UNIVERSITÀ CA' FOSCARI DI VENEZIA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

Corso di Data Mining
Approfondimento

SPADE

Studente: Marco Lionello

Anno Accademico 2005-2006

Introduzione a SPADE (1/2)

- Sequential Pattern Discovery using Equivalence Classes
- Caratteristiche:
  - Suddivisione del problema in sottoproblemi
  - Efficiente tecnica di ricerca “lattice”
  - L’algoritmo termina con al max tre scansioni del database
  - Altamente scalabile
- Scoperta delle sequenze θ(m^K) dove m=attributi K=lunghezza
- Regole che descrivono eventi in sequenza temporale
- Esempio il 70% delle persone che compra “Codice da vinci” compra successivamente “Angeli e Demoni”
- SPADE non utilizza scansioni multiple del database ne Hash tree complicati

Database originale (2/2)

Goal di SPADE:

- Uso di un database verticale
- Uso della teoria lattice
- Disaccoppiamento dei problemi di ricerca e scomposizione

DATABASE

Customer-Id	Transaction-Time	Items
1	10	C D
1	15	A B C
1	20	A B F
1	25	A C D F
2	15	A B F
2	20	E
3	10	A B F
4	10	D G H
4	20	B F
4	25	A G H

FREQUENT SEQUENCES

Frequent 1-Sequences

A	4
B	4
D	2
F	4

Frequent 2-Sequences

AB	3
AF	3
B->A	2
BF	4
D->A	2
D->B	2
D->F	2
F->A	2

Frequent 3-Sequences

ABF	3
BF->A	2
D->BF	2
D->B->A	2
D->F->A	2

Frequent 4-Sequences

D->BF->A	2

Alcune definizioni (1/2)

- **Item**: insieme di m attributi distinti I={ i_1,….,i_m }
- **Itemset**: collezione disordinata e non vuota di item
- **Sequenza**: lista di item ordinata
- **K-sequenza**: sequenza di k item
- **Sotto Sequenza**: una sequenza A(a_1→ …. →a_n) è una sottosequenza di B(b_1→ …. →b_n) e si indica con A≤B se esiste un intero i_1≤i_2≤…≤i_n per il quale a_j è contenuto in b_{ij} per ogni a_j

Esempio: **(B→AC) ≤ (AB→E→ACD)** perchè B contenuto in AB e AC contenuto in ACD

- **Sotto Sequenza propria**: si denota con A<B se A≤B ma non B≤A
- **Sequenza massima**: se non è sottosequenza si altre sequenze

Alcune definizioni (2/2)

- **Transazione**: record contenente un insieme di item e un unico identificatore
- **Cliente**: record costituito da un'identificatore unico a cui sono associate una lista di transazioni

ASSUNZIONI:

- Nessun cliente ha più di una transazione con la stessa marca temporale quindi TID= tempo della transazione
- Lista delle transazioni-cliente ordinata sui clienti e sulla transazione

- **$\alpha \leq C$** Indica il cliente C contiene la sequenza α
- **Supporto**: σ(α) è il numero totale di clienti che contiene la sequenza α

Teoria “lattice”

- Sia P un Itemset. Un *ordine parziale* L in P è una relazione
  - Riflessiva: $X \leq X$
  - Antissimetrica: da $X \leq Y$ e da $Y \leq X$ si ha $X=Y$

- Un ordine parziale si dice *Lattice* se le due operazioni
  - Join X,Y
  - Meet X,Y

  esistono per ogni X,Y appartenenti a L.

- Un Lattice si dice *completo* se ogni join e meet esistono su ogni sottoinsieme arbitrario di L.

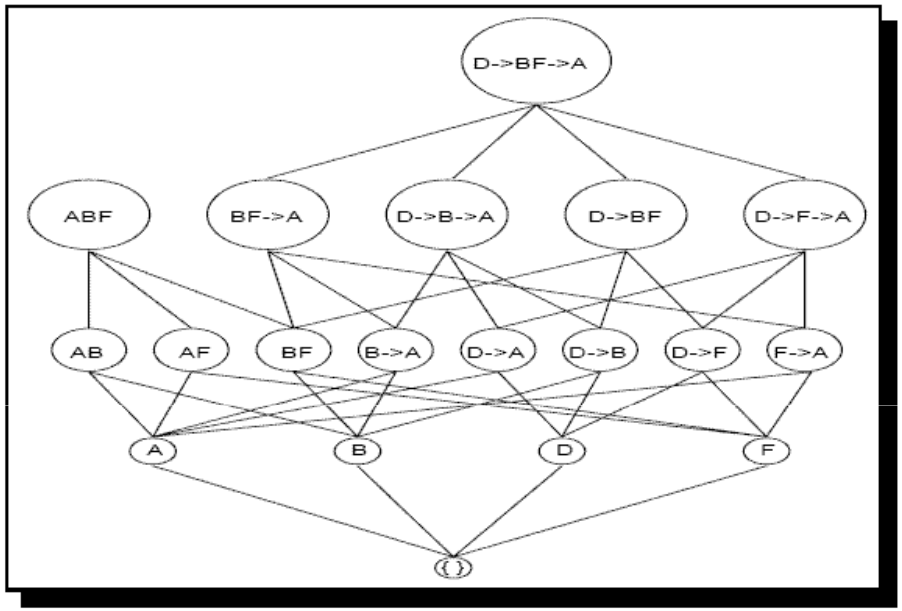
- Tutti i Lattice *finiti* sono completi.

Teoremi e lemmi

- **Teorema:** Dato in insieme di item I, l'insieme ordinato S di tutte le possibili sequenze di item, è un lattice completo nel quale join e meet sono date rispettivamente dall'unione e intersezione.

- **Lemma1:** Sottosequenze di sequenze frequenti sono frequenti
- **Lemma2**: Ogni sequenza può essere ottenuta come unione dei suoi atomi e il suo supporto è ottenibile come intersezione delle rispettive id-list.
- **Lemma3:** Ogni supporto di una k-sequenza è ottenibile intersecando alcune delle sue k-1-sequenze
- **Lemma4:** Se X è sottosequenza di Y allora la cardinalità di Y è minore o uguale a quella di X

Esempio D→BF→A (1/2)



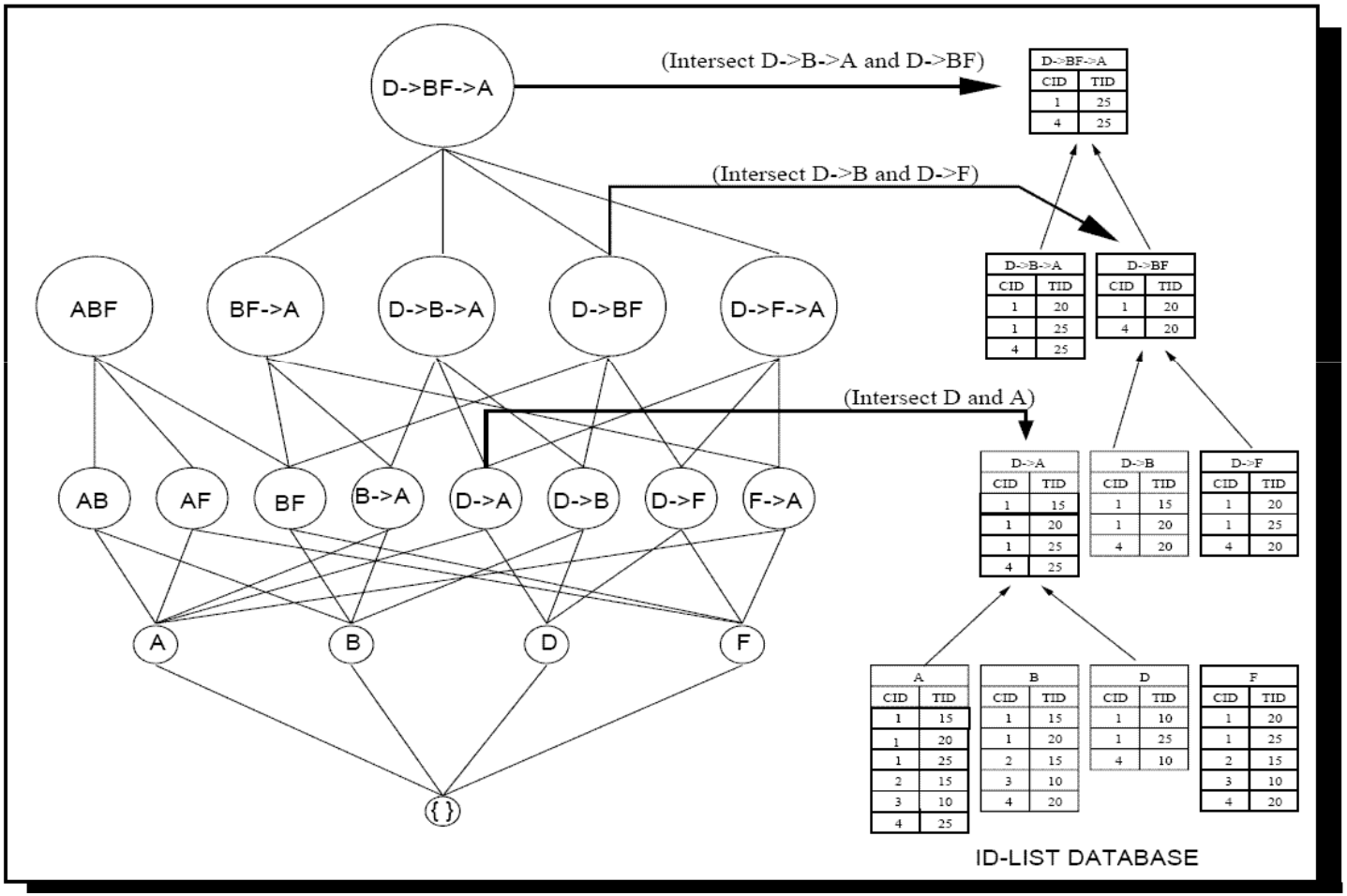
A	
CID	TID
1	15
1	20
1	25
2	15
3	10
4	25

B	
CID	TID
1	15
1	20
2	15
3	10
4	20

D	
CID	TID
1	10
1	25
4	10

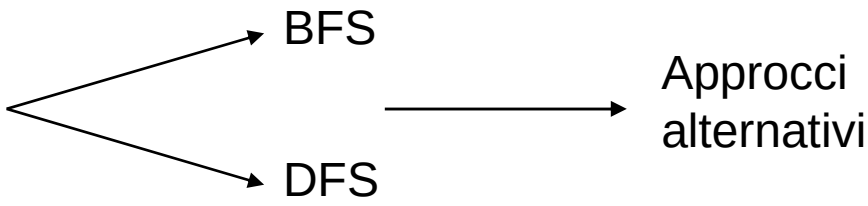
F	
CID	TID
1	20
1	25
2	15
3	10
4	20

Esempio D→BF→A (2/2)



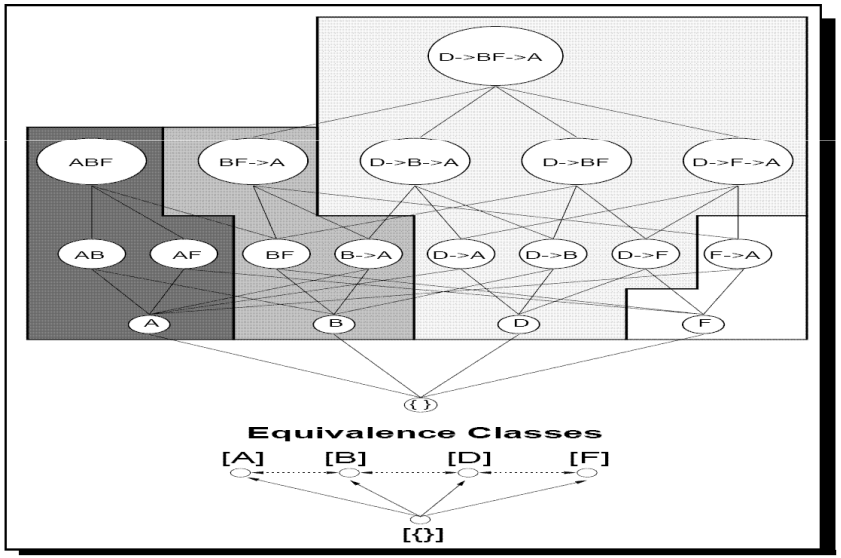
Decomposizione del lattice

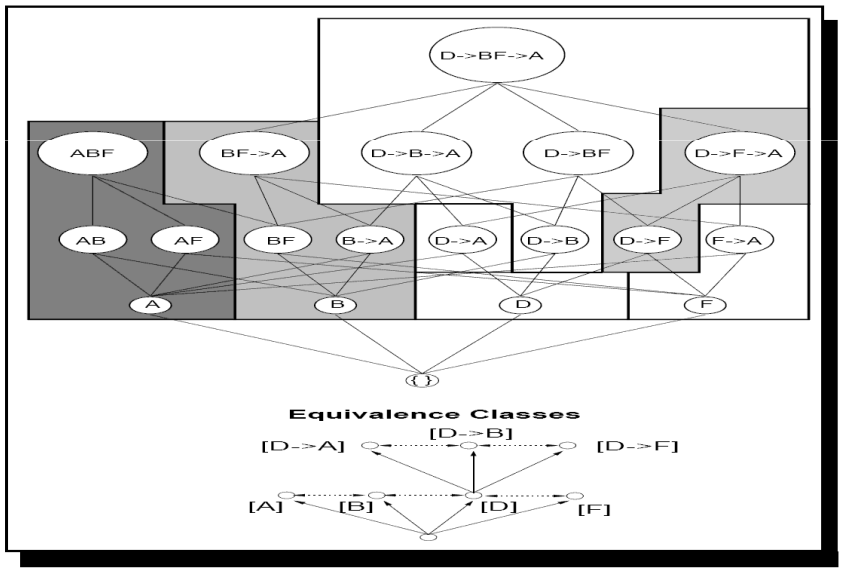
- Una *relazione di equivalenza* è una relazione :riflessiva , simmetrica e transitiva di tipo binario.
- Una *classe di equivalenza* è il risultato del partizionamento dell'insieme in sottoinsiemi disgiunti.
  - Due sequenze X e Y sono della stessa classe sse condividono la stessa lunghezza di prefisso k "prefix based equivalence"
- **Lemma5:** ogni classe di equivalenza indotta dalla relazione θk è un sub-lattice.
- Due strategie di ricerca:



Classi di equivaleza

- Indotta da θ_1 in S
- Indotta da θ_1 in S e θ_2 in $[D]_{\theta_1}$





SPADE l'algoritmo

- Livello 1: sfruttiamo il database verticale. Intera scansione del database si incrementa il supporto quando si incontra un nuovo CID.

- Livello2: Implementazione naive: computazione di tutte le possibili intersezioni delle id-list →N/2 data scan (N numero di item frequenti).

Approcci alternativi:

1. Utilizzo di informazioni pre-processate
2. Trasformazione del database "al volo" in orrizzontale.

Recovery del database da verticale a orrizzontale

•Per ogni cliente e transazione (c,t) in L(i) inseriamo (i,t) nella lista del customer c
•Si forma una lista di tutte le 2-sequenze in ogni sequenza sei clienti, e si aggiorna il conto in un array bidimensionale indicizzato dagli item frequenti
•Seconda scansione del database

cid	$(item, tid)$ pairs
1	(A 15) (A 20) (A 25) (B 15) (B 20) (C 10) (C 15) (C 25) (D 10) (D 25) (F 20) (F 25)
2	(A 15) (B 15) (E 20) (F 15)
3	(A 10) (B 10) (F 10)
4	(A 25) (B 20) (D 10) (F 20) (G 10) (G 25) (H 10) (H 25)

Sequenze frequenti con k≥3

- L'input è un sub-lattice S.
- Le sequenze frequenti vengono generate come unione di id-list di livello precedente.
- Prima di effettuare l'intersezione viene effettuato una potatura.
- Si caricano quindi solo le sequenze frequenti → terza scan del database

PRUNING

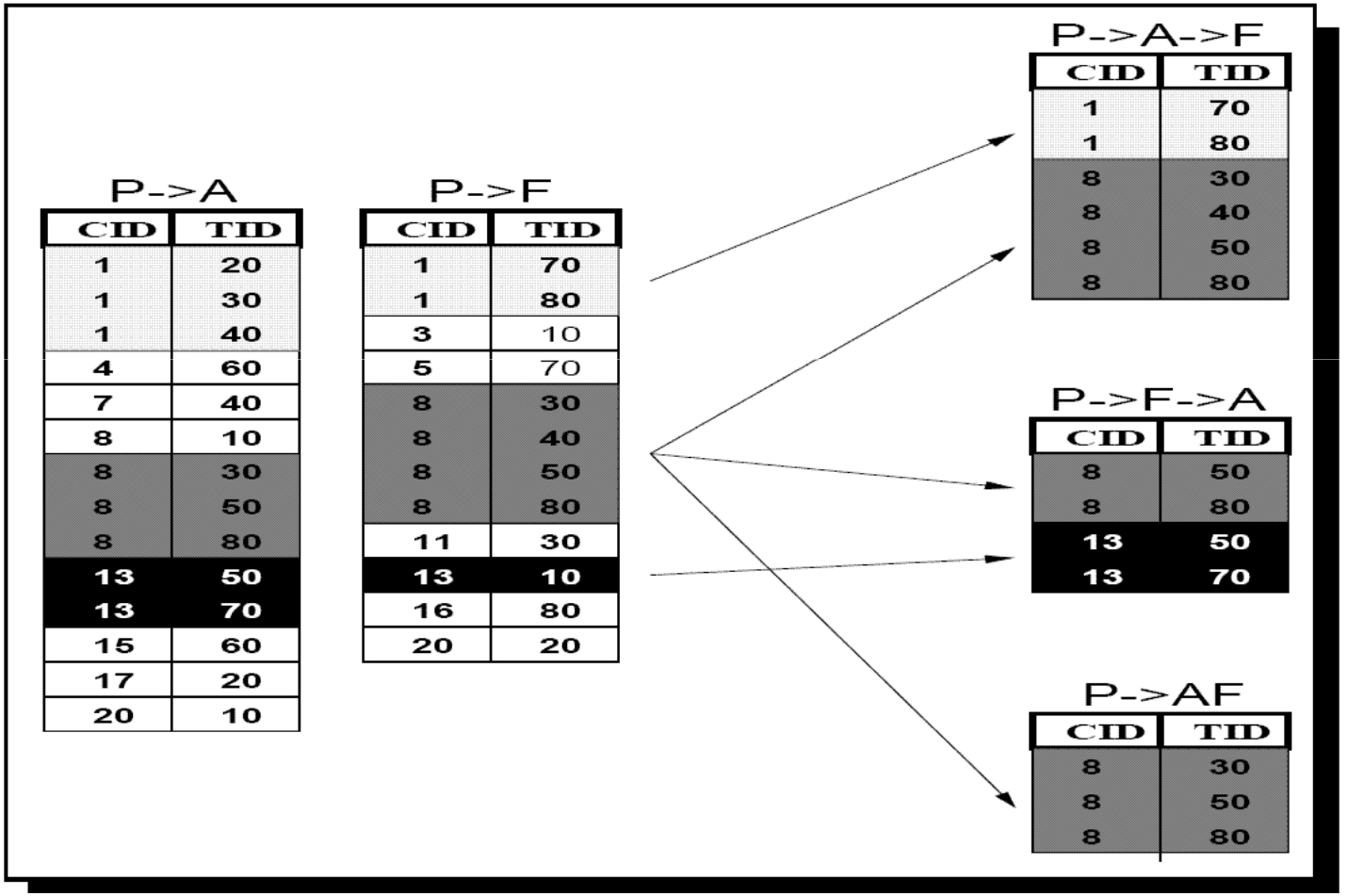
- Evita computazione inutile
- Se le (k-1)-sequenze della sequenza S non sono frequenti pota S

Intersezione delle id-list (1/2)

- Si consideri [B→A] e {B→AB,B→AD,B→A→A,B→A→D,B→A→F} se si sostituisce P con il prefisso B→A si ottengono due classi:
  - {P→A, P→D, P→F}
  - {PD, PB}

- Tre tipi di intersezione per ottenere atomi:
  - Itemset atom con Itemset atom: da PB e PD si ottiene PDB
  - Itemset atom con Sequence atom: da PB e P→A si ottiene PB→A
  - Sequence atom con Sequence atom: da P→A e P→F si ottiene
    - Nuovo itemset atom P→AF
    - Nuovo sequence atom P→A→F
    - Nuovo sequence atom P→F→A

Intersezione delle id-list (1/2)



L'algoritmo SPADE

SPADE $(min\_sup, \mathcal{D})$:
$\mathcal{F}_1 = \{$ frequent items or 1-sequences $\}$;
$\mathcal{F}_2 = \{$ frequent 2-sequences $\}$;
$\mathcal{E} = \{$ equivalence classes $[X]_{\theta_1}\}$;
for all $[X] \in \mathcal{E}$ **do** *Enumerate-Frequent-Seq*([X]);

Enumerate-Frequent-Seq(S)**:**
for all atoms $A_i \in S$ **do**
$T_i = \emptyset$;
for all atoms $A_j \in S$, with $j > i$ **do**
$R = A_i \cup A_j$;
if (*Prune*(R) == FALSE) **then**
$\mathcal{L}(R) = \mathcal{L}(A_i) \cap \mathcal{L}(A_j)$;
if $\sigma(R) \geq min\_sup$ **then**
$T_i = T_i \cup \{R\}$; $\mathcal{F}_{|R|} = \mathcal{F}_{|R|} \cup \{R\}$;
end
if (Depth-First-Search) **then** *Enumerate-Frequent-Seq*(T_i);
end
if (Breadth-First-Search) **then**
for all $T_i \neq \emptyset$ **do** *Enumerate-Frequent-Seq*(T_i);

Prune (β):
for all $(k-1)$-subsequences, $\alpha \prec \beta$ **do**
if ($[\alpha_1]$ has been processed, and $\alpha \notin \mathcal{F}_{k-1}$) **then**
return TRUE;
return FALSE;

Risultati Sperimentali (1/5)

Test esguiti su una macchina con:

- 100 Mhz MIPS
- 256MB di RAM
- O.S. IRIX 6.2
- 2 GB di disco non locale

$\mathcal{F}_1$ = { frequent 1-sequences };
for $(k = 2; \mathcal{F}_{k-1} \neq \emptyset; k = k + 1)$ **do**
  C_k = Set of candidate k-sequences;
  for all customer-sequences $\mathcal{E}$ in the database **do**
    Increment count of all $\alpha \in C_k$ contained in $\mathcal{E}$
  $\mathcal{F}_k = \{\alpha \in C_k | \alpha.sup \geq min\_sup\}$;
Set of all frequent sequences = $\bigcup_k \mathcal{F}_k$;

Due dataset di test:

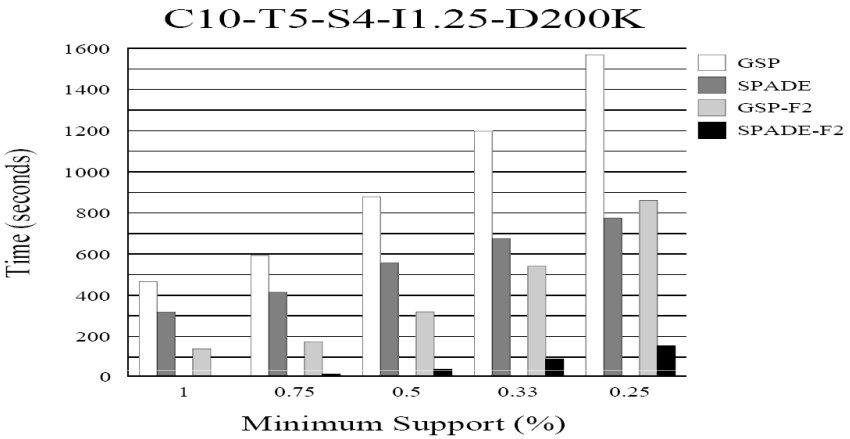
- **Dataset sintetico**: fornito dall'IBM nel "Quest Data Mining Project"

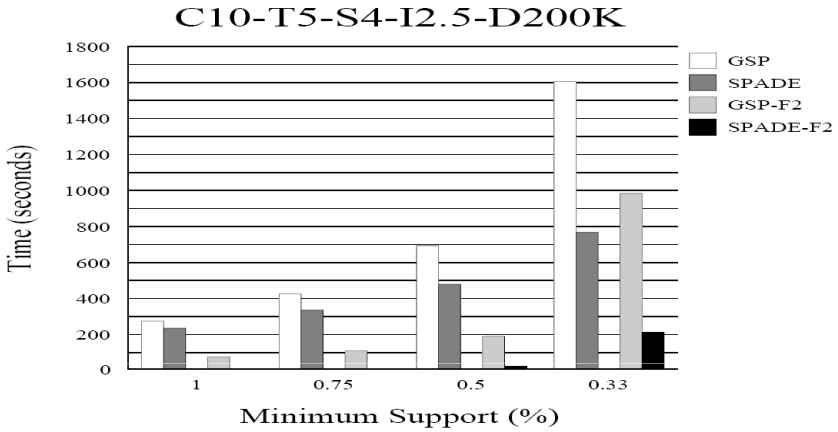
- **Dataset di pianificazione**: lo scopo del mining è quello di trovare i percorsi fallimentari
  - 77 (item)
  - 202071 piani (clienti)
  - 829236 eventi (transazioni)

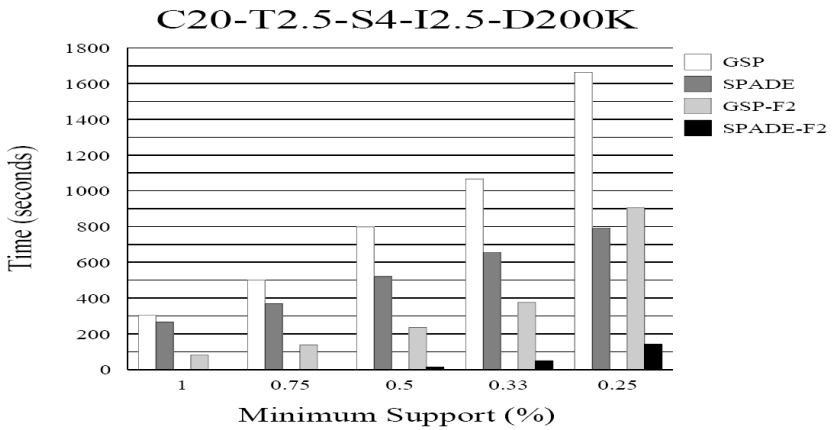
Dataset	Size (MB)
C10-T2.5-S4-I1.25-D200K	36.8
C10-T2.5-S4-I1.25-D500K	92.0
C10-T2.5-S4-I1.25-D1000K	184.0
C10-T5-S4-I1.25-D200K	56.5
C10-T5-S4-I2.5-D200K	54.3
C20-T2.5-S4-I1.25-D200K	76.7
C20-T2.5-S4-I2.5-D200K	66.5
C20-T2.5-S8-I1.25-D200K	76.4

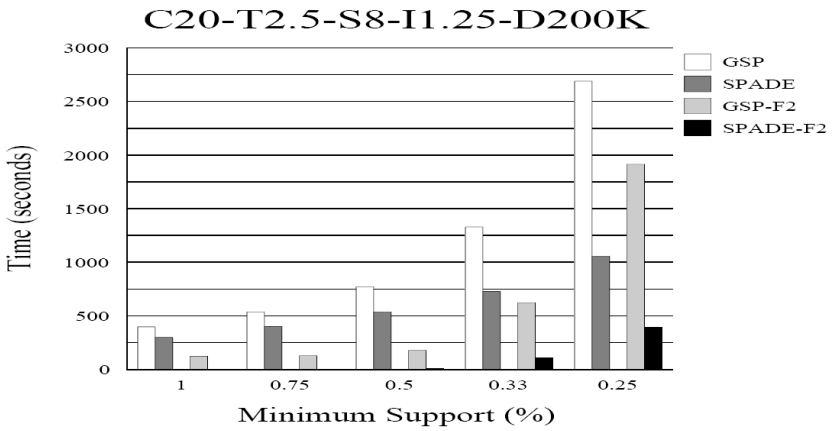
Table 1: Synthetic Datasets

Risultati Sperimentali (2/5)

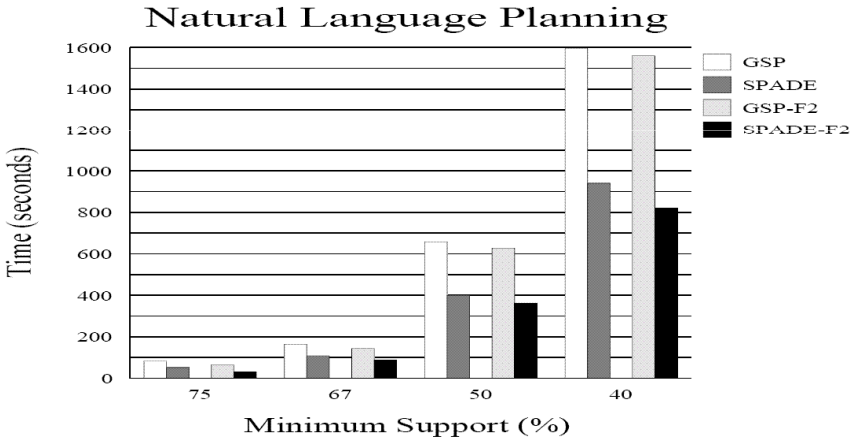


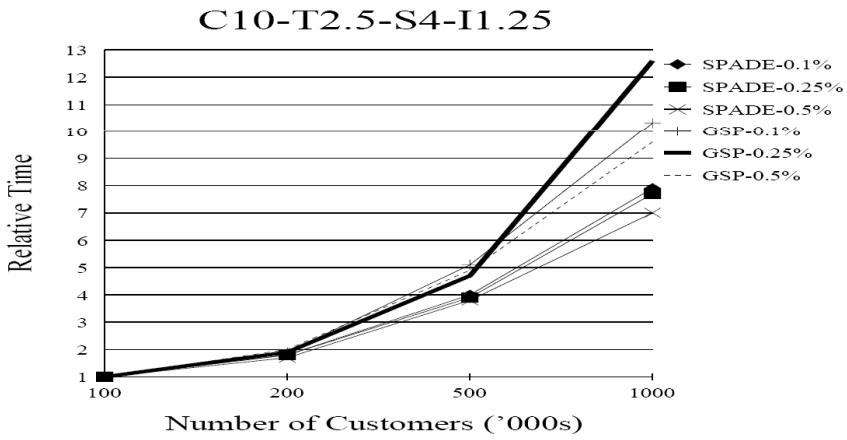






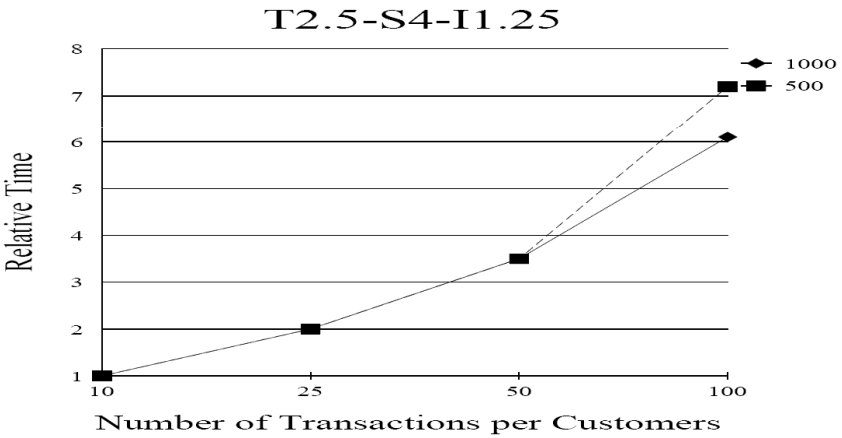
Risultati Sperimentali (3/5)

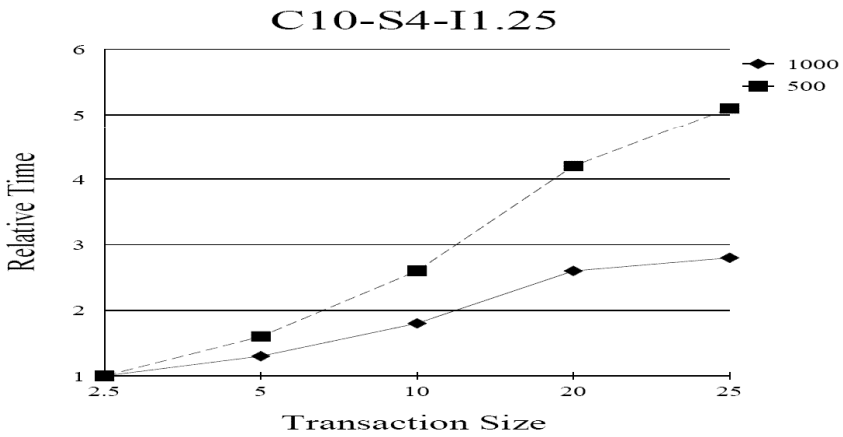




Risultati Sperimentali (4/5)

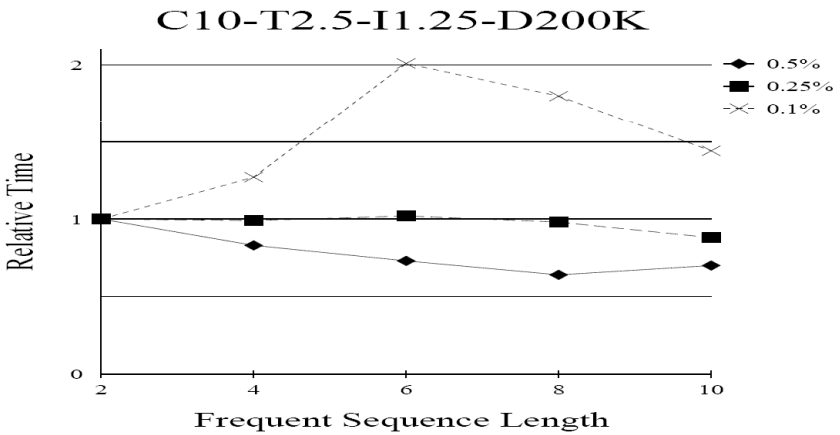
- Transazioni / Customer
- Dimensione della transazione

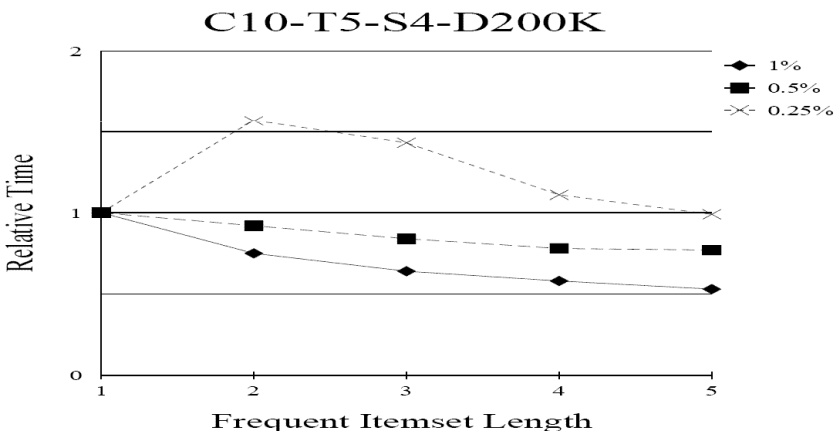




Risultati Sperimentali (5/5)

- Lunghezza delle sequenze frequenti
- Lunghezza degli itemset frequenti





Conclusioni

- SPADE non utilizza:
  - Scan multipli del database
  - Hash complicati
- SPADE utilizza:
  - Decomposizione in sottoproblemi
  - Solo tre scan del database
- Rispetto a GSP è minimo due volte più performante
- Buona scalabilità su:
  - Parametri
  - Customer
  - Transazioni
  - Dimensione delle transazioni

Bibliografia

[1] ZAKI M. J., *Efficient Enumeration of Frequent Sequences*., 7th International Conference on Information and Knowledge Management, Washington DC, November 1998.

[2] DAVEY, B. A. and PRIESTLEY, H. A. (1990)., *Introduction to Lattices and Order*, Cambridge University Press.

[3] AGRAWAL, R.(1996)., *Fast discovery of association rules*., Advances in KDD, AAAI Press.

[4] HAN J. and KAMBER M.(2001)., *Data Mining: Concepts and Techniques*., Morgan Kaufmann.

[5] DUNHAM M. H., *Data Mining: Introductory and Advanced Topics*., Prentice Hall.