大数据导论
Introduction to Big Data

关联规则挖掘

叶允明
计算机科学与技术学院
哈尔滨工业大学（深圳）

目录

(1) 关联规则是什么?

(2) 关联规则挖掘的两阶段算法框架

(3) 频繁项集生成方法

(4) 关联规则生成及相关性评价

(5) 挖掘多维量化关联规则

主要参考资料

- Jiawei Han, MIcheline Kamber, Jian Pei著；范明，孟小峰等译. 数据挖掘：概念与技术. 机械工业出版社, ISBN: 9787111391401, 2012.
- 第6章 挖掘频繁模式、关联和相关性：基本概念和方法

关联规则是什么？

什么是关联规则（Association Rule）？

牛奶 → 面包

尿布 → 啤酒

{牛奶, 面包} → {啤酒，尿布}

X → Y

TID（事务ID）	商品列表
T001	牛奶、啤酒、尿布
T002	鸡蛋、牛奶、面包、啤酒、尿布
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	牛奶、面包、啤酒、尿布

age(“25-35”) ∧ buy(“华为手机”) ⟶ buy(“格力空调”)

关联规则有什么用?

- 零售行业：优化货架
- 电商行业：商品推荐
- 库存优化
- ……

关联规则的形式化定义

- 基本概念
  - 事务(transaction)
  - 项（item）
  - **项集(item set)**
    - k-项集
  - **关联规则：**
    - 形如 $X \rightarrow Y$ 的**蕴涵表达式**
      - X 和 Y 为非空集合
      - X 和 Y 是**不相交**的两个项集

TID（事务ID）	商品列表
T001	牛奶、啤酒、尿布
T002	鸡蛋、牛奶、面包、啤酒、尿布
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	牛奶、面包、啤酒、尿布

TID（事务ID）	商品列表
T001	尿布、牛奶、啤酒
T002	尿布、鸡蛋、牛奶、面包、啤酒
T003	鸡蛋、牛奶、面包
T004	面包、啤酒
T005	尿布、牛奶、面包、啤酒

TID（事务ID）	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

$X \rightarrow Y$很多！

如何衡量关联规则的“质量”？

$X \rightarrow Y$

- **项集的支持度计数：**
  - ✓ 在数据库中包含项集W的事务个数
  - ✓ $\sigma(W) = |\{t_i | W \sqsubseteq t_i, t_i \in \mathrm{T}\}|$
- **支持度（support）**
  - ✓ 给定数据集中X和Y的共现频度
  - ✓ 令 $W = X \cup Y$
  - ✓ $s(X \rightarrow Y) = \frac{\sigma(W)}{|T|}$
- **置信度（confidence）**
  - ✓ 在包含X的事务子集中Y出现的频繁程度
  - ✓ $c(X \rightarrow Y) = \frac{\sigma(W)}{\sigma(X)}$

TID（事务ID）	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

R_1：I5 → I1

$$s(R_1) = \frac{\sigma(\{I5, I1\})}{|T|} = \frac{3}{5} = 0.6$$

$$c(R_1) = \frac{\sigma(\{I5, I1\})}{\sigma(\{I5\})} = \frac{3}{4} = 0.75$$

强关联规则（strong association rule）

- 定义
  - 最小支持度阈值：*min-sup*，　最小置信度阈值：*min-conf*
  - 关联规则*R*是强关联规则，当且仅当：

$$(1).\ s(R) \geq min\text{-}sup \qquad (2).\ c(R) \geq min\text{-}conf$$

- **关联规则挖掘任务：找到所有强关联规则！**

关联规则挖掘的两阶段算法框架

关联规则挖掘任务

- 输入：
  - 事务数据库、
  - 阈值参数：*min-sup* ， *min-conf*
- 输出：
  - 所有满足*min-sup* 、*min-conf* 的强关联规则
  - 包括每条强关联规则的支持度和置信度取值

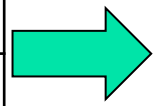
TID（事务ID）	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

最简单的关联规则挖掘算法——枚举法

- 假设：*min-sup* = 0.6，*min-conf* = 0.8
- 枚举法：
  - Step 1: 列出全部可能的关联规则
  - Step 2: 计算每条关联规则的支持度和置信度
  - Step 3: 筛选出满足min-sup和min-conf条件的规则

TID（事务ID）	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}



关联规则	支持度	置信度
{I1}→{I2}	0.2	0.33
{I2}→{I1}	0.2	0.5
{I1}→{I3}	0.6	1.0
{I3}→{I1}	0.6	0.75
……	…	…
{I1, I2}→{I3}	0.2	1.0
{I3}→{I1, I2}	0.2	0.25
……	…	…
{I1, I2, I3}→{I4}	0.2	1.0
{I4}→{I1, I2, I3}	0.2	0.25
……	…	…
{I1, I2, I3, I4}→{I5}	0.2	1.0
{I5}→{I1, I2, I3, I4}	0.2	0.25
……	…	…

min-sup=0.6

min-conf=0.8

关联规则	支持度	置信度
{I1}→{I3}	0.6	1.0
{I1}→{I5}	0.6	1.0
{I5}→{I3}	0.6	1.0
{I1}→{I3, I5}	0.6	1.0
{I3, I5}→{I1}	0.6	1.0
{I1, I5}→{I3}	0.6	1.0
{I1, I3}→{I5}	0.6	1.0

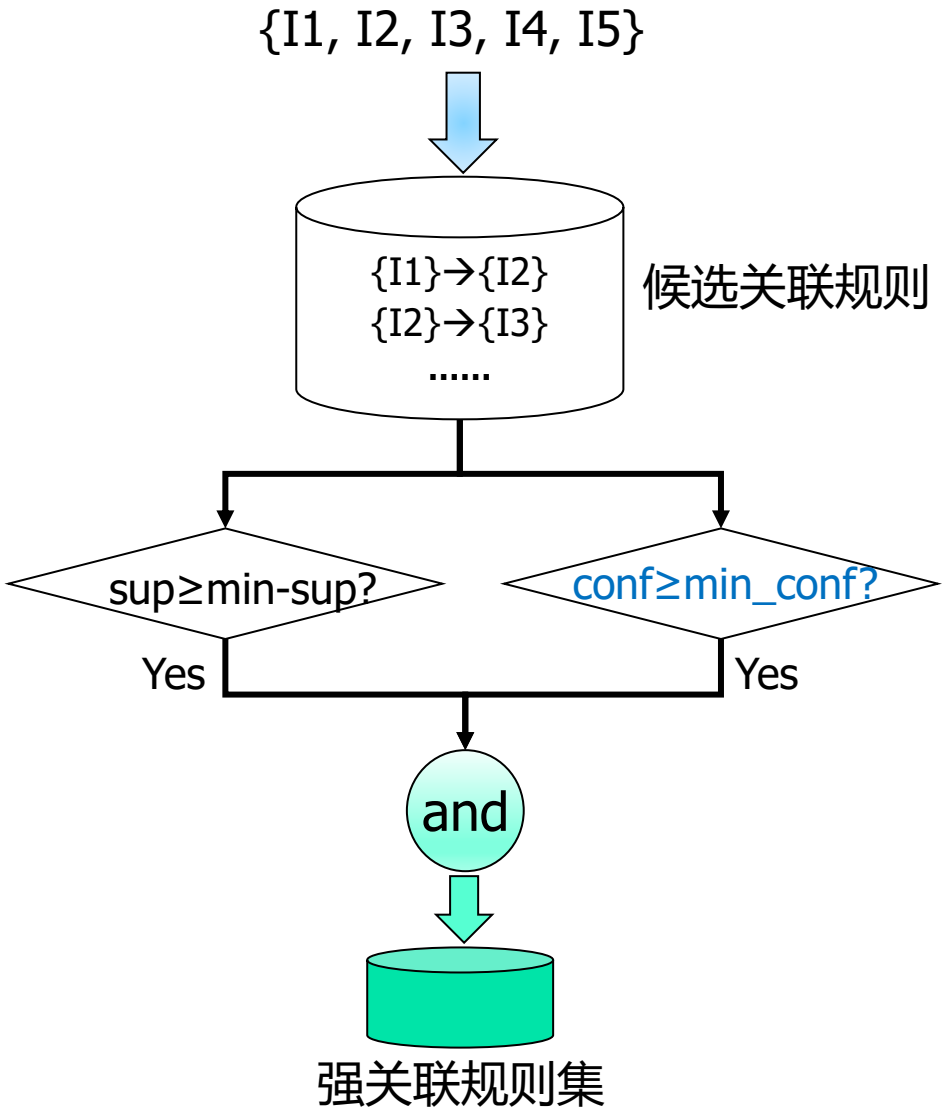
枚举法的特点分析

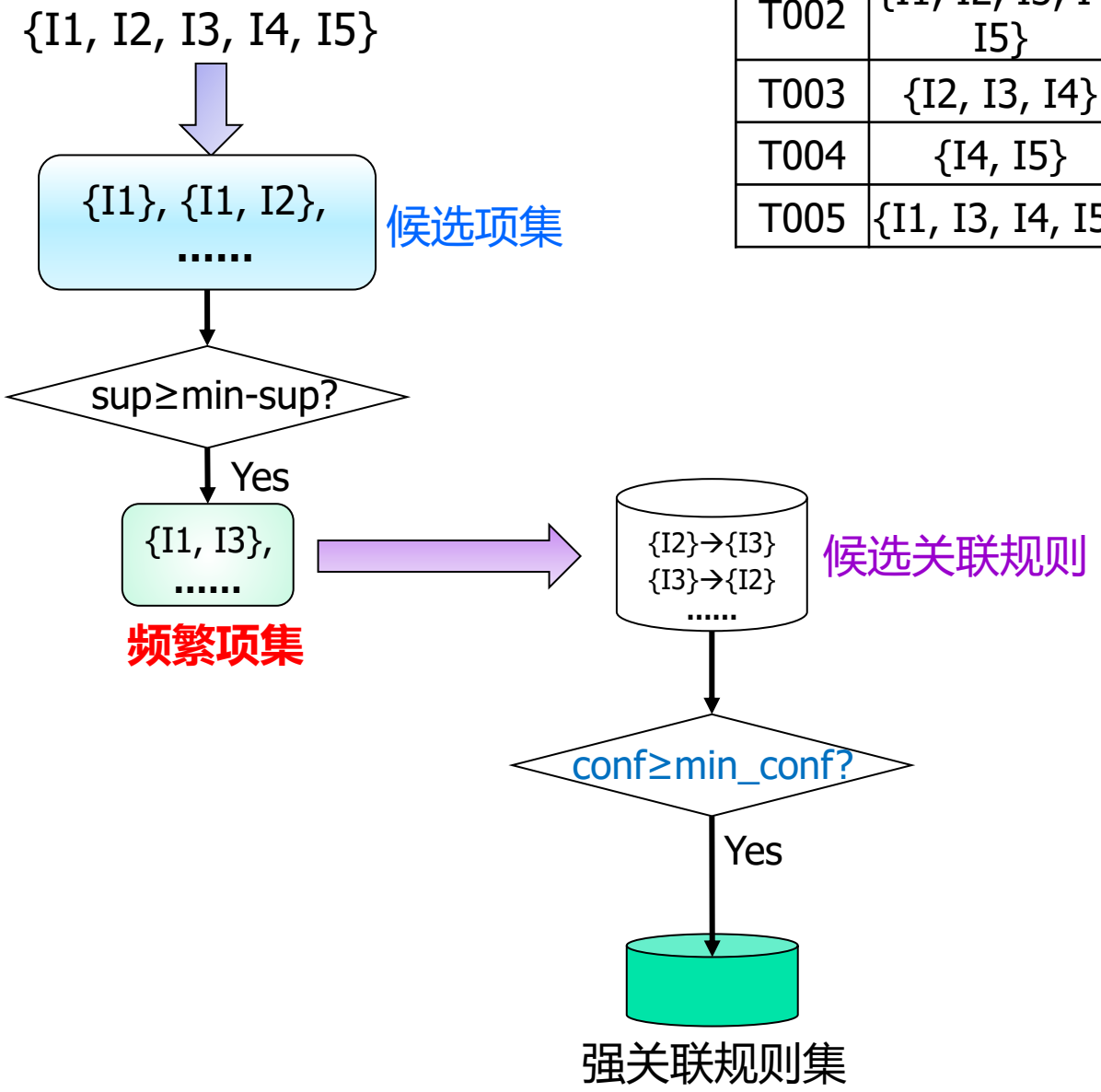
- 优点：实现简单，容易理解
- 缺点：待计算的关联规则数量随数据集增大呈指数增长
  - 包含n个项的数据集可提取的关联规则总数M为：$M = 3^n - 2^{n+1} + 1$

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

n=5, 可产生180条关联规则!

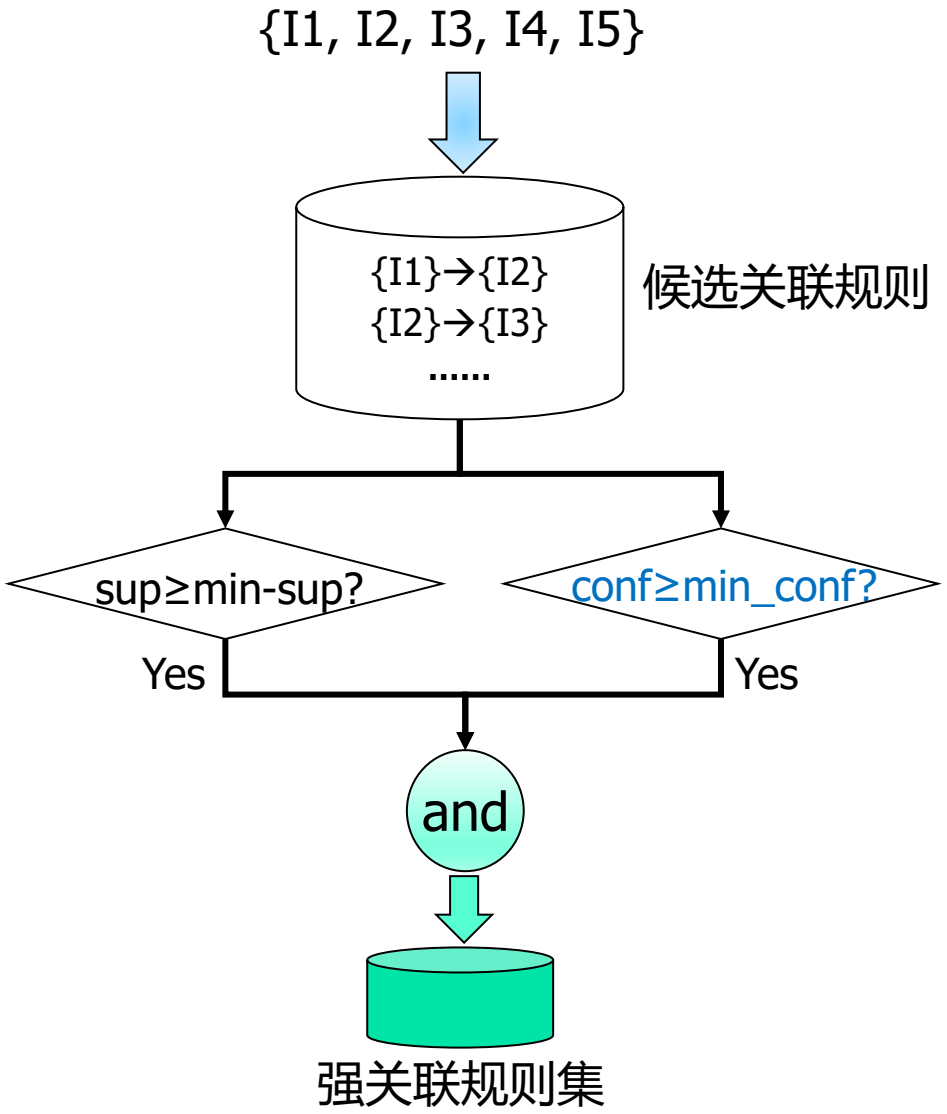
- 改进思路：从单步筛选变为“**两阶段**”筛选

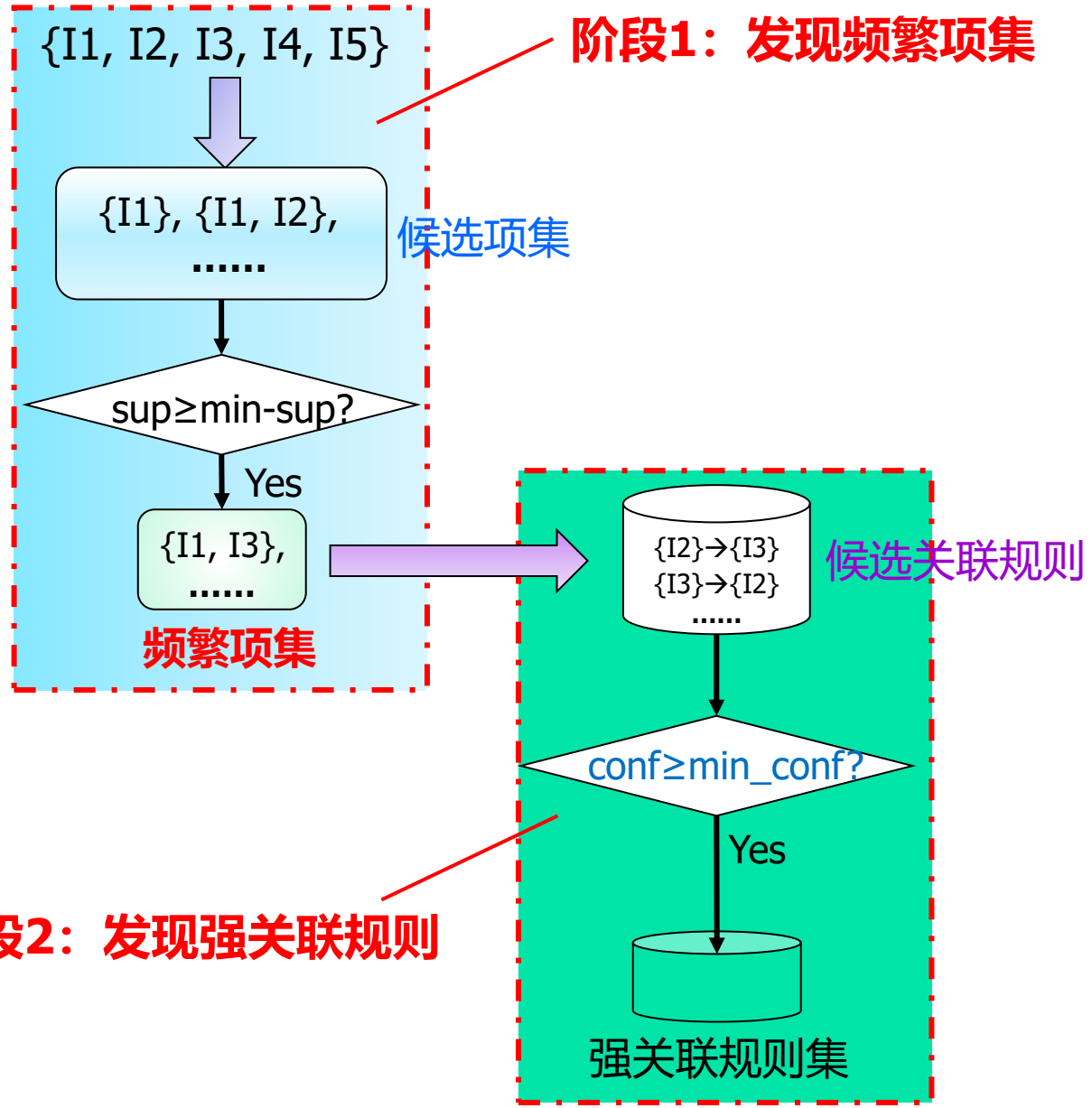




TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

关联规则挖掘的两阶段算法框架





阶段1：发现频繁项集

min-sup = 0.6

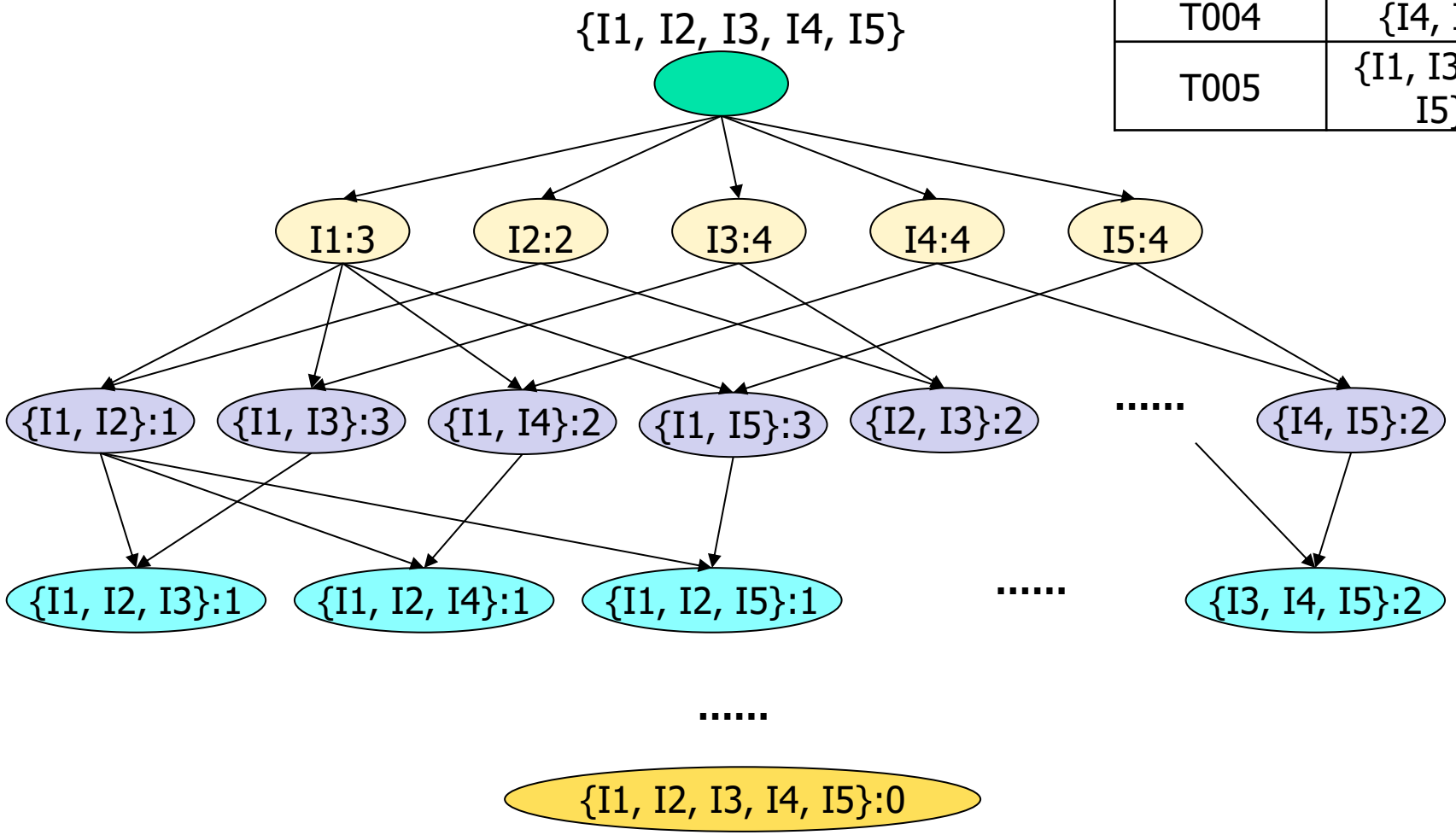
TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

{I1, I2, I3, I4, I5}

{I1}, {I1, I2}, ……

候选项集

频繁项集	支持度计数
{I1}	3
{I3}	4
{I4}	4
{I5}	4
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3



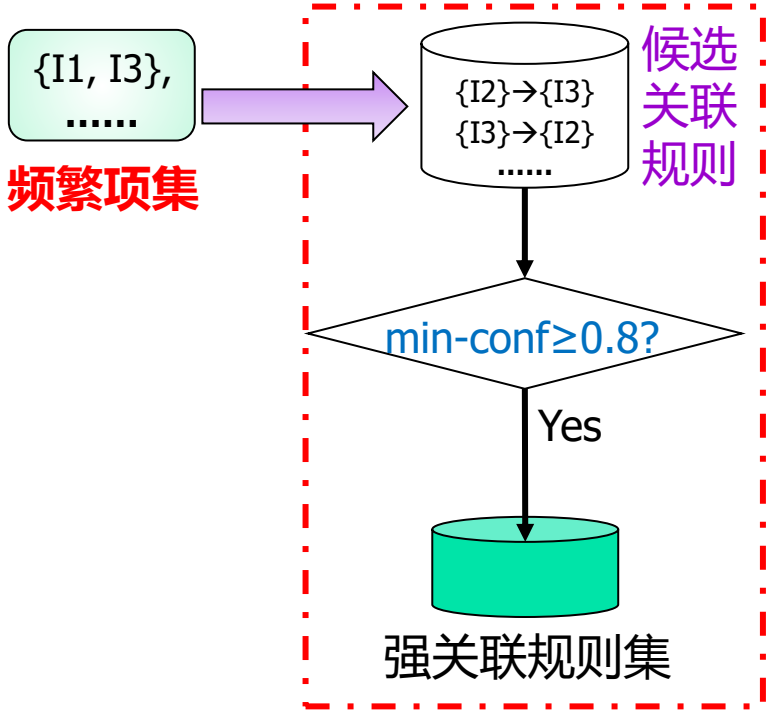
阶段2：发现强关联规则

min-conf = 0.8

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

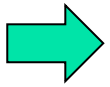
频繁项集	支持度计数
{I1}	3
{I3}	4
{I4}	4
{I5}	4
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3

频繁项集	支持度计数
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3

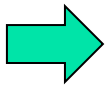


两阶段算法与单步枚举法的计算量比较

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}



频繁项集	支持度计数
{I1}	3
{I3}	4
{I4}	4
{I5}	4
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3
{I1, I3, I5}	3



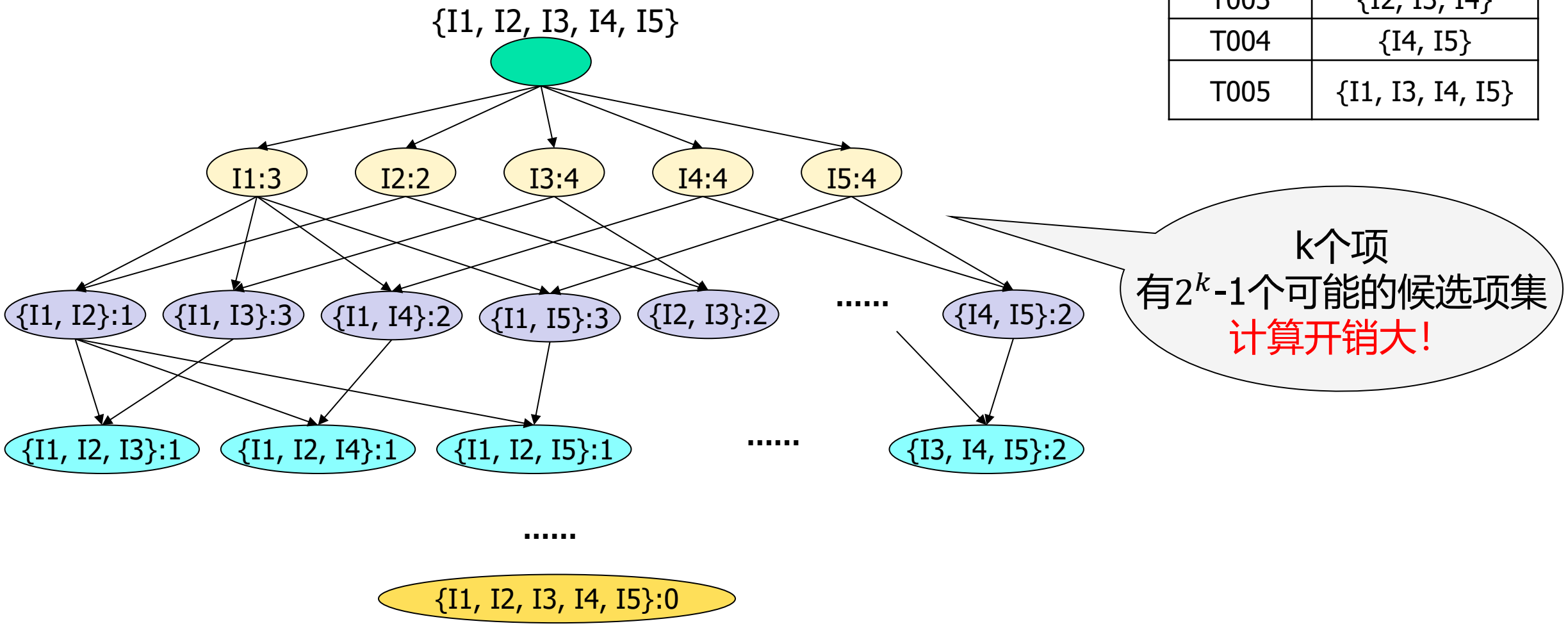
关联规则	置信度
{I1}→{I3}	1.0
{I1}→{I5}	1.0
{I5}→{I3}	1.0
{I1}→{I3, I5}	1.0
{I3, I5}→{I1}	1.0
{I1, I5}→{I3}	1.0
{I1, I3}→{I5}	1.0

- 阶段一：频繁项集发现
  - 计算31个项集的支持度
  - 筛选得到10个频繁项集
- 阶段二：关联规则生成
  - 计算16条候选关联规则的置信度
  - 筛选得到7条强关联规则
- 单步枚举法需要计算180个关联规则的支持度和置信度
- **计算量对比：360 → 47**

频繁项集发现：Apriori算法

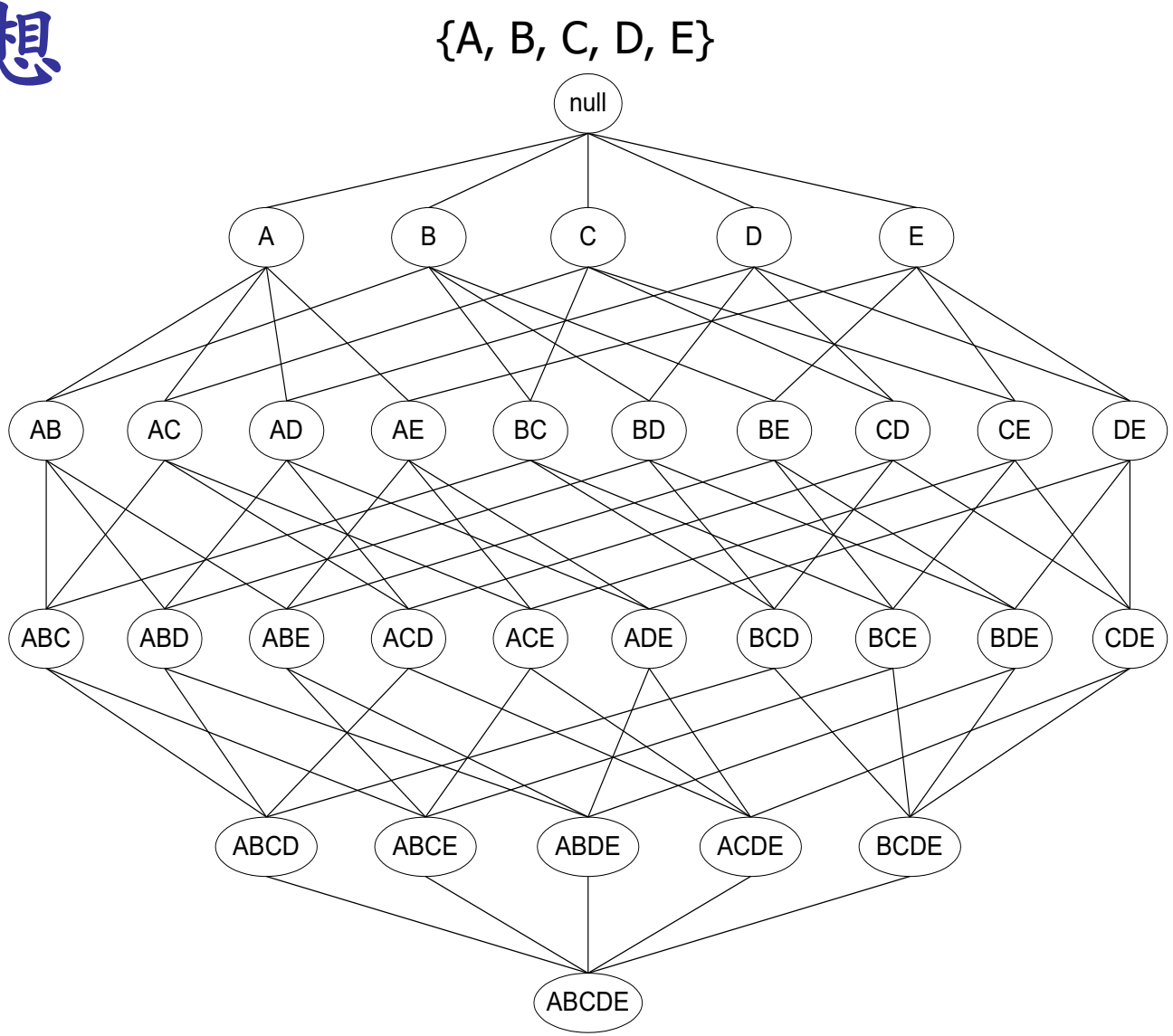
用枚举法发现频繁项集的问题

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}



Aprior算法的基本思想

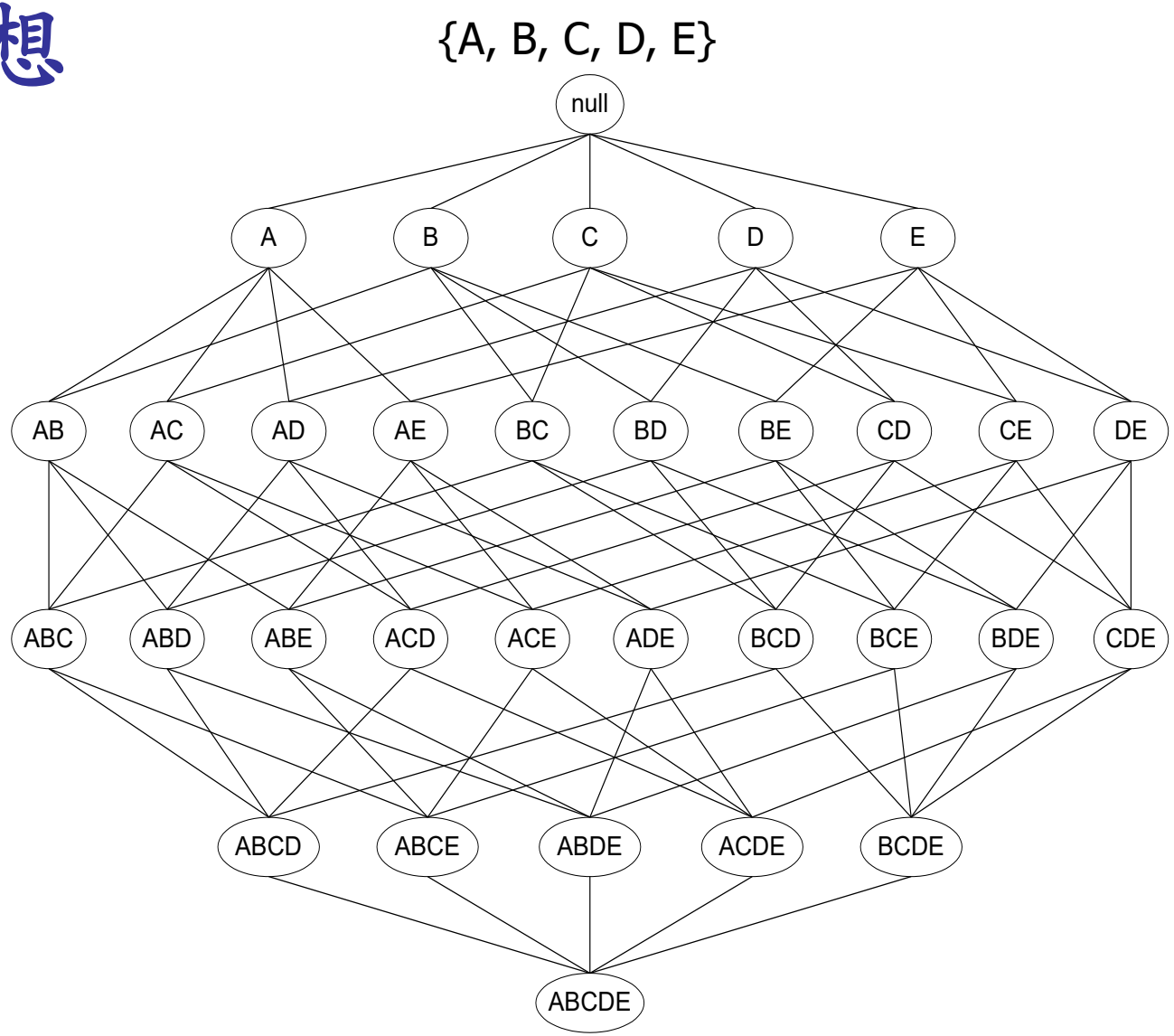
- 减少候选项集的数量!
- 剪枝!



Aprior算法的基本思想

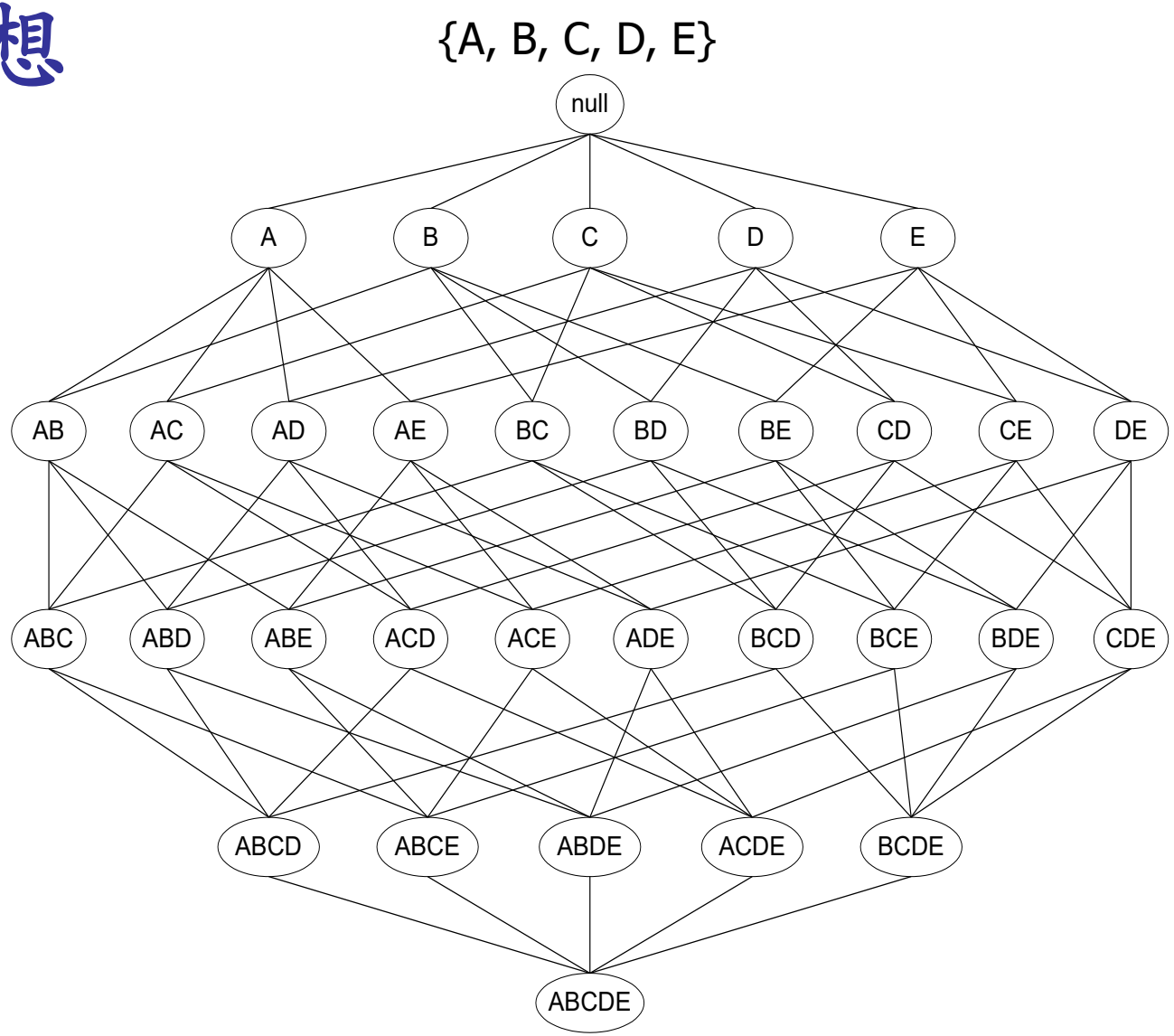
- **先验原理**
  - 频繁项集的子集一定也是频繁的！

- **反单调性**
  - 一个项集的支持度不超过它的子集的支持度
  - 如果对于项集Y的每个真子集X（即X⊏Y），有f(Y)≤f(X)，那么称度量f具有反单调性。



Aprior算法的基本思想

- 基本思想:
  - 逐层搜索迭代
  - 用上一轮迭代得到的k项集来探索下一轮迭代的（k+1）项集



Aprior算法的主要步骤

(1). 扫描数据库，得到所有频繁1项集

(2). 用k-频繁项集生成(k+1)-候选项集

(3). 扫描数据库计算每个(k+1)-候选项集的支持度，如果超过*min-sup*则为(k+1)-频繁项集

(4). 重复(2)、(3)直到无法生成新的候选项集

min-sup = 0.6

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

TID	商品ID列表
T001	{I1, I2, I3, I4, I5}
T002	{I1, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I1, I3, I5}
T005	{I4, I5}

1st scan

Itemset	sup
{I1}	3
{I2}	2
{I3}	4
{I4}	4
{I5}	4

C1

Itemset	sup
{I1}	3
{I3}	4
{I4}	4
{I5}	4

L1

Itemset
{I1, I3}
{I1, I4}
{I1, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C2

2nd scan

Itemset	sup
{I1, I3}	3
{I1, I4}	2
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3

C2

Database
min-sup=3

Itemset	sup
{I1, I3}	3
{I1, I5}	3
{I3, I4}	3
{I3, I5}	3
{I4, I5}	3

L2

Itemset
~~{I1, I3, I4}~~
{I1, I3, I5}
{I3, I4, I5}

C3

3rd scan

Itemset	sup
~~{I1, I3, I4}~~	~~2~~
{I1, I3, I5}	3
{I3, I4, I5}	2

C3

Itemset	sup
{I1, I3, I5}	3

L3

Aprior算法的特点分析

- 优点
  - 原理简单，易于实现
  - 适合于稀疏数据集中的频繁模式挖掘
- 缺点
  - 候选项集数量可能很大
    - ✓ 假设L_1有10^4项，则C_2将包含10^7项
    - ✓ 假设要挖掘L_{100}，需要产生$2^{100} \approx 10^{30}$个候选项
  - 重复扫描数据库
    - ✓ 每轮迭代对候选项集进行支持度计数时，都需要扫描一遍数据库，从而产生不可忽视的I/O开销

如何进一步提高频繁项集生成的效率?

- 减少候选项集的数量(M)
- 减少需要扫描的事务(记录)数量(N)
- 减少候选项集与事务的比较次数 (NM)

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

改进Aprior算法的两个思路

- 压缩迭代时扫描的事务数
  - 如果一个事务不包含任何一个频繁k-项集，那么这个事务也一定不包含任何一个频繁（k+1）-项集
- 基于散列(hashing)优化支持度计数
  - 如散列2-项集时，可以设散列函数：

$$h(x,y) = \left(f(x) * 10 + f(y)\right) mod\ 7$$

TID	商品ID列表
T001	{I1, I3, I5}
T002	{I1, I2, I3, I4, I5}
T003	{I2, I3, I4}
T004	{I4, I5}
T005	{I1, I3, I4, I5}

基于散列优化

TID	Items
T001	A, C, D
T002	B, C, E
T003	A, B, C, E
T004	B, E

min-sup=3

自定义映射：

Itemset	order
{A}	1
{B}	2
{C}	3
{D}	4
{E}	5

h({x y}) = {{order of x}*10 + {order of y}} mod 7

T001	{A, C}, {A, D}, {C, D}
T002	{B, C}, {B, E}, {C, E}
T003	{A, B}, {A, C}, {A, E}, {C, E}
T004	{B, E}

h({A, C})
=(1*10+3) mod 7
= 6

桶内容	{C, E} {C, E} {A, D}	~~{A, E}~~	~~{B, C}~~ ~~{B, C}~~		{B, E} {B, E} {B, E}	~~{A, B}~~	{A, C} {C, D} {A, C}
桶计数	3	1	2	0	3	1	3
h({x, y})	0	1	2	3	4	5	6

C1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L1

Itemset	sup
{B}	3
{C}	3
{E}	3

提取桶计数>min-sup的项

Itemset	sup
~~{C, E}~~	/
~~{A, D}~~	/
{B, E}	3
~~{A, C}~~	/
~~{C, D}~~	/

先验原理剪枝

根据支持度筛选 →

Itemset	sup
{B, E}	3