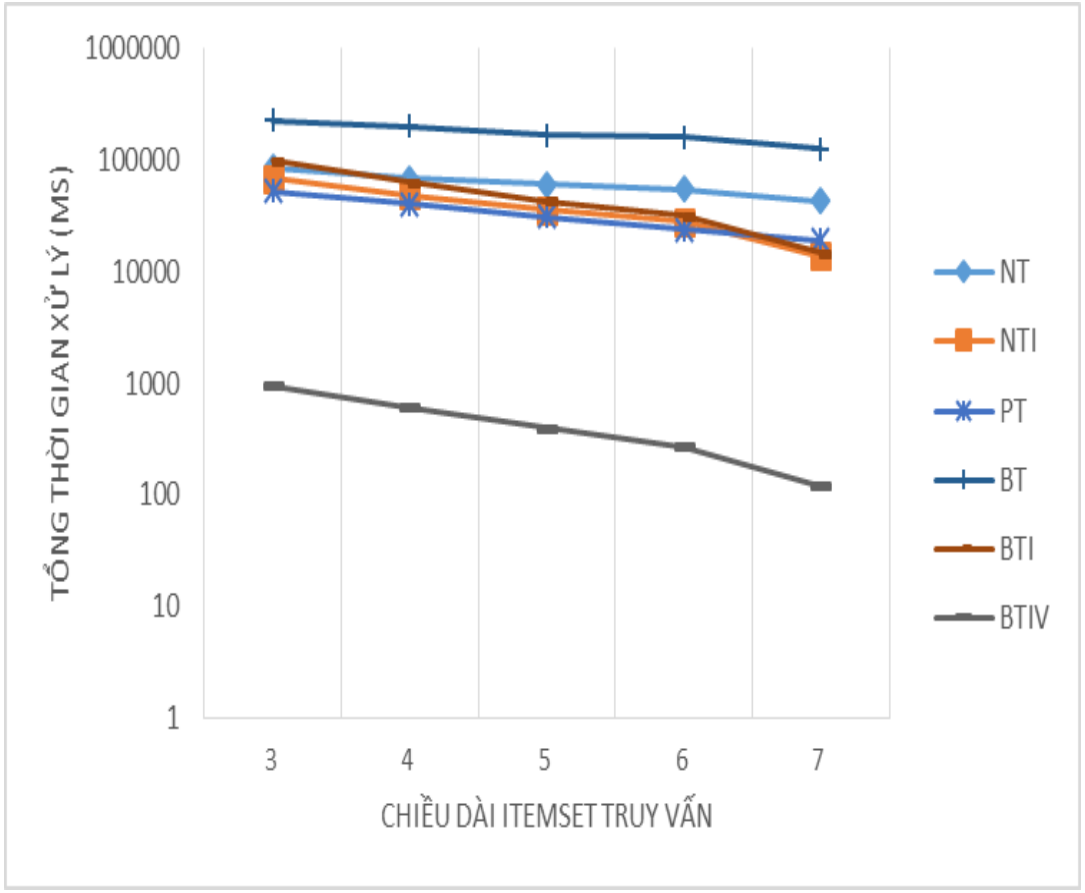
4. Kết quả thực nghiệm
C. Thời gian tiền xử lý(tt)

Thời gian tiền xử lý càng tăng khi tập dữ liệu đầu vào càng lớn. Thuật toán BT có thời gian tiền xử lý lâu hơn BTI và BTIV là vì thuật toán BT không rút tỉa cơ sở dữ liệu đầu vào trước khi chuyển qua BitTable.

4. Kết quả thực nghiệm
D. Thời gian xử lý

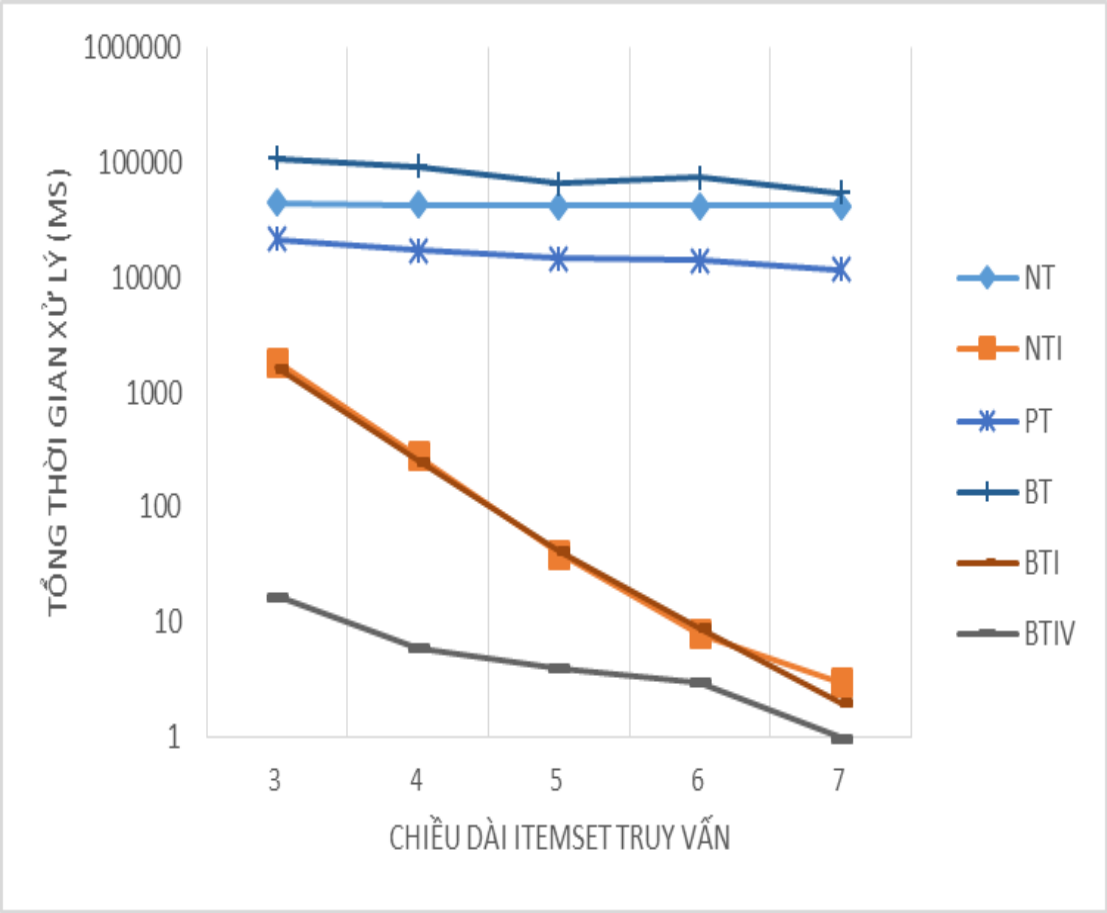


Tập Connect

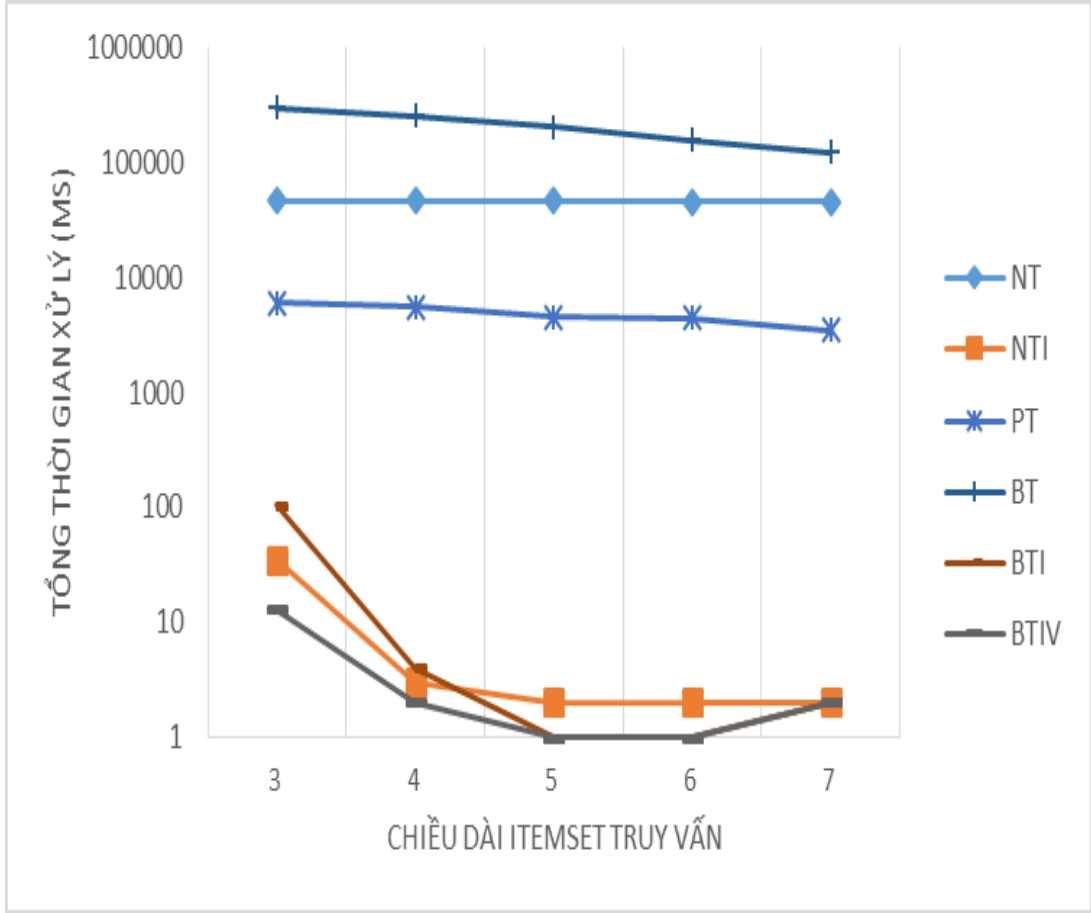


Tập Accidents

4. Kết quả thực nghiệm
D. Thời gian xử lý(tt)



Tập Syn_data1



Tập Syn_data2

4. Kết quả thực nghiệm
D. Thời gian xử lý(tt)

Thời gian xử lý của thuật toán BT và BTI không nhanh hơn thuật toán PT của tác giả bài báo gốc. Tuy nhiên, thuật toán $BTIV$ có thời gian xử lý nhanh hơn đáng kể so với những thuật toán khác.

Thời gian xử lý càng nhanh khi chiều dài itemset truy vấn càng dài. Điều này có thể được giải thích là vì có thể độ dài itemset truy vấn càng dài thì số giao dịch có chứa itemset truy vấn càng giảm nên thời gian xử lý cũng giảm theo.

5. Kết luận và hướng phát triển
A. Kết luận

- Nghiên cứu tổng quan về bài toán khai thác top-k sự kiện đồng xuất hiện trên cơ sở dữ liệu giao tác.
- Tiếp cận giải pháp biểu diễn dữ liệu bằng BitTable thông qua công trình gốc và những nghiên cứu liên quan.
- Đề xuất giải thuật cải tiến dựa trên phân tích ưu và khuyết điểm của các phương pháp trên.
- Trình bày kết quả thực nghiệm của các phương pháp đề xuất so với ba phương pháp NT,NTI và PT. Từ kết quả thực nghiệm cho thấy thuật toán đề xuất $BTIV$ cho kết quả tốt hơn so với ba thuật toán NT,NTI và PT.
- Hạn chế
  - Chưa thực nghiệm trên các bộ dữ liệu thực lớn.
  - Tài nguyên phần cứng còn hạn chế.

5. Kết luận và hướng phát triển
B. Hướng phát triển

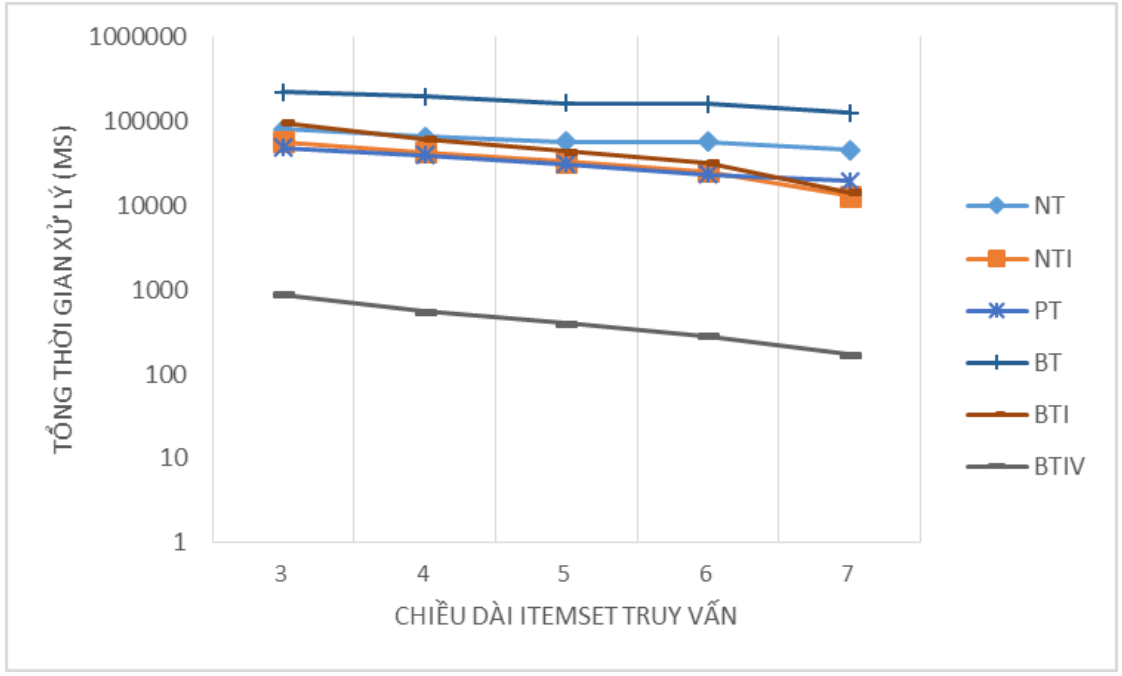
Thực nghiệm cho thấy thời gian xử lý với phương pháp đề xuất thấp. Điều này gợi mở khả năng áp dụng của phương pháp đề xuất trên dữ liệu quy mô lớn hơn. Tuy nhiên do hạn chế về tài nguyên xử lý và lưu trữ, đề tài xem đây là hướng phát triển tương lai.

Tài liệu tham khảo

1. J. C.-W. L. V. T. C. Z. B. L. Philippe Fournier-Viger, "A Survey of Itemset Mining," 2017.
2. Z.-H. Deng, "Mining Top-K Co-Occurrence Items," 2015
3. M. H. Jie Dong, "BitTableFI: An efficient mining frequent itemsets algorithm," Elsevier B.V, 2006.
4. B. Y. Z. X. Wei Song, "Index-BitTableFI: An improved algorithm for mining frequent itemsets," 2008.
5. T.-P. H. L. Bay Vo, "Dynamic bit vectors: An efficient approach for mining," 2011.
6. E. D. P. R. A.Saleem Raja, "CBT-fi: Compact BitTable Approach for Mining Frequent Itemsets," 2014.
7. M. J. Zaki, "Scalable algorithms for association mining," no. IEEE TKDE Journal, 2000.

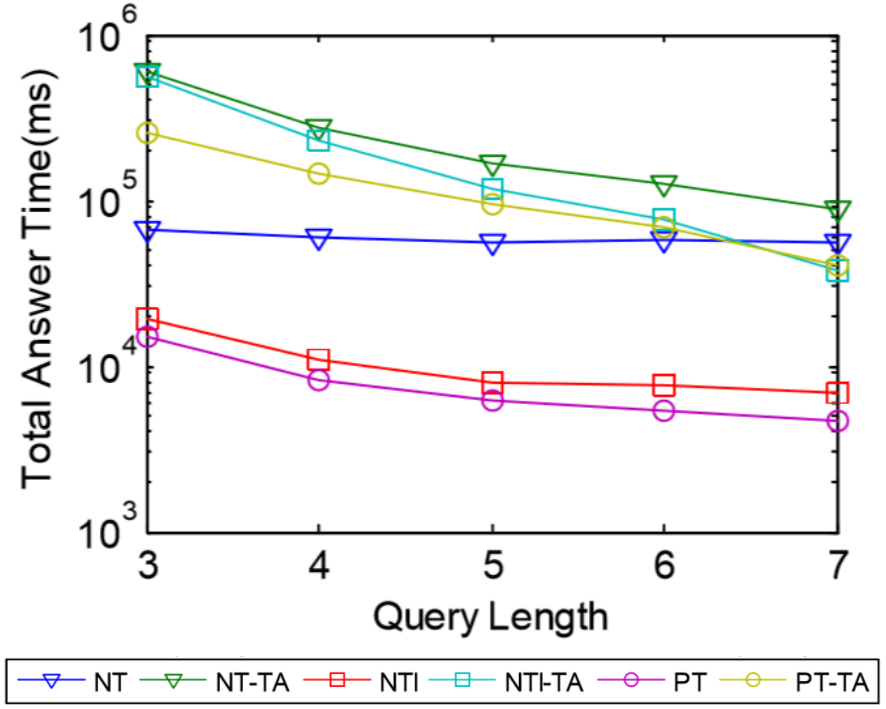
So sánh thời gian xử lý với kết quả tác giả đã công bố

Kết quả của đồ án



Win8 64bit, Intel Core i-4200U 1.6 GHz, Ram 8Gb

Kết quả công bố trong bài báo gốc



Win Server 2003 64bit, Intel processor 2GHz, Ram 16Gb

So sánh hai thuật toán BT và BTIV

- **Thuật toán BT**
- **Input**: cơ sở dữ liệu giao tác DB, itemset P, và ngưỡng k.
- **Output**: R_{TK}, Top-k sự kiện đồng xuất hiện của P
- 1: Chuyển DB thành BTh
- 2: Tạo P_{bit} từ P
- 3: **foreach** $T_{bit} \in BTh$ **do**
- 4: **if** $P_{bit}\ AND\ T_{bit} == P_{bit}$ **then**
- 5: **foreach** $i \in T_{bit}$ **do**
- 6: **if** $i == 1$ **then**
- **7:** $CO(item) \leftarrow getItem(i.index)$;
- 8: **if** $CO(item)\ is\ NULL$ **then**
- 9: $CO(item) \leftarrow 1$;
- 10: **else**
- 11: $CO(item) \leftarrow CO(item) + 1$;
- 12: $R_{TK} \leftarrow \{x|CO(x)$ là một trong top-k phần tử lớn nhất của $\{CO(item)\}\}$;
- 13: **return** R_{TK};

- **Thuật toán BTIV**
- **Input**: cơ sở dữ liệu giao tác DB, itemset P, và ngưỡng k.
- **Output**: R_{TK}, top-k sự kiện đồng xuất hiện của P.
- 1: Xây dựng DB_P từ DB;
- 2: Chuyển DB_P thành bảng BitTable theo chiều dọc BTv;
- 3: **foreach** $i \in P$ **do**
- 4: loại bỏ $i \in BTv$;
- 5: **foreach** $i \in BTv$ **do**
- 6: $CO(i) \leftarrow countBitArray(i.bitArray)$;
- 7: $R_{TK} \leftarrow \{x|CO(x)$ is one of top-k biggest elements of $\{CO(i)\}\}$;
- 8: **return** R_{TK};

Thuật toán NT gốc

- **Input**: cơ sở dữ liệu giao tác DB, itemset P, và ngưỡng k.
- **Output**: R_{TK}, Top-k sự kiện đồng xuất hiện của P
- 1: **foreach** $T \in DB$ **do**
- 2: **if** $P \subseteq T$ **then**
- 3: **foreach** $i \in T - P$ **do**
- 4: **if** i chưa có trong danh sách **then**
- 5: $CO(i) \leftarrow 1$;
- 6: **else**
- 7: $CO(i) \leftarrow CO(i) + 1$;
- 8: $R_{TK} \leftarrow \{x | CO(x)$ là một trong top-k phần tử lớn nhất của $\{CO(i)\}\}$;
- 9: **return** R_{TK};