

D2.P2:
A Query Planner Based on
Quantitative and Structural
Information

Francesco Scarcello

Alfredo Mazzitelli

Università della Calabria

Query Planners

- Given a query and some information on the database, find the best query plan
- A hard problem, if the query involves many relations
- Populating and refreshing data warehouses often requires a number of quite long batch queries on the reconciled database
- Many techniques
 - Quantitative methods
 - Structural methods
- Our prototype combine these separate worlds

The Prototype

- Takes as its input a query (SQL or Datalog) and quantitative information on the data
- Outputs:
 - An optimal hypertree decomposition HD
 - A script representing a query plan corresponding to HD (with some hints for the Oracle optimizer)
- Note that it deals with non-Boolean queries, possibly containing aggregate operators

Two different approaches

1. Data oriented:

- Maintain a dictionary with information on the data
- Exploit this information for finding the best query plan

2. Structure Oriented:

- Look for structural properties of the query
- Possibly, answer the query in (output) polynomial time

What is actually used?

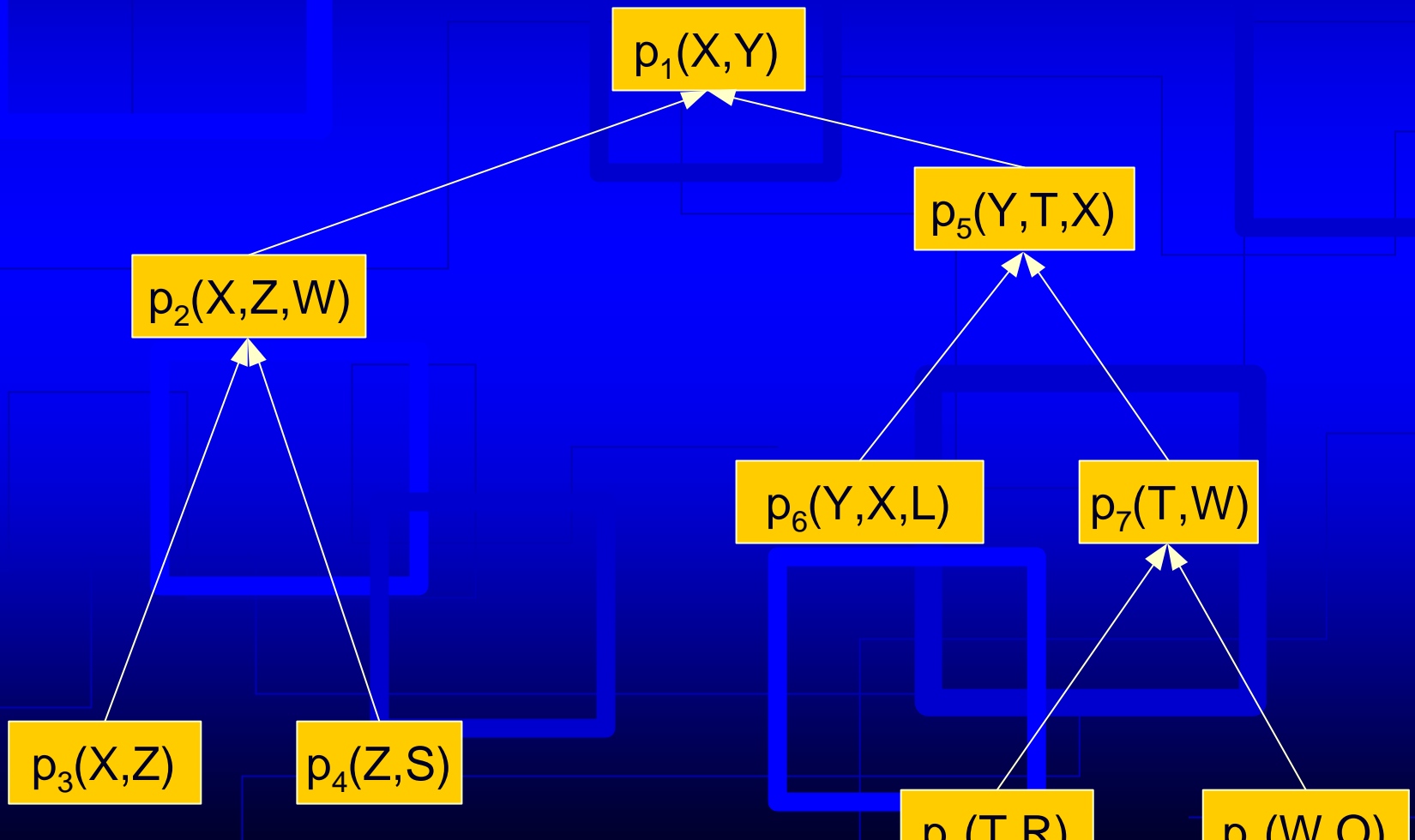
- All commercial DBMS are equipped with Data-oriented query planners
- Many important theoretical papers deals with structural properties of queries

 Wide tractability classes have been identified

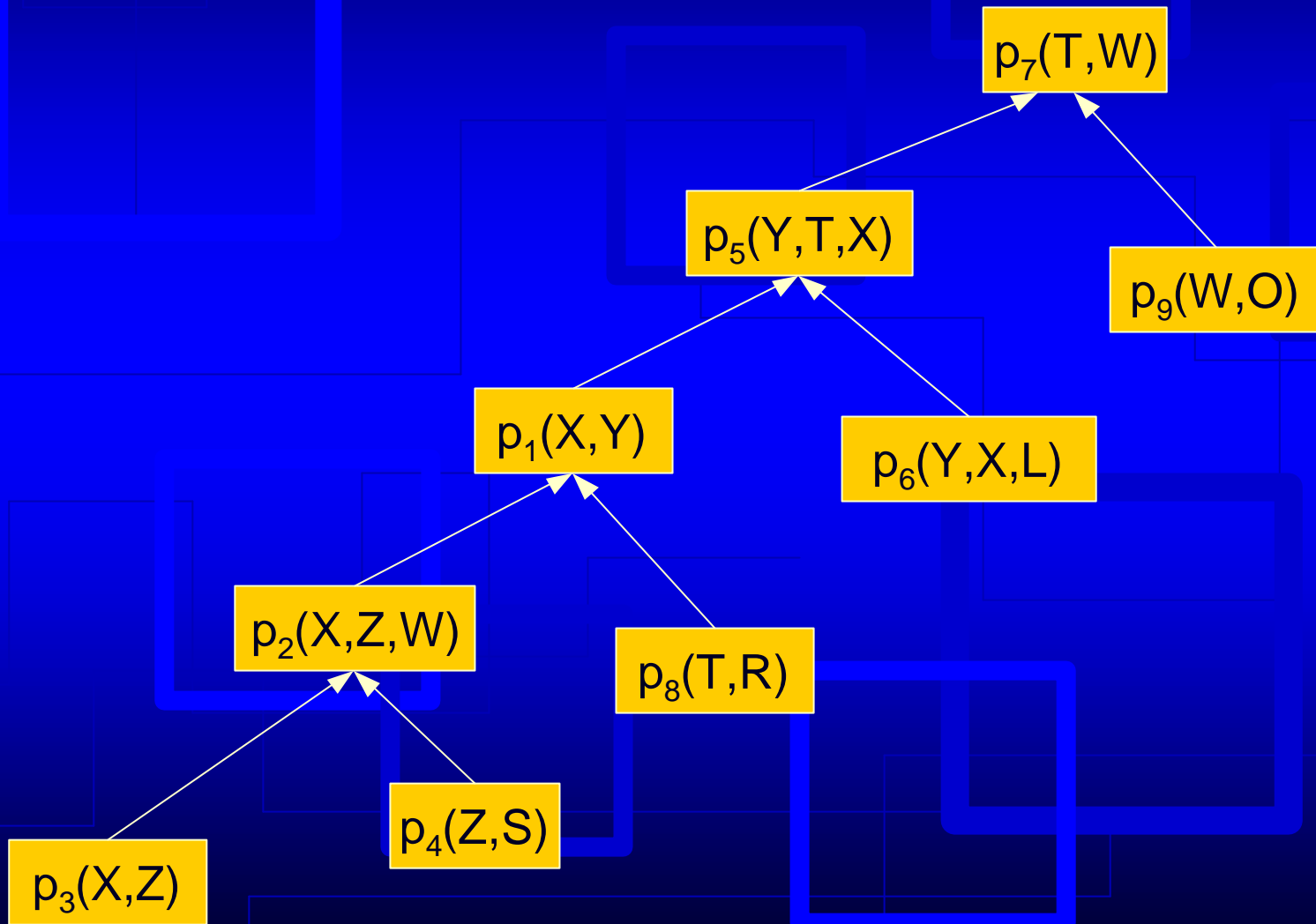
Why?

- **Data Oriented:**
 - Good amount of statistical information available
 - Efficient algorithms (mainly heuristics)
 - Simple kind of queries (typically short queries)
- **Structure Oriented:**
 - Polynomial Upper Bound on the cost of answering the query
 - Very good for long queries
 - Able to deal with databases without information on the data

What is the best choice?

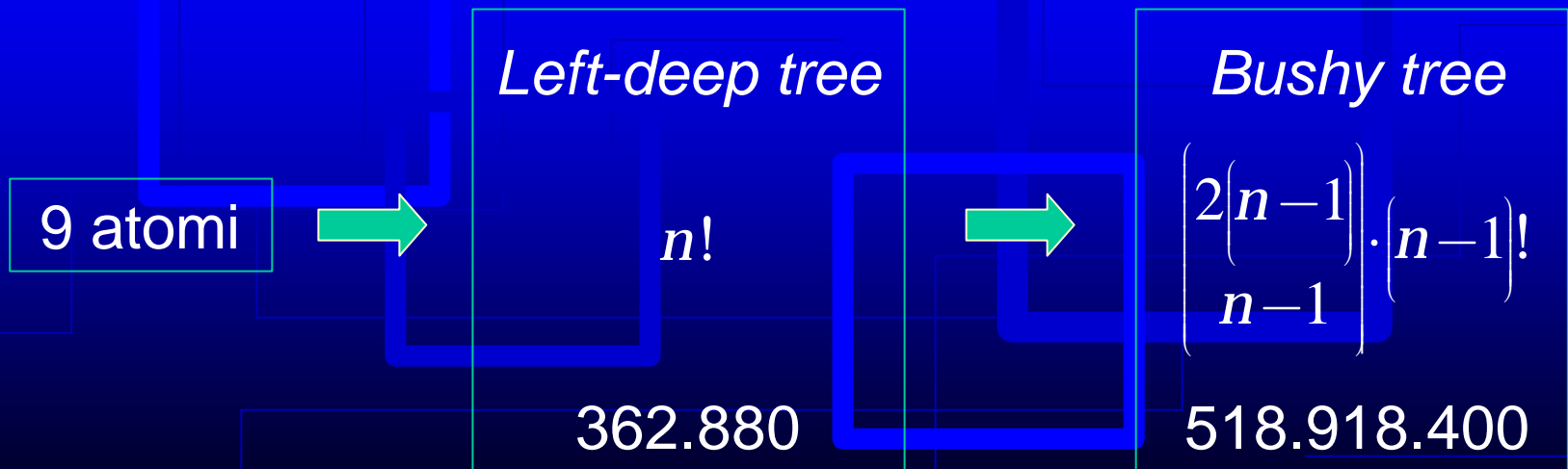


What is the best choice?



Some data oriented planners

- Dynamic programming
- Greedy
- Iterative Dynamic Programming
[Kossmann & Stoker, TODS, 2000]
- Restrictions of the search space:



Structural methods: acyclicity

- The answer of an acyclic (conjunctive) query can be computed in **output polynomial time**
[Yannakakis, VLDB'81]
- An Acyclic Boolean Conjunctive Query (**ABCQ**) Q may be answered in $O(|T| n \log n)$, where T is a *join tree* for Q and n is the size of the largest relation
- ABCQ is **LOGCFL-complete** and thus is highly parallelizable [GLS 98]
- Yannakakis's Algorithm is based on the use of **semi-join**

How to deal with such a long query?

$ans \leftarrow a(S, X, X', C, F) \wedge b(S, Y, Y', C', F') \wedge c(C, C', Z) \wedge d(X, Z) \wedge$
 $e(Y, Z) \wedge f(F, F', Z') \wedge g(X', Z') \wedge h(Y', Z') \wedge$
 $j(J, X, Y, X', Y') \wedge p(B, X', F) \wedge q(B', X', F)$

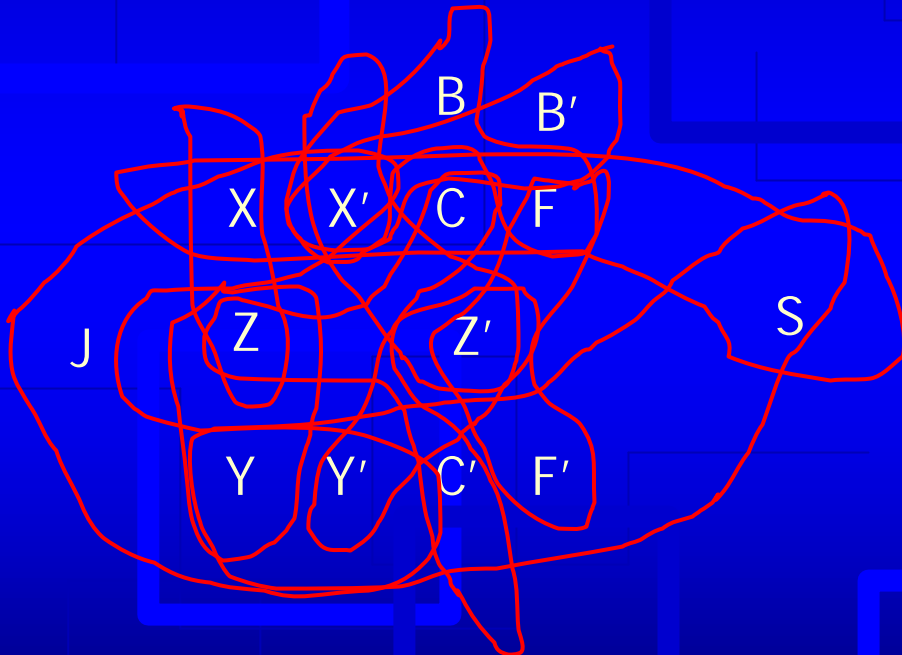
$m = 11 !$

n size of the database
 m number of atoms in the query

- Classical methods worst-case complexity: $O(n^m)$

Look at the hypergraph...

$ans \leftarrow a(S, X, X', C, F) \wedge b(S, Y, Y', C', F') \wedge c(C, C', Z) \wedge d(X, Z) \wedge$
 $e(Y, Z) \wedge f(F, F', Z') \wedge g(X', Z') \wedge h(Y', Z') \wedge$
 $j(J, X, Y, X', Y') \wedge p(B, X', F) \wedge q(B', X', F)$

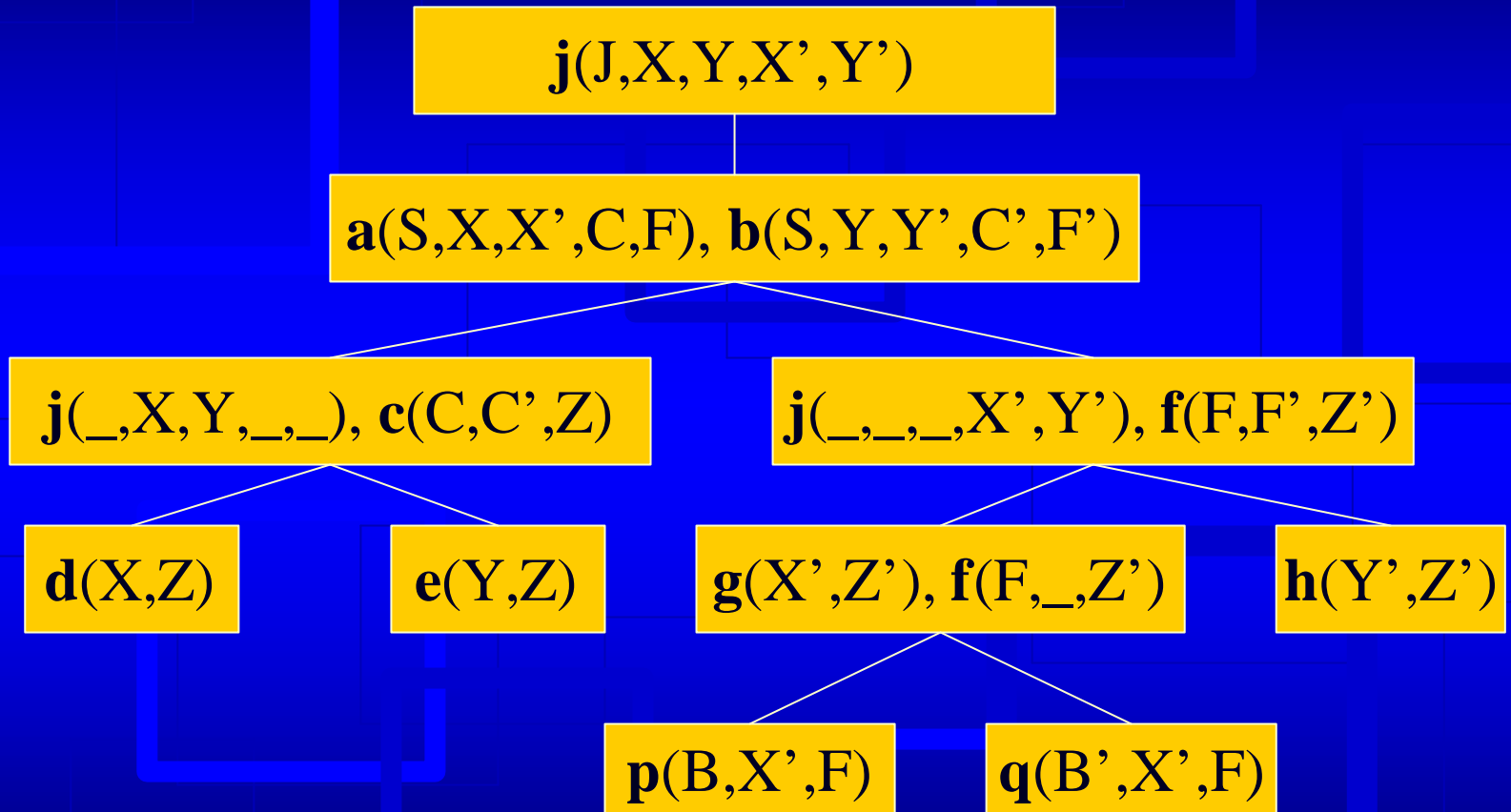


Despite its appearance,
this query is nearly acyclic



It can be evaluated in $O(m \cdot n^2 \cdot \log n)$

Hypertrees and Query Plans



It is just a partial specification!

A quantitative extension for HYPERTREE

- A new method, which exploits quantitative and structural information
- A cost model for evaluating plans determined by hypertree decompositions
- An algorithm for computing optimal bounded hypertree decompositions:
 - find the NF hypertree decomposition of width at most k such that its associated cost is the minimum over all k -bounded NF hypertree decompositions

A cost model for Hypertree

The cost of a **subtree** rooted at p

$$\begin{aligned}
 \text{cost}(T_p) &\triangleq \text{cost}(p) \\
 &+ \sum_{c \in \text{children}(p)} \text{cost}(T_c) \\
 &+ \sum_{c \in \text{children}(p): c(c) - c(p) \neq \emptyset} \mathbf{p}_{c(p)}(S_c) \\
 &+ \text{cost}(E(S_p))
 \end{aligned}$$

$$S_p \triangleq \begin{cases} J_p, & \forall p \in \text{leaves}(T_p) \\ J_p \quad \bowtie \quad \mathbf{p}_{c(p)}(S_c), & \text{otherwise} \\ & c \in \text{children}(p) \end{cases}$$

Example

Consider again the conjunctive query:

$$\begin{aligned} \mathit{ans} \leftarrow & a(S, X, X_p, C, F) \wedge b(S, Y, Y_p, C_p, F_p) \wedge c(C, C_p, Z) \wedge \\ & \wedge d(X, Z) \wedge e(Y, Z) \wedge f(F, F_p, Z_p) \wedge \\ & \wedge g(X_p, Z_p) \wedge h(Y_p, Z_p) \wedge j(J, X, Y, X_p, Y_p). \end{aligned}$$

Quantitative Information

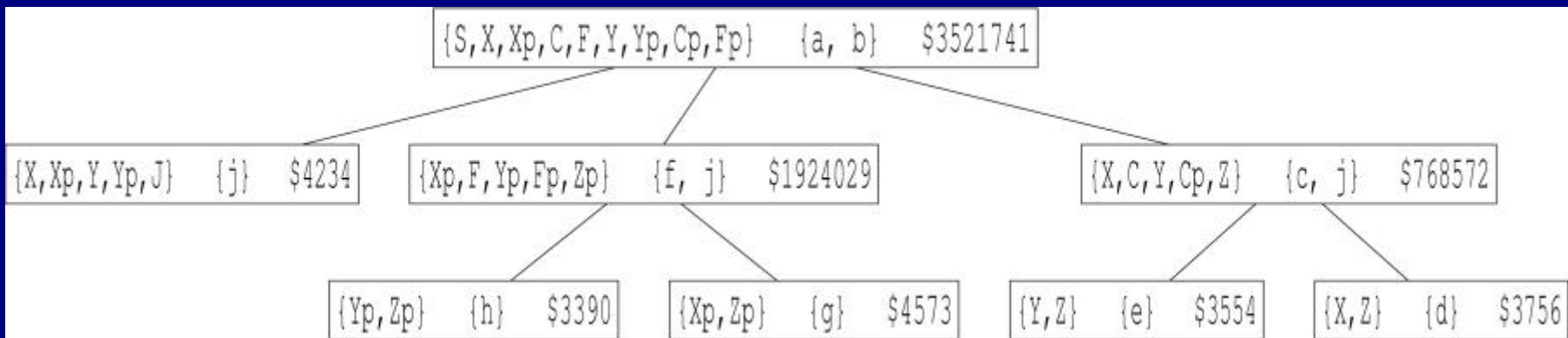
<ATOM>	<CARDINALITY>
a(S,X,Xp,C,F)	4606
b(S,Y,Yp,Cp,Fp)	2808
c(C,Cp,Z)	1748
d(X,Z)	3756
e(Y,Z)	3554
f(F,Fp,Zp)	2892
g(Xp,Zp)	4573
h(Yp,Zp)	3390
j(J,X,Y,Xp,Yp)	4234

**Cardinality of relations in
the Database**

<ATOM>	<VARIABLE>	<SELECTIVITY>
a(S,X,Xp,C,F)	S	14
a(S,X,Xp,C,F)	X	24
a(S,X,Xp,C,F)	Xp	16
a(S,X,Xp,C,F)	C	21
a(S,X,Xp,C,F)	F	15
b(S,Y,Yp,Cp,Fp)	S	17
b(S,Y,Yp,Cp,Fp)	Y	5
b(S,Y,Yp,Cp,Fp)	Yp	12
b(S,Y,Yp,Cp,Fp)	Cp	20
b(S,Y,Yp,Cp,Fp)	Fp	7
c(C,Cp,Z)	C	18
c(C,Cp,Z)	Cp	7
c(C,Cp,Z)	Z	19
d(X,Z)	X	18
d(X,Z)	Z	7
e(Y,Z)	Y	21
e(Y,Z)	Z	13

Attribute selectivity

Example: $k = 2$



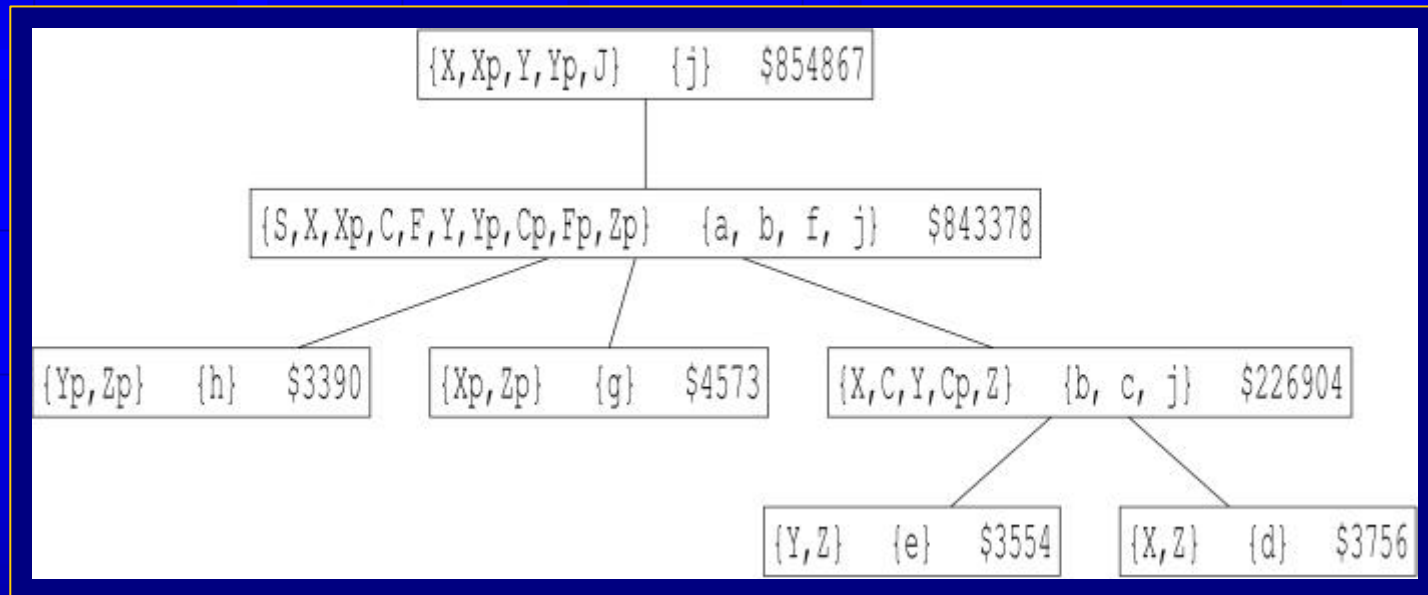
$k = 2$



3521741

width = 2

Example: $k = 4$



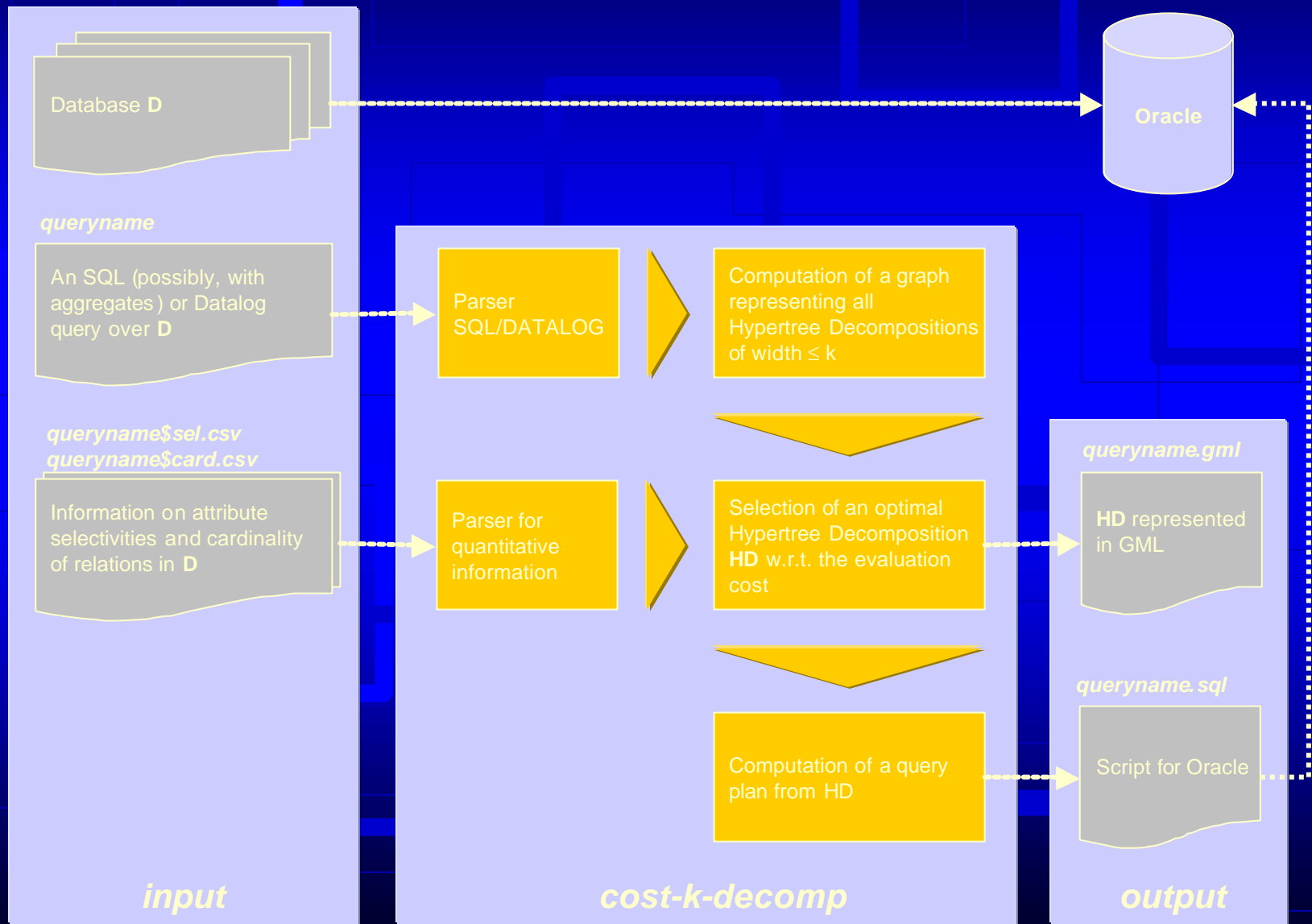
$k = 4$ \longrightarrow 854867 $width = 4$

In Summary

The prototype implements a new method for computing query plans based on hypertree decomposition:

- Exploits quantitative information on the data
- Exploits the structure of the query
- Guarantees a polynomial-time upper bound
- Quality/Efficiency Ratio may be tuned through the parameter k
- The algorithm is parallelizable
- It is not Fixed-Parameter Tractable

The Prototype Architecture



Example: Input Query

DATALOG

```
out(X,Y):-  
a(S,X,XP,C,F),  
b(S,Y,YP,CP,FP),  
c(C,CP,Z),  
d(X,Z),  
e(Y,Z),  
f(F,FP,ZP),  
g(XP,ZP),  
h(YP,ZP),  
j(J,X,Y,XP,YP).
```

SQL

```
SELECT a.X,b.Y  
FROM a,b,c,d,e,f,g,h,j  
WHERE a.S=b.S AND a.C=c.C AND ...
```

hdq5.txt

Example: Input Database

The screenshot shows the TOAD Schema Browser interface for the database ALFELLI@MAINDB. The main window displays a table with the following data:

S	X	Xp	C	F
i	v	l	j	n
k	d	o	j	c
d	l	j	g	a
i	u	k	p	j
i	c	e	k	k
j	b	l	a	j
e	l	b	t	j
k	u	k	h	e
d	f	g	e	i
g	h	c	h	m
n	g	e	k	g
n	u	n	k	d
l	x	l	k	k
h	p	e	j	o
e	v	f	o	n
g	c	n	r	p
j	d	p	a	b
a	l	f	k	f
j	j	j	u	a
e	b	b	d	l
c	d	k	o	o
g	o	o	h	j
n	q	k	l	g
m	g	d	m	e

The interface includes a menu bar (File, Edit, Grid, SQL-Window, Create, Database, Tools, View, DBA, Debug, Window, Help), a toolbar with various icons, and a schema browser tree on the left showing the table structure. The status bar at the bottom indicates 'Cnt: 9', 'ALFELLI@MAINDB', and 'Commit is OFF'.

Running the prototype

C:\ Prompt dei comandi

```
C:\hypertree>costk1decomp hdq5.txt 2_
```


Some output information

C:\ Prompt dei comandi

```
C:\hypertree>costk1decomp hdq5.txt 2
```

```
query file      = hdq5.txt  
upper bound    = 2  
representation = non verbose
```

```
running ... elapsed time (sec): 0.01
```

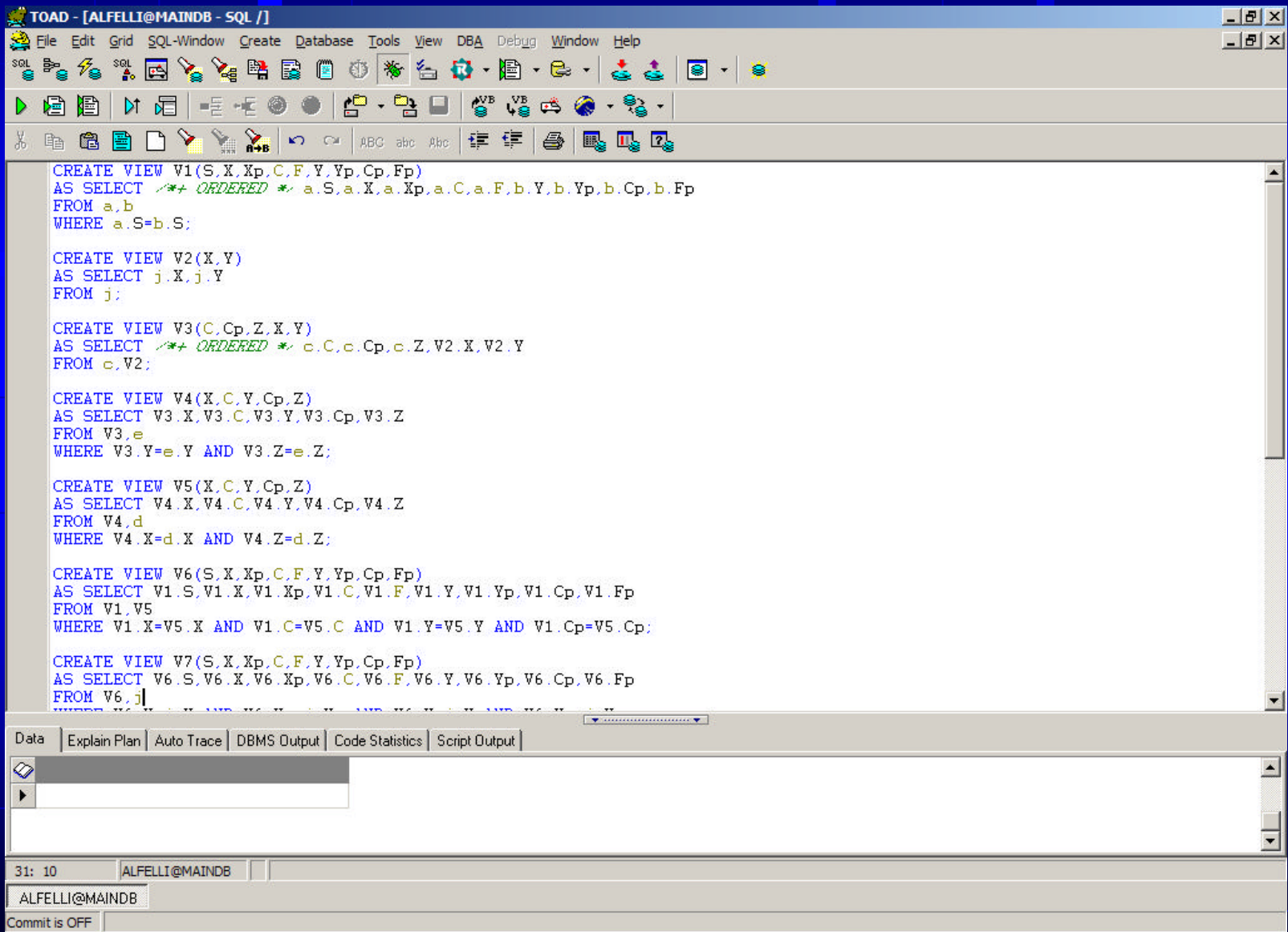
```
>> query atoms      = 9  
>> query variables  = 12
```

```
>> OPTIMAL COST BASED HYPERTREE DECOMPOSITION FOUND!
```

```
>> cost      = 3521741  
>> width    = 2  
>> depth    = 3  
>> vertices = 8  
>> OUTPUT   = hdq5.txt.gml, hdq5.txt.sql
```

```
C:\hypertree>
```

A script for Oracle (with hints)



The screenshot shows the TOAD interface with a SQL script containing several CREATE VIEW statements. The script uses the HINTS_ENABLED hint to force Oracle to use a specific execution plan for each view. The views are defined as follows:

```
CREATE VIEW V1(S, X, Xp, C, F, Y, Yp, Cp, Fp)
AS SELECT /*+ ORDERED */ a.S, a.X, a.Xp, a.C, a.F, b.Y, b.Yp, b.Cp, b.Fp
FROM a, b
WHERE a.S=b.S;

CREATE VIEW V2(X, Y)
AS SELECT j.X, j.Y
FROM j;

CREATE VIEW V3(C, Cp, Z, X, Y)
AS SELECT /*+ ORDERED */ c.C, c.Cp, c.Z, V2.X, V2.Y
FROM c, V2;

CREATE VIEW V4(X, C, Y, Cp, Z)
AS SELECT V3.X, V3.C, V3.Y, V3.Cp, V3.Z
FROM V3, e
WHERE V3.Y=e.Y AND V3.Z=e.Z;

CREATE VIEW V5(X, C, Y, Cp, Z)
AS SELECT V4.X, V4.C, V4.Y, V4.Cp, V4.Z
FROM V4, d
WHERE V4.X=d.X AND V4.Z=d.Z;

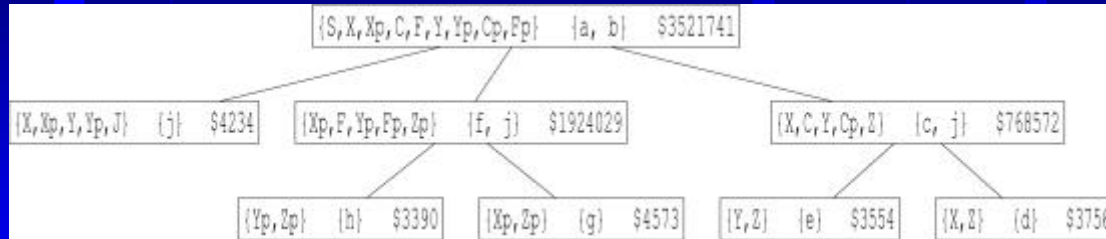
CREATE VIEW V6(S, X, Xp, C, F, Y, Yp, Cp, Fp)
AS SELECT V1.S, V1.X, V1.Xp, V1.C, V1.F, V1.Y, V1.Yp, V1.Cp, V1.Fp
FROM V1, V5
WHERE V1.X=V5.X AND V1.C=V5.C AND V1.Y=V5.Y AND V1.Cp=V5.Cp;

CREATE VIEW V7(S, X, Xp, C, F, Y, Yp, Cp, Fp)
AS SELECT V6.S, V6.X, V6.Xp, V6.C, V6.F, V6.Y, V6.Yp, V6.Cp, V6.Fp
FROM V6, j
```

The interface also shows a toolbar with various icons, a status bar at the bottom indicating '31: 10 ALFELLI@MAINDB', and a 'Commit is OFF' indicator.

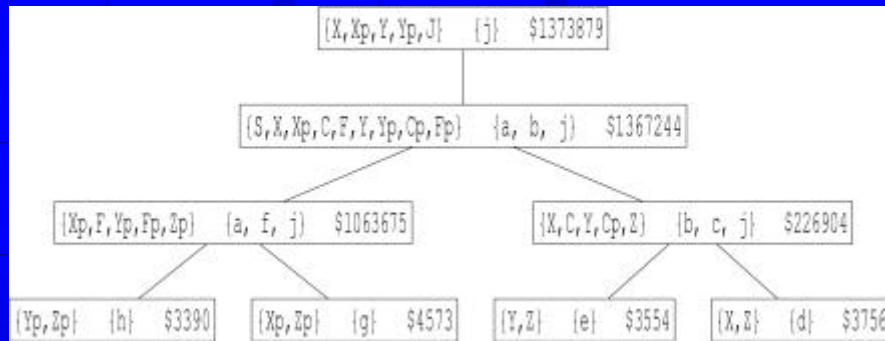
Some preliminary results

k = 2



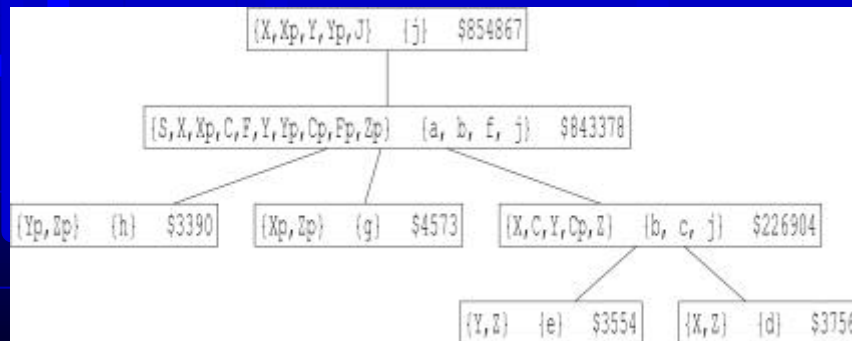
0.6

k = 3



0.04

k = 4 ...



0.02