An Efficient Algorithm for Mining Frequent Sequences by a New Strategy without Support Counting

Ding-Ying Chiu, Yi-Hung Wu, Arbee L. P. Chen

National Tsing Hua University

Presented by: Dr. Yi-Hung Wu
Date: 2004/3/30

Outline

- Introduction
  - Three strategies – the related works
- The DISC strategy
- The DISC-all Algorithm
- Performance Evaluation
- Conclusion

Introduction

- **Motivation**
  - Problem of mining sequential patterns
    - **Candidate generation + support counting**
  - Three strategies used in the related works
  - Can we mine all the frequent sequences without counting the supports of non-frequent sequences?
- **Goal**
  - A new strategy to meet this demand
  - An efficient algorithm combining all the strategies

Introduction

❑ An Example

- Support count: number of customer sequences
- Minimum support count δ (=2)
- All the frequent k-sequences
  - **Frequent 3-sequences**

    (a, g)(b), (a, g)(h),

    (a, g)(f), (a)(b)(h),

    (a)(b)(f), (a)(b, f),

    …

CID	Customer Sequences
1	(a, e, g)(b)(h)(f)(c)(b, f)
2	(b)(d, f)(e)
3	(b, f, g)
4	(f)(a, g)(b, f, h)(b, f)

Introduction

❑ An Example

- Support count: number of customer sequences
- Minimum support count δ (=2)
- All the frequent k-sequences
  - **Frequent 3-sequences**

    (a, g)(b), (a, g)(h),

    (a, g)(f), (a)(b)(h),

    (a)(b)(f), (a)(b, f),

    …

CID	Customer Sequences
1	(a, e, g)(b)(h)(f)(c)(b, f)
2	(b)(d, f)(e)
3	(b, f, g)
4	(f)(a, g)(b, f, h)(b, f)

Support count of <(b, f)>=2

Introduction

- **Strategy 1: Candidate Sequence Pruning**
  - Prune the candidates as early as possible
    - **Anti-monotone property**
      - *If (a)(b) is not frequent, (a)(b)(c), (a)(b, c), … are not, either.*
  - Reduce the processing costs and storage overheads for support counting
  - Has been used in all the related works
    - **GSP, SPADE, PrefixSpan, SPAM, …**

Introduction

- **Strategy 2: Database Partitioning**
  - PrefixSpan
    - **Projected database: the customer sequences having the same prefix**
  - SPADE/SPAM
    - **ID-list/bitmap: the customer sequences and the positions where a sequence appears**
  - Eliminate the unnecessary decompositions of customer sequences
    - **But add extra costs on projections or merging**

Introduction

- **Strategy 3: Customer Sequence Reducing**
  - Shorten the customer sequences
  - PrefixSpan removes items during the projections
    - **Pseudo uses pointers to simulate the item removal**
  - Reduce the processing and storage costs for decomposing the customer sequences

CID	**Customer Sequences**
1	**(a, e, g)(b)(h)(f)(c)(b, f)**
2	**(b)(d, f)(e)**
3	**(b, f, g)**
4	**(f)(a, g)(b, f, h)(b, f)**

Introduction

❑ Strategy 3: Customer Sequence Reducing

- Shorten the customer sequences
- PrefixSpan removes items during the projections
  - **Pseudo uses pointers to simulate the item removal**
- Reduce the processing and storage costs for decomposing the customer sequences

CID	Customer Sequences
1	(a, e, g)(b)(h)(f)(c)(b, f)
2	(b)(d, f)(e)
3	(b, f, g)
4	(f)(a, g)(b, f, h)(b, f)

prefix
(a)(b)

CID	Customer Sequences
1	(h)(f)(c)(b, f)
4	(_, f, h)(b, f)

Introduction

- **Strategy 4: DIrect Sequence Comparison (DISC)**
  - Recognize all the frequent k-sequences without counting the supports of non-frequent k-sequences
  - Reduce the costs for support counting and decomposing the customer sequences

Introduction

- **Strategy 4: DIrect Sequence Comparison (DISC)**
  - Recognize all the frequent k-sequences without counting the supports of non-frequent k-sequences
  - Reduce the costs for support counting and decomposing the customer sequences

Strategy \ Algorithm	GSP	SPADE	SPAM	PrefixSpan	DISC-all
Candidate Sequence Pruning	√	√	√	√	√
Database Partitioning		√	√	√	√
Customer Sequence Reducing				√	√
DISC					√

The DISC Strategy

- **The Original Database $\Rightarrow$ K-Sorted Database**
  - A method to compare sequences

    $A=(a_1b_1c_1)(b_2d_2)$
    $B=(a_1c_1d_1)(a_2)$
    $C=(a_1b_1)(c_2d_2)$

    - **Deferential point**
      - *Leftmost different item or transaction number*
      - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
    - **Comparative order: the alphanumeric order on DP**
      - *A<C<B (compare the items first)*
  - K-minimum subsequence $\Rightarrow$ k-minimum order
    - **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
    - **2-minimum order: $B<_2C=_2A$**

The DISC Strategy

- **The Original Database $\Rightarrow$ K-Sorted Database**
  - A method to compare sequences

$A=(a_1b_1c_1)(b_2d_2)$
$B=(a_1c_1d_1)(a_2)$
$C=(a_1b_1)(c_2d_2)$

    - **Deferential point**
      - *Leftmost different item or transaction number*
      - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
    - **Comparative order: the alphanumeric order on DP**
      - *A<C<B (compare the items first)*
  - K-minimum subsequence $\Rightarrow$ k-minimum order
    - **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
    - **2-minimum order: $B<_2C=_2A$**

The DISC Strategy

❑ The Original Database ⇒ K-Sorted Database

$A=(a_1 b_1 c_1)(b_2 d_2)$
$B=(a_1 c_1 d_1)(a_2)$
$C=(a_1 b_1)(c_2 d_2)$

➢ A method to compare sequences

- **Deferential point**
  - *Leftmost different item or transaction number*
  - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
- **Comparative order: the alphanumeric order on DP**
  - *A<C<B (compare the items first)*

➢ K-minimum subsequence ⇒ k-minimum order

- **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
- **2-minimum order: $B<_2C=_2A$**

The DISC Strategy

❑ The Original Database ⇒ K-Sorted Database

$A=(a_1b_1c_1)(b_2d_2)$
$B=(a_1c_1d_1)(a_2)$
$C=(a_1b_1)(c_2d_2)$

- A method to compare sequences
  - **Deferential point**
    - *Leftmost different item or transaction number*
    - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
  - **Comparative order: the alphanumeric order on DP**
    - *A<C<B (compare the items first)*
- K-minimum subsequence ⇒ k-minimum order
  - **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
  - **2-minimum order: $B<_2C=_2A$**

The DISC Strategy

❑ **The Original Database $\Rightarrow$ K-Sorted Database**

$A=(a_1b_1c_1)(b_2d_2)$
$B=(a_1c_1d_1)(a_2)$
$C=(a_1b_1)(c_2d_2)$

➢ A method to compare sequences

- **Deferential point**
  - *Leftmost different item or transaction number*
  - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
- **Comparative order: the alphanumeric order on DP**
  - *A<C<B (compare the items first)*

➢ K-minimum subsequence $\Rightarrow$ k-minimum order

- **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
- **2-minimum order: $B<_2C=_2A$**

The DISC Strategy

- **The Original Database $\Rightarrow$ K-Sorted Database**
  - A method to compare sequences

    $A=(a_1 b_1 c_1)(b_2 d_2)$
    $B=(a_1 c_1 d_1)(a_2)$
    $C=(a_1 b_1)(c_2 d_2)$

    - **Deferential point**
      - *Leftmost different item or transaction number*
      - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
    - **Comparative order: the alphanumeric order on DP**
      - *A<C<B (compare the items first)*
  - K-minimum subsequence $\Rightarrow$ k-minimum order
    - **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
    - **2-minimum order: $B <_2 C =_2 A$**

The DISC Strategy

❑ The Original Database ⇒ K-Sorted Database

➢ A method to compare sequences

- **Deferential point**
  - *Leftmost different item or transaction number*
  - *DP(A,B)=2, DP(A,C)=3, DP(B,C)=2*
- **Comparative order: the alphanumeric order on DP**
  - $A<C<B$ *(compare the items first)*

$A=(a_1b_1c_1)(b_2d_2)$
$B=(a_1c_1d_1)(a_2)$
$C=(a_1b_1)(c_2d_2)$

(a, b)
(a)(b)
(a, c)
(a)(d)
(b)(b)
…

➢ K-minimum subsequence ⇒ k-minimum order

- **K=2: KMS_A=(a, b), KMS_B=(a)(a), KMS_C=(a, b)**
- **2-minimum order: $B<_2C=_2A$**

The DISC Strategy

- **K-sorted Database ⇒ Compare α_1 and α_δ**
  - Equality: if $\delta=2$ ⇒ (a)(b)(b) is frequent
    - **For CID=1 and 4, find new KMS > (a)(b)(b)**

α_1

α_δ

CID	3-minimum Subsequence	Customer Sequence
1	(a)(b)(b)	(a, e, g)(b)(h)(f)(c)(b, f)
4	(a)(b)(b)	(f)(a, g)(b, f, h)(b, f)
2	(b)(d)(e)	(b)(d, f)(e)
3	(b, f, g)	(b, f, g)

The DISC Strategy

- **K-sorted Database $\Rightarrow$ Compare α_1 and α_δ**
  - Equality: if δ=2 $\Rightarrow$ (a)(b)(b) is frequent
    - **For CID=1 and 4, find new KMS > (a)(b)(b)**
  - Inequality: if δ=3 $\Rightarrow$ (a)(b)(b) is not frequent
    - **For CID=1 and 4, find new KMS $\geq$ (b)(d)(e)**
    - **Avoid unnecessary decomposition of KMS < (b)(d)(e)**

α_1

α_δ

CID	**3-minimum Subsequence**	**Customer Sequence**
1	**(a)(b)(b)**	**(a, e, g)(b)(h)(f)(c)(b, f)**
4	**(a)(b)(b)**	**(f)(a, g)(b, f, h)(b, f)**
2	**(b)(d)(e)**	**(b)(d, f)(e)**
3	**(b, f, g)**	**(b, f, g)**

The DISC Strategy

- **K-sorted Database $\Rightarrow$ Compare α_1 and α_δ**
  - Equality: if δ=2 $\Rightarrow$ (a)(b)(b) is frequent
    - **For CID=1 and 4, find new KMS > (a)(b)(b)**
  - Inequality: if δ=3 $\Rightarrow$ (a)(b)(b) is not frequent
    - **For CID=1 and 4, find new KMS ≥ (b)(d)(e)**
    - **Avoid unnecessary decomposition of KMS < (b)(d)(e)**

Conditional KMS

α_1

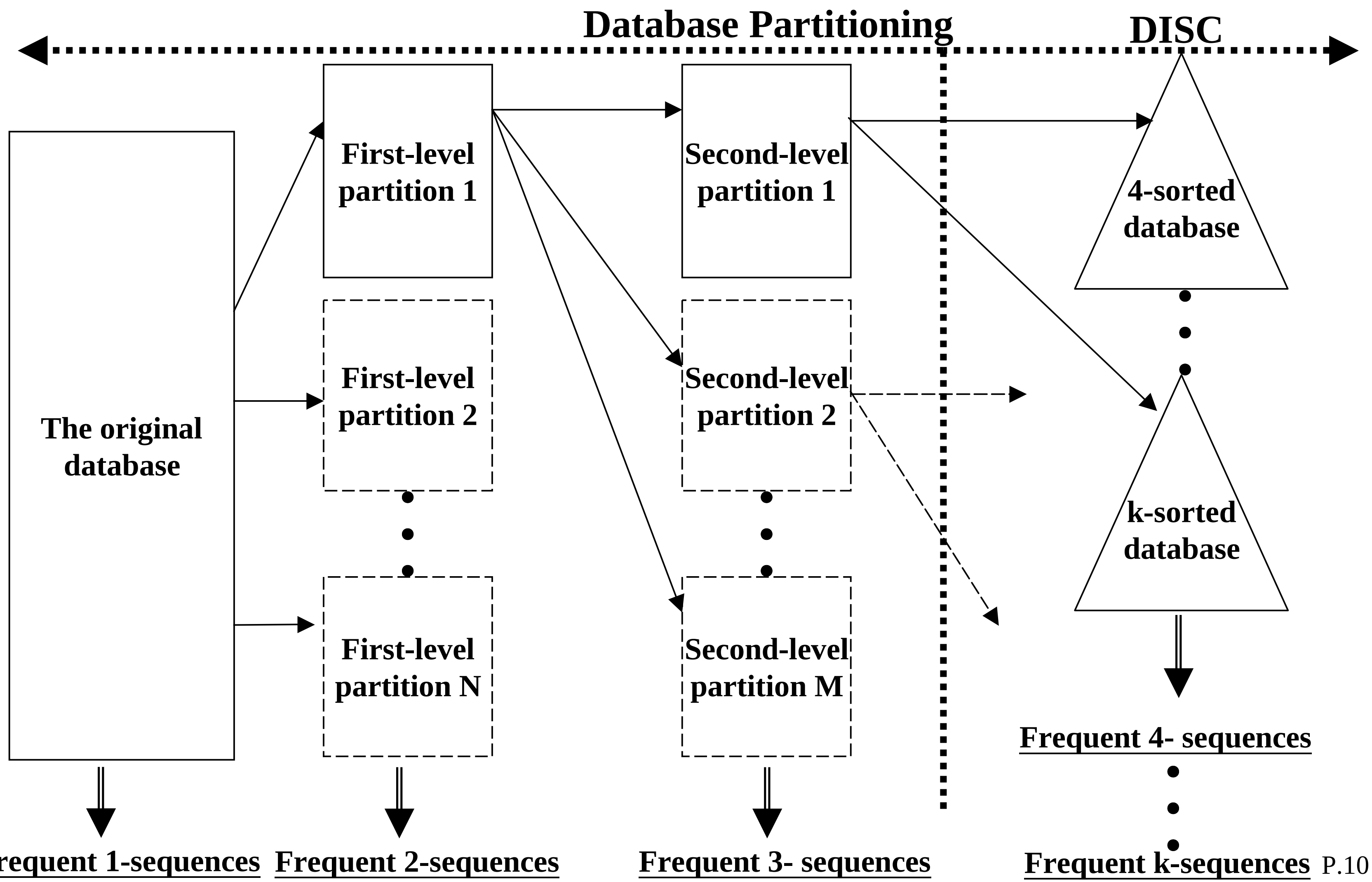
α_δ

CID	3-minimum Subsequence	Customer Sequence
1	(a)(b)(b)	(a, e, g)(b)(h)(f)(c)(b, f)
4	(a)(b)(b)	(f)(a, g)(b, f, h)(b, f)
2	(b)(d)(e)	(b)(d, f)(e)
3	(b, f, g)	(b, f, g)

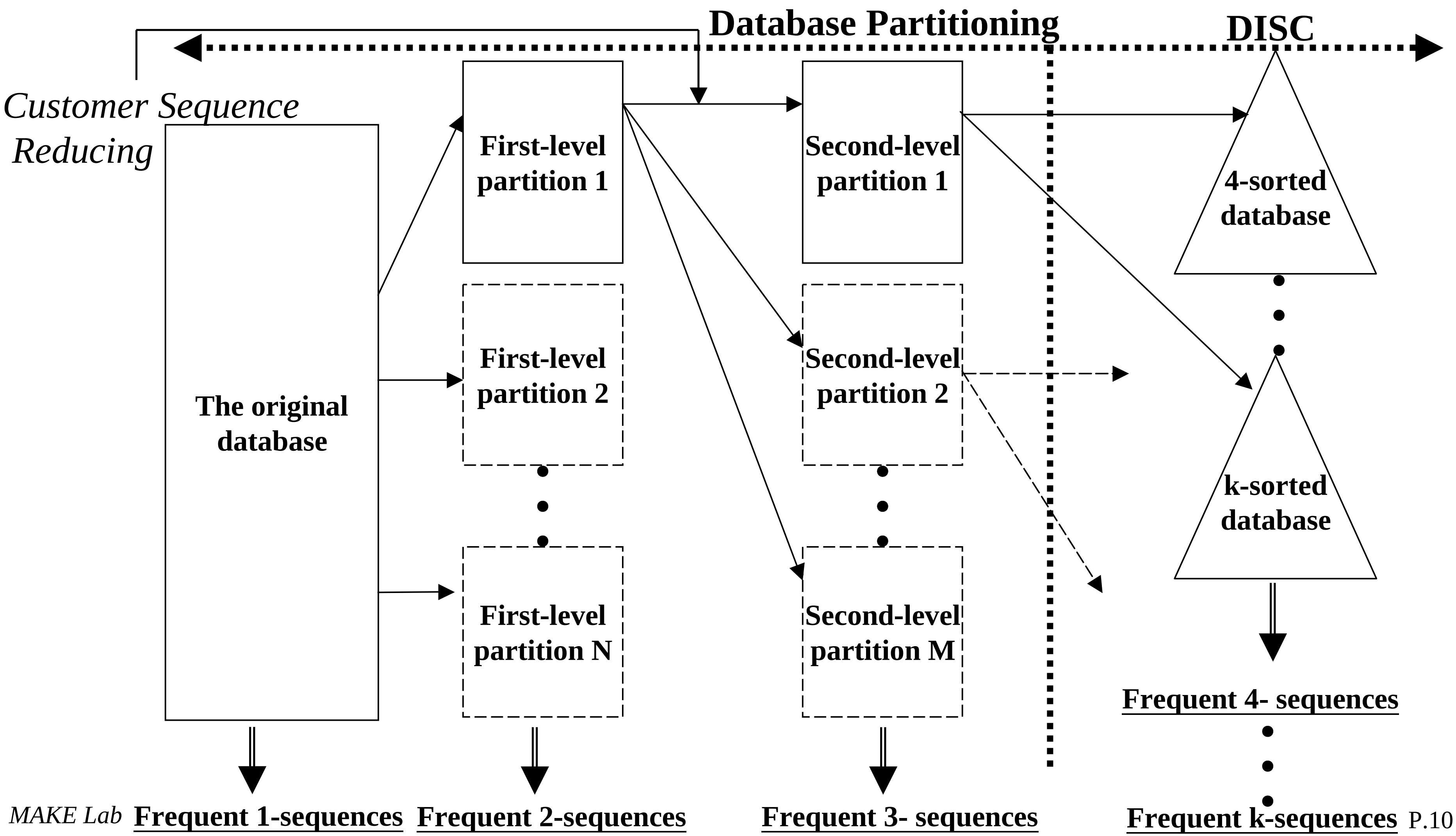
The DISC Strategy

- **Advantages**
  - Only the frequent sequences are counted
  - The costs of sequence decomposition are reduced
  - The frequent k-sequences can be discovered directly without following the bottom-up approach
- **Core Techniques**
  - Algorithms to quickly find KMS and CKMS
    - **Apriori-KMS/Apriori-CKMS**
  - Data structure to maintain the k-sorted database

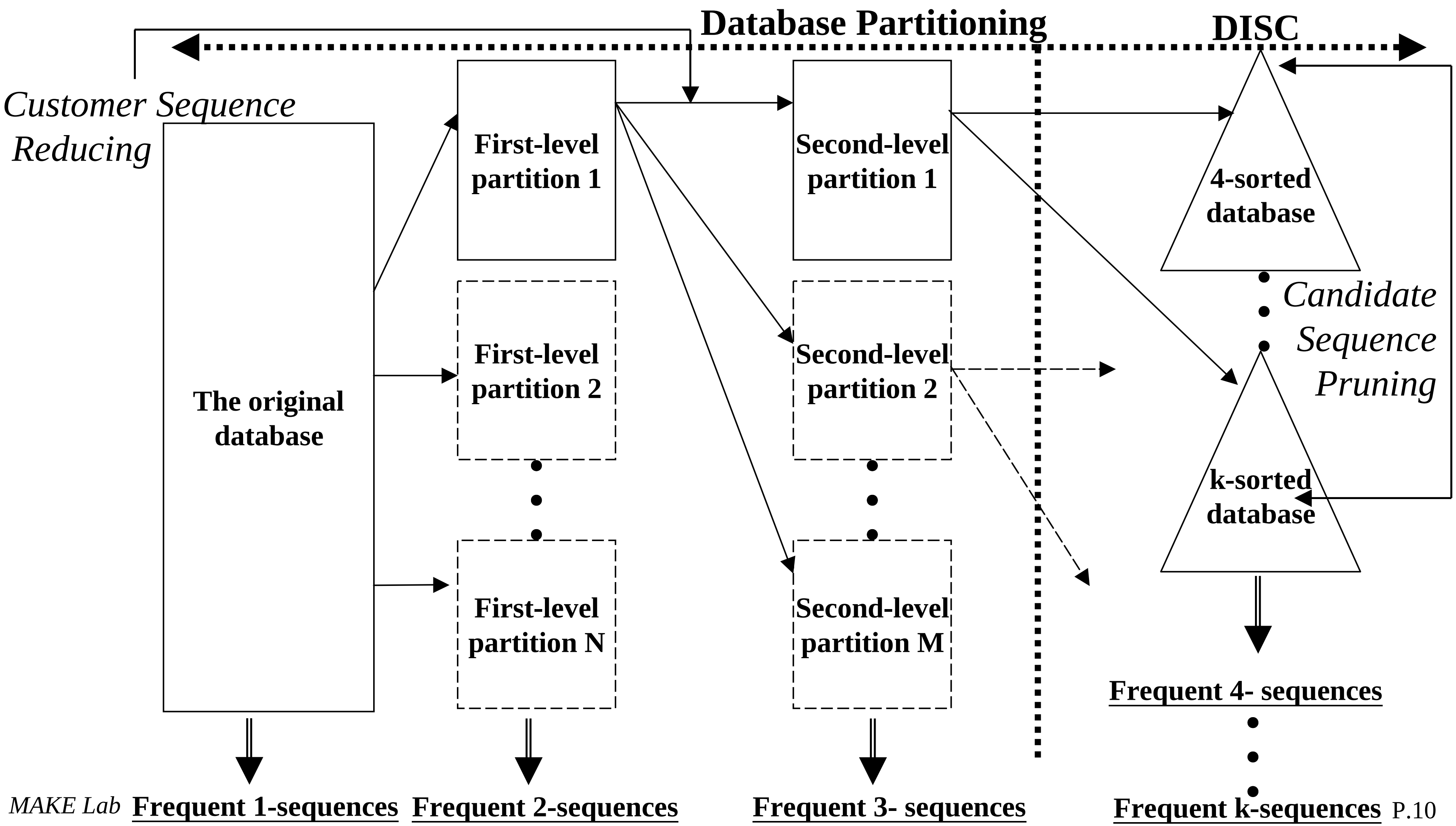
The DISC-all Algorithm



The DISC-all Algorithm



The DISC-all Algorithm



The DISC-all Algorithm

❑ Strategy 2: Multi-level Partitioning

CID	Customer Sequences	Initial Partitions	After <(a)>-partition
1	(a, d)(d)(a, g, h)(c)	<(a)>-partition	<(c)>-partition
2	(b)(a)(f)(a, c, e, g)	<(a)>-partition	<(b)>-partition
3	(a, f, g)(a, e, g, h)(c, g, h)	<(a)>-partition	<(c)>-partition
4	(f)(a, c, f)(a, c, e, g, h)	<(a)>-partition	<(c)>-partition
5	(a, g)	<(a)>-partition	<(g)>-partition
6	(a, f)(a, e, g, h)	<(a)>-partition	<(e)>-partition
7	(a, b, g)(a, e, g)(g, h)	<(a)>-partition	<(b)>-partition
8	(b, f)(b, e)(e, f, h)	<(b)>-partition	<(b)>-partition
9	(d, f)(d, f, g, h)	<(d)>-partition	<(d)>-partition
10	(b, f, g)(c, e, h)	<(b)>-partition	<(b)>-partition
11	(e, g)(f)(e, f)	<(e)>-partition	<(e)>-partition

The DISC-all Algorithm

❑ Strategy 2: Multi-level Partitioning

CID	Customer Sequences	Initial Partitions	After <(a)>-partition
1	(a, d)(d)(a, g, h)(c)	<(a)>-partition	<(c)>-partition
2	(b)(a)(f)(a, c, e, g)	<(a)>-partition	<(b)>-partition
3	(a, f, g)(a, e, g, h)(c, g, h)	<(a)>-partition	<(c)>-partition
4	(f)(a, c, f)(a, c, e, g, h)	<(a)>-partition	<(c)>-partition
5	(a, g)	<(a)>-partition	<(g)>-partition
6	(a, f)(a, e, g, h)	<(a)>-partition	<(e)>-partition
7	(a, b, g)(a, e, g)(g, h)	<(a)>-partition	<(b)>-partition
8	(b, f)(b, e)(e, f, h)	<(b)>-partition	<(b)>-partition
9	(d, f)(d, f, g, h)	<(d)>-partition	<(d)>-partition
10	(b, f, g)(c, e, h)	<(b)>-partition	<(b)>-partition
11	(e, g)(f)(e, f)	<(e)>-partition	<(e)>-partition

The DISC-all Algorithm

❑ Strategy 2: Multi-level Partitioning

CID	Customer Sequences	Initial Partitions	After <(a)>-partition
1	(a, d)(d)(a, g, h)(c)	<(a)>-partition	<(c)>-partition
2	(b)(a)(f)(a, c, e, g)	<(a)>-partition	<(b)>-partition
3	(a, f, g)(a, e, g, h)(c, g, h)	<(a)>-partition	<(c)>-partition
4	(f)(a, c, f)(a, c, e, g, h)	<(a)>-partition	<(c)>-partition
5	(a, g)	<(a)>-partition	<(g)>-partition
6	(a, f)(a, e, g, h)	<(a)>-partition	<(e)>-partition
7	(a, b, g)(a, e, g)(g, h)	<(a)>-partition	<(b)>-partition
8	(b, f)(b, e)(e, f, h)	<(b)>-partition	<(b)>-partition
9	(d, f)(d, f, g, h)	<(d)>-partition	<(d)>-partition
10	(b, f, g)(c, e, h)	<(b)>-partition	<(b)>-partition
11	(e, g)(f)(e, f)	<(e)>-partition	<(e)>-partition

The DISC-all Algorithm

- Strategy 2: Multi-level Partitioning

CID	Customer Sequences	Initial Partitions	After <(a)>-partition
1	(a, d)(d)(a, g, h)(c)	<(a)>-partition	<(c)>-partition
2	(b)(a)(f)(a, c, e, g)	<(a)>-partition	<(b)>-partition
3	(a, f, g)(a, e, g, h)(c, g, h)	<(a)>-partition	<(c)>-partition
4	(f)(a, c, f)(a, c, e, g, h)	<(a)>-partition	<(c)>-partition
5	(a, g)	<(a)>-partition	<(g)>-partition
6	(a, f)(a, e, g, h)	<(a)>-partition	<(e)>-partition
7	(a, b, g)(a, e, g)(g, h)	<(a)>-partition	<(b)>-partition
8	(b, f)(b, e)(e, f, h)	<(b)>-partition	<(b)>-partition
9	(d, f)(d, f, g, h)	<(d)>-partition	<(d)>-partition
10	(b, f, g)(c, e, h)	<(b)>-partition	<(b)>-partition
11	(e, g)(f)(e, f)	<(e)>-partition	<(e)>-partition

The DISC-all Algorithm

❑ Strategy 2: Multi-level Partitioning

CID	Customer Sequences	Initial Partitions	After <(a)>-partition
1	(a, d)(d)(a, g, h)(c)	<(a)>-partition	<(c)>-partition
2	(b)(a)(f)(a, c, e, g)	<(a)>-partition	<(b)>-partition
3	(a, f, g)(a, e, g, h)(c, g, h)	<(a)>-partition	<(c)>-partition
4	(f)(a, c, f)(a, c, e, g, h)	<(a)>-partition	<(c)>-partition
5	(a, g)	<(a)>-partition	<(g)>-partition
6	(a, f)(a, e, g, h)	<(a)>-partition	<(e)>-partition
7	(a, b, g)(a, e, g)(g, h)	<(a)>-partition	<(b)>-partition
8	(b, f)(b, e)(e, f, h)	<(b)>-partition	<(b)>-partition
9	(d, f)(d, f, g, h)	<(d)>-partition	<(d)>-partition
10	(b, f, g)(c, e, h)	<(b)>-partition	<(b)>-partition
11	(e, g)(f)(e, f)	<(e)>-partition	<(e)>-partition

The DISC-all Algorithm

❑ Strategy 3: Customer Sequence Reducing

➢ Reduce <(a)>-partition (δ=3)

- **Remove non-frequent 1-sequence (d)**
- **Remove non-frequent 2-sequences**
  - *(a)(b), (a)(d), (a)(f), (ab), (ac), and (ad)*

(a)(c) is frequent

CID	Customer Sequences
1	(a, d)(d)(a, g, h)(c)
2	(b)(a)(f)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, c, f)(a, c, e, g, h)
~~5~~	~~(a, g)~~
6	(a, f)(a, e, g, h)
7	(a, b, g)(a, e, g)(g, h)

→

CID	Reduced Customer Sequences
1	(a)(a, g, h)(c)
2	(b)(a)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, f)(a, c, e, g, h)
6	(a, f)(a, e, g, h)
7	(a, g)(a, e, g)(g, h)

The DISC-all Algorithm

❑ Strategy 3: Customer Sequence Reducing

➢ Reduce <(a)>-partition (δ=3)

- **Remove non-frequent 1-sequence (d)**
- **Remove non-frequent 2-sequences**
  – *(a)(b), (a)(d), (a)(f), (ab), (ac), and (ad)*

CID	Customer Sequences
1	(a, ~~d~~)~~(d)~~(a, g, h)(c)
2	(b)(a)(f)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, c, f)(a, c, e, g, h)
~~5~~	~~(a, g)~~
6	(a, f)(a, e, g, h)
7	(a, b, g)(a, e, g)(g, h)

→

(a)(c) is frequent

CID	Reduced Customer Sequences
1	(a)(a, g, h)(c)
2	(b)(a)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, f)(a, c, e, g, h)
6	(a, f)(a, e, g, h)
7	(a, g)(a, e, g)(g, h)

The DISC-all Algorithm

- **Strategy 3: Customer Sequence Reducing**
  - Reduce <(a)>-partition (δ=3)
    - **Remove non-frequent 1-sequence (d)**
    - **Remove non-frequent 2-sequences**
      - *(a)(b), (a)(d), (a)(f), (ab), (ac), and (ad)*

(a)(c) is frequent

CID	Customer Sequences
1	(a, ~~d~~)(~~d~~)(a, g, h)(c)
2	(b)(a)(~~f~~)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, ~~c~~, f)(a, c, e, g, h)
~~5~~	~~(a, g)~~
6	(a, f)(a, e, g, h)
7	(a, ~~b~~, g)(a, e, g)(g, h)

→

CID	Reduced Customer Sequences
1	(a)(a, g, h)(c)
2	(b)(a)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, f)(a, c, e, g, h)
6	(a, f)(a, e, g, h)
7	(a, g)(a, e, g)(g, h)

The DISC-all Algorithm

- **Strategy 3: Customer Sequence Reducing**
  - Reduce <(a)>-partition (δ=3)
    - **Remove non-frequent 1-sequence (d)**
    - **Remove non-frequent 2-sequences**
      - *– (a)(b), (a)(d), (a)(f), (ab), (ac), and (ad)*

CID	Customer Sequences
1	(a, ~~d~~)~~(d)~~(a, g, h)(c)
2	(b)(a)~~(f)~~(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, ~~c~~, f)(a, c, e, g, h)
~~5~~	~~(a, g)~~
6	(a, f)(a, e, g, h)
7	(a, ~~b~~, g)(a, e, g)(g, h)

(a)(c) is frequent

CID	Reduced Customer Sequences
1	(a)(a, g, h)(c)
2	(b)(a)(a, c, e, g)
3	(a, f, g)(a, e, g, h)(c, g, h)
4	(f)(a, f)(a, c, e, g, h)
6	(a, f)(a, e, g, h)
7	(a, g)(a, e, g)(g, h)

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition $\Rightarrow$ 4-sorted database
  - Use the 3-sorted list:

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition ⇒ 4-sorted database
  - Use the 3-sorted list:

No	Frequent 3-sequences
1	(a)(a, e)
2	(a)(a, g)
3	(a)(a, h)

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition $\Rightarrow$ 4-sorted database
  - Use the 3-sorted list:

No	Frequent 3-sequences
1	(a)(a, e)
2	(a)(a, g)
3	(a)(a, h)

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition ⇒ 4-sorted database
  - Use the 3-sorted list:

No	Frequent 3-sequences
1	(a)(a, e)
2	(a)(a, g)
3	(a)(a, h)

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition ⇒ 4-sorted database
  - Use the 3-sorted list:

No	Frequent 3-sequences
1	(a)(a, e)
2	(a)(a, g)
3	(a)(a, h)

α_δ α_1

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition ⇒ 4-sorted database
  - Use the 3-sorted list:

No	Frequent 3-sequences
1	(a)(a, e)
2	(a)(a, g)
3	(a)(a, h)

α_δ α_1

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

The DISC-all Algorithm

- **Strategy 1+4: Candidate Sequence Pruning+DISC**
  - <(a)(a)>-partition ⇒ 4-sorted database
  - Use the 3-sorted list:

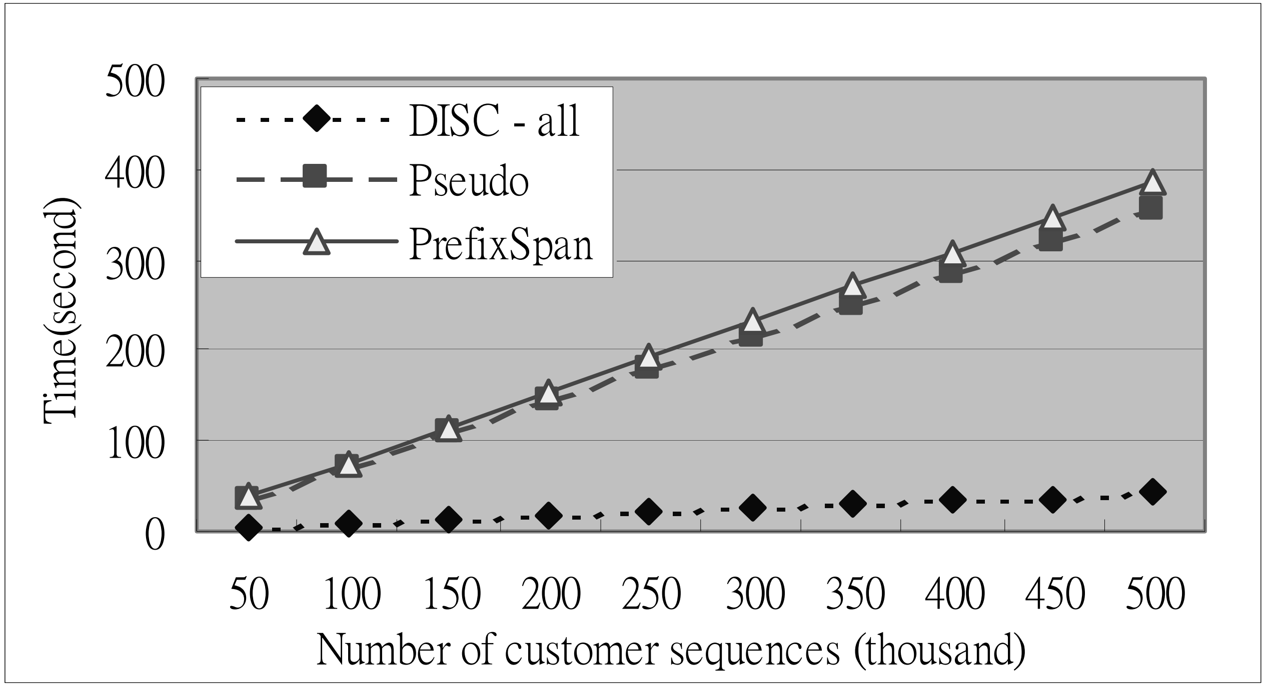
No	Frequent 3-sequences
1	(a)(a, e)
2	(a)(a, g)
3	(a)(a, h)

α_δ α_1

CID	4-minimum Subsequences	Customer Sequences	Pointer
3	(a)(a, e)(c)	(a, f, g)(a, e, g, h)(c, g, h)	1
2	(a)(a, e, g)	(b)(a)(a, c, e, g)	1
4	(a)(a, e, g)	(f)(a, f)(a, c, e, g, h)	1
6	(a)(a, e, g)	(a, f)(a, e, g, h)	1
7	(a)(a, e, g)	(a, g)(a, e, g)(g, h)	1
1	(a)(a, g)(c)	(a)(a, g, h)(c)	2

Performance Evaluation

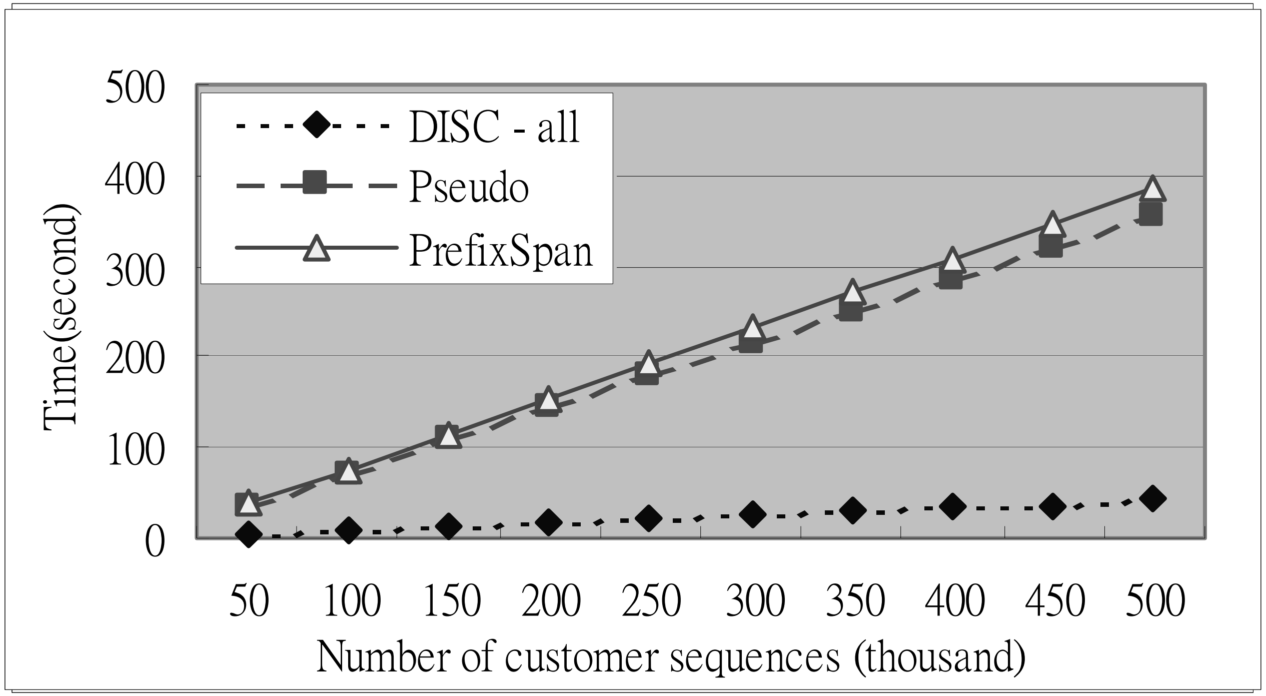
- **Comparisons with PrefixSpan and Pseudo**
  - Database size
  - Minimum support threshold



Performance Evaluation

❑ Comparisons with PrefixSpan and Pseudo

- ➢ Database size
- ➢ Minimum support threshold

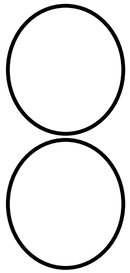


Performance Evaluation

- **Non-reduction rate (NRR)**

$$NRR_Q = \frac{1}{N_Q} \sum_{p \text{ is a child partition of } Q} \frac{Size_p}{Size_Q}$$

  - Average NRR vs. DISC-all's improvement
  - Partitioning strategy can benefit from the partition with a low NRR



Performance Evaluation

- **Non-reduction rate (NRR)**

$$NRR_Q = \frac{1}{N_Q} \sum_{p \text{ is a child partition of } Q} \frac{Size_p}{Size_Q}$$

  - Average NRR vs. DISC-all's improvement
  - Partitioning strategy can benefit from the partition with a low NRR

MST \ Average NRR	Original	1	2	3	4	5	6	7
0.0025	0.0018	0.08	0.43	0.83	0.85	0.85	0.86	0.87
0.005	0.0019	0.11	0.64	0.9	0.94	0.97	0.99	-
0.0075	0.002	0.12	0.9	0.98	0.98	-	-	-
0.01	0.0022	0.14	0.92	-	-	-	-	-
0.0125	0.0024	0.15	-	-	-	-	-	-
0.015	0.0025	0.16	-	-	-	-	-	-
0.0175	0.0026	0.18	-	-	-	-	-	-

Performance Evaluation

- **Non-reduction rate (NRR)**

$$NRR_Q = \frac{1}{N_Q} \sum_{p \text{ is a child partition of } Q} \frac{Size_p}{Size_Q}$$

  - Average NRR vs. DISC-all's improvement
  - Partitioning strategy can benefit from the partition with a low NRR

MST \ Average NRR	Original	1	2	3	4	5	6	7
0.0025	0.0018	0.08	0.43	0.83	0.85	0.85	0.86	0.87
0.005	0.0019	0.11	0.64	0.9	0.94	0.97	0.99	-
0.0075	0.002	0.12	0.9	0.98	0.98	-	-	-
0.01	0.0022	0.14	0.92	-	-	-	-	-
0.0125	0.0024	0.15	-	-	-	-	-	-
0.015	0.0025	0.16	-	-	-	-	-	-
0.0175	0.0026	0.18	-	-	-	-	-	-

MST	Pseudo/DISC-all
0.0025	3.588026
0.005	7.723559
0.0075	8.294165
0.01	8.140177
0.0125	7.792428
0.015	7.445235
0.0175	6.962742

Performance Evaluation

- **Non-reduction rate (NRR)**

$$NRR_Q = \frac{1}{N_Q} \sum_{p \text{ is a child partition of } Q} \frac{Size_p}{Size_Q}$$

  - Average NRR vs. DISC-all's improvement
  - Partitioning strategy can benefit from the partition with a low NRR

MST \ Average NRR	Original	1	2	3	4	5	6	7
0.0025	0.0018	0.08	0.43	0.83	0.85	0.85	0.86	0.87
0.005	0.0019	0.11	0.64	0.9	0.94	0.97	0.99	-
0.0075	0.002	0.12	0.9	0.98	0.98	-	-	-
0.01	0.0022	0.14	0.92	-	-	-	-	-
0.0125	0.0024	0.15	-	-	-	-	-	-
0.015	0.0025	0.16	-	-	-	-	-	-
0.0175	0.0026	0.18	-	-	-	-	-	-

MST	Pseudo/DISC-all
0.0025	3.588026
0.005	7.723559
0.0075	8.294165
0.01	8.140177
0.0125	7.792428
0.015	7.445235
0.0175	6.962742

Performance Evaluation

- **Non-reduction rate (NRR)**

$$NRR_Q = \frac{1}{N_Q} \sum_{p \text{ is a child partition of } Q} \frac{Size_p}{Size_Q}$$

  - Average NRR vs. DISC-all's improvement
  - Partitioning strategy can benefit from the partition with a low NRR

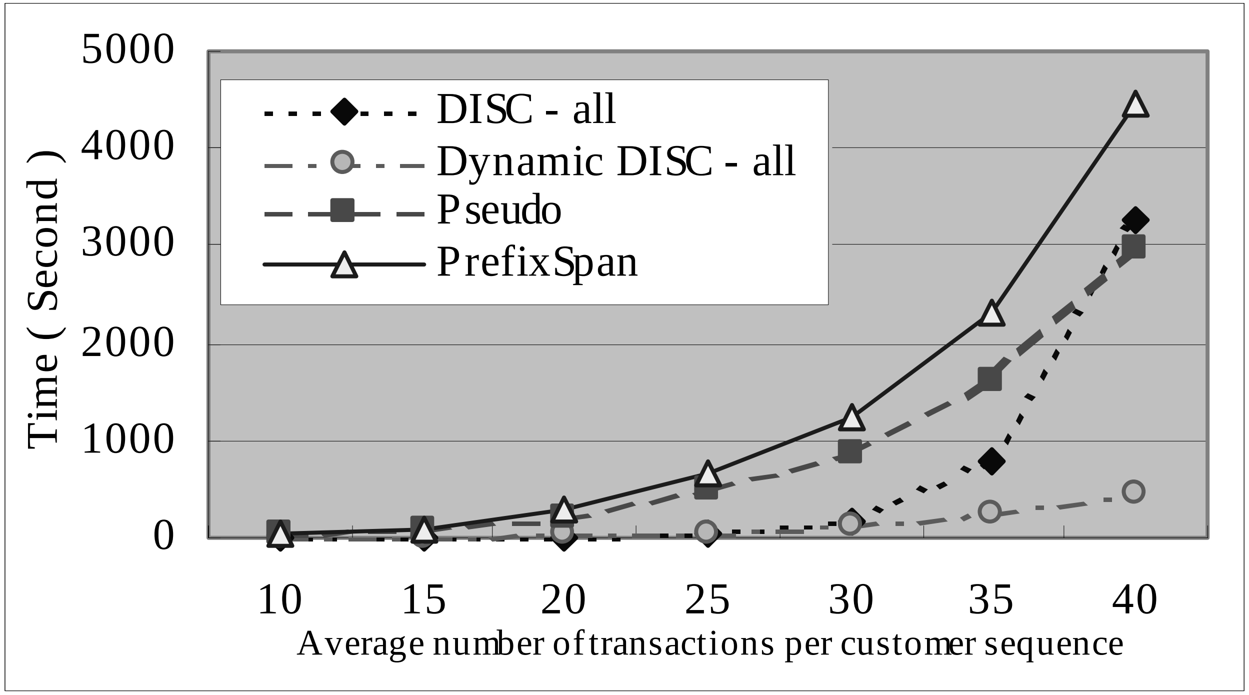
MST \ Average NRR	Original	1	2	3	4	5	6	7
0.0025	0.0018	0.08	0.43	0.83	0.85	0.85	0.86	0.87
0.005	0.0019	0.11	0.64	0.9	0.94	0.97	0.99	-
0.0075	0.002	0.12	0.9	0.98	0.98	-	-	-
0.01	0.0022	0.14	0.92	-	-	-	-	-
0.0125	0.0024	0.15	-	-	-	-	-	-
0.015	0.0025	0.16	-	-	-	-	-	-
0.0175	0.0026	0.18	-	-	-	-	-	-

MST	Pseudo/DISC-all
0.0025	3.588026
0.005	7.723559
0.0075	8.294165
0.01	8.140177
0.0125	7.792428
0.015	7.445235
0.0175	6.962742

Performance Evaluation

❑ The Dynamic DISC-all Algorithm

➢ Maximum NRR threshold



Conclusion

- **Contribution**
  - The DISC strategy for mining sequential patterns
  - The DISC-all algorithm that takes advantage of all the strategies
  - The Dynamic DISC-all algorithm that can achieve a much better performance
- **Future Work**
  - Apply the DISC strategy to *weighting applications*
    - **Web log analysis, biological data mining**