Параллельный алгоритм поиска частых наборов для многоядерных процессорных систем

М.Л. Цымблер

Научный семинар по информационным технологиям
Челябинск, ЮУрГУ
2 мая 2017 г.

Содержание

- Задача поиска частых наборов
- Алгоритм Apriori
- Алгоритм DIC (Dynamic Itemset Counting)
- Распараллеливание алгоритма DIC
- Вычислительные эксперименты

Примеры практических задач

- Супермаркет: какие товары часто покупают совместно?

- Поиск побочных эффектов лекарств

Поиск частых наборов: определения

- Объект (*item*).
- Набор (*itemset*) – конечное множество объектов.
- k-*набор* (k-*itemset*) – множество из $k \geq 1$ объектов.
- *Транзакция* – <TID, k-набор>.
- База данных транзакций – D.
- *Поддержка набора (itemset support count)* – количество транзакций, в которые входит набор.

D

TID	Набор
10	орешки, пиво, подгузники
20	кофе, пиво, подгузники
30	кофе, пиво, подгузники, хлеб
40	молоко, орешки, хлеб
50	кофе, молоко, орешки, подгузники, хлеб

- Пример:
  - *supp*({пиво})=3
  - *supp*({орешки,пиво})=1
  - *supp*({кофе,подгузники})=2
  - *supp*({пиво,подгузники})=3
  - *supp*({кофе,подгузники,хлеб})=2

Задача поиска частых наборов

- *Минимальная поддержка* $minsup$ – пороговое значение поддержки.
- *Частый* k*-набор* ($k \geq 1$) – k-набор с поддержкой не ниже $minsup$ (иначе набор *редкий*).
- L_k – множество всех частых k-наборов.
- Задача: для заданных D и $minsup$ найти все L_k ($k \geq 1$).

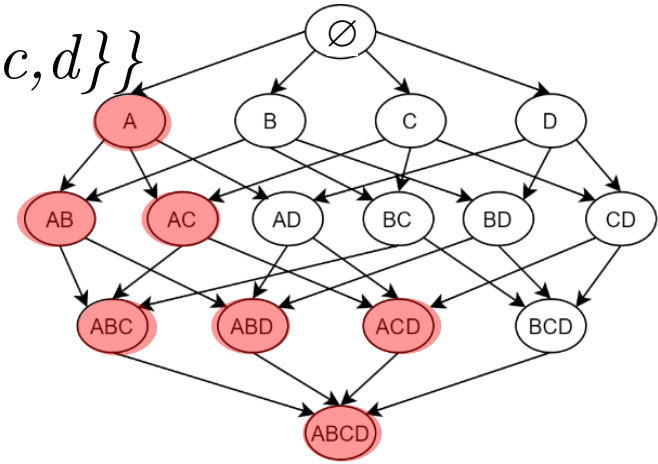
D

TID	Набор
10	орешки, пиво, подгузники
20	кофе, пиво, подгузники
30	кофе, пиво, подгузники, хлеб
40	молоко, орешки, хлеб
50	кофе, молоко, орешки, подгузники, хлеб

- При *minsup*=3
  - L_1={{орешки}: supp=3, {пиво}: supp=3, {подгузники}: supp=4, {хлеб}: supp=3, {кофе}: supp=3}
  - L_2={{пиво,подгузники}: supp=3}
- При *minsup*=2
  - L_1={{орешки}: supp=3, {пиво}: supp=3, {подгузники}: supp=4, {хлеб}: supp=3, {кофе}: supp=3, {молоко}: supp=2}
  - L_2={{кофе,пиво }: supp=2, {пиво,подгузники }: supp=3, …}
  - L_3=({кофе,подгузники,хлеб}: supp=2}

Алгоритм Apriori: идеи

- Последовательность шагов
  - k:=1; Найти L_1
  - Для всех k≥2
    - На основе L_{k-1} генерировать C_k, множество наборов-кандидатов в частые наборы
    - Отбросить заведомо редкие наборы из C_k
    - Найти L_k, сканируя D и вычисляя поддержку k-наборов из C_k
    - Закончить, если L_k=∅
- Генерация кандидатов: $JOIN$
  - L_3=$\{\{a,b,c\},\{a,b,d\},\{a,c,d\},\{a,c,e\},\{b,c,d\}\}$
  - C_4 =L_3 $JOIN$ L_3=$\{\{a,b,c,d\},\{a,c,d,e\}\}$
- Отбрасывание кандидатов: Apriori
  - Любое непустое подмножество частого набора должно быть частым набором.



Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

C_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2
I6	1

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
10	I6

C_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2
I6	1

L_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

C_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2
I6	1

L_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2

C_2

Набор
I1,I2
I1,I3
I1,I4
I1,I5
I2,I3
I2,I4
I2,I5
I3,I4
I3,I5
I4,I5

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

C_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2
I6	1

L_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2

C_2

Набор
I1,I2
I1,I3
I1,I4
I1,I5
I2,I3
I2,I4
I2,I5
I3,I4
I3,I5
I4,I5

C_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Алгоритм Apriori: пример

D, *minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

C_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2
I6	1

L_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2

C_2

Набор
I1,I2
I1,I3
I1,I4
I1,I5
I2,I3
I2,I4
I2,I5
I3,I4
I3,I5
I4,I5

C_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

L_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

C_3

Набор
I1,I2,I3
I1,I2,I5
I1,I3,I5
I2,I3,I4
I2,I3,I5
I2,I4,I5

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

C_3

Набор
I1,I2,I3
I1,I2,I5
I1,I3,I5
I2,I3,I4
I2,I3,I5
I2,I4,I5

C_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_2

Набор	SUPP
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

C_3

Набор
I1,I2,I3
I1,I2,I5
I1,I3,I5
I2,I3,I4
I2,I3,I5
I2,I4,I5

C_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

L_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

C_4

Набор
I1,I2,I3,I5

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

C_4

Набор
I1,I2,I3,I5

C_4

Набор
Ø

СТОП

Алгоритм Apriori: пример

***D**, minsup=2*

TID	Набор
10	I1, I2, I5
20	I2, I4
30	I2, I3
40	I1, I2, I4
50	I1, I3
60	I2, I3
70	I1, I3
80	I1, I2, I3, I5
90	I1, I2, I3
100	I6

L_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

C_4

Набор
I1,I2,I3,I5

C_4

Набор
Ø

СТОП

L_1

Набор	SUPP
I1	6
I2	7
I3	6
I4	2
I5	2

L_2

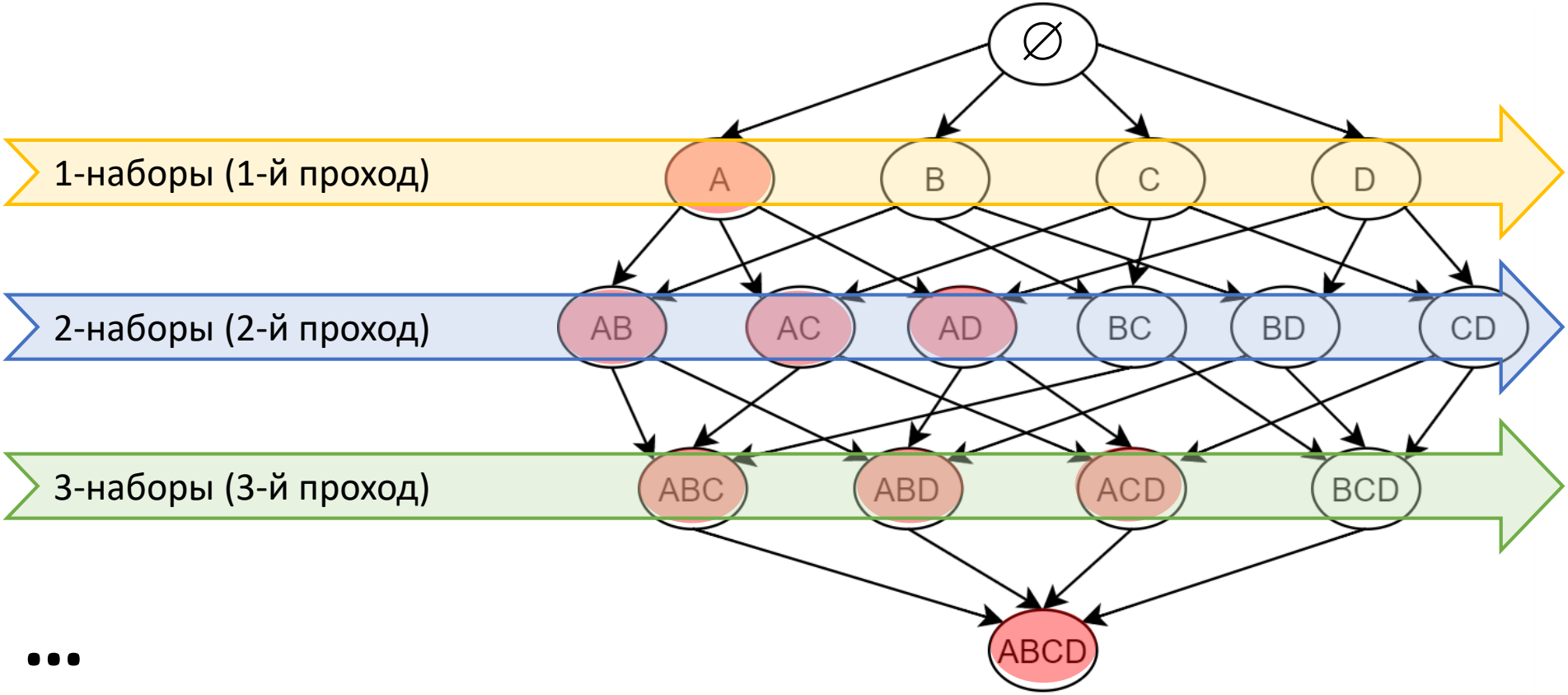
Набор	SUPP
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

L_3

Набор	SUPP
I1,I2,I3	2
I1,I2,I5	2

Алгоритм DIC (Dynamic Itemset Counting)

- Brin S., Motwani R., Ullman J.D., Tsur S. Dynamic Itemset Counting and Implication Rules for Market Basket Data. Proc. of the SIGMOD'1997, ACM Press, 1997, pp. 255–264.
- Основная идея – улучшить Apriori путем сокращения количества проходов по базе данных транзакций

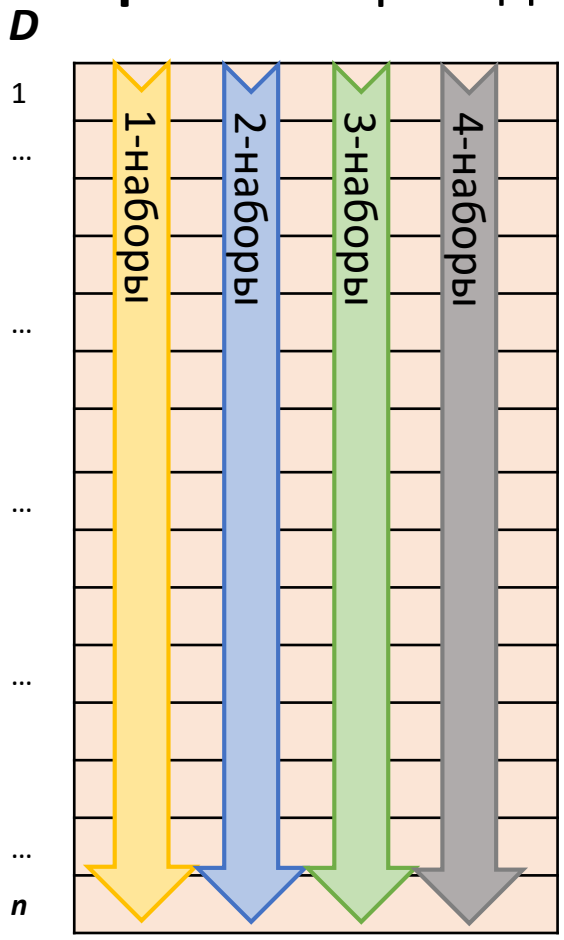


...

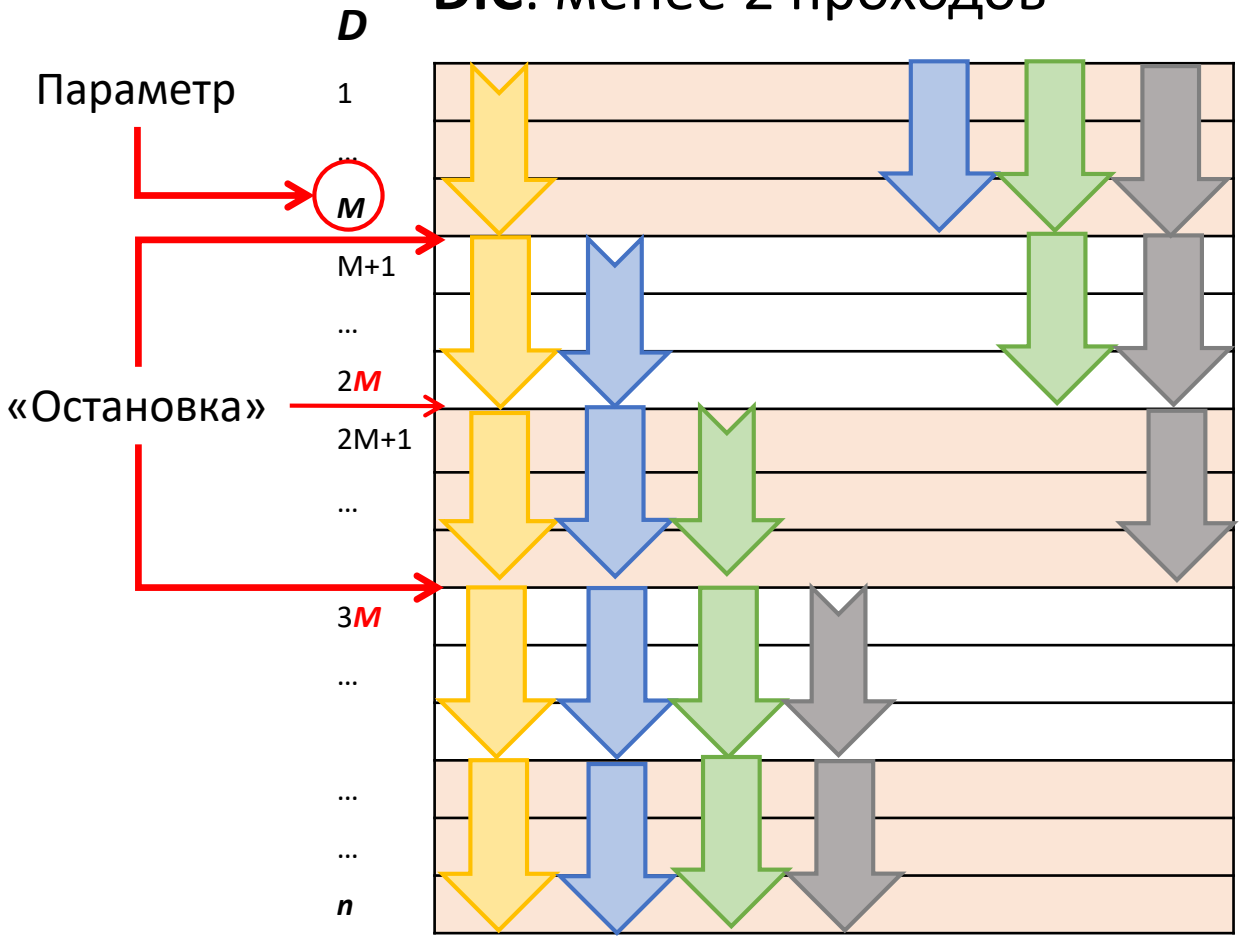
DIC: сокращение числа проходов

- Подсчет поддержки *(k+1)*-набора можно начать до окончания *k* прохода (если набор, возможно, частый)

Apriori: 4 прохода

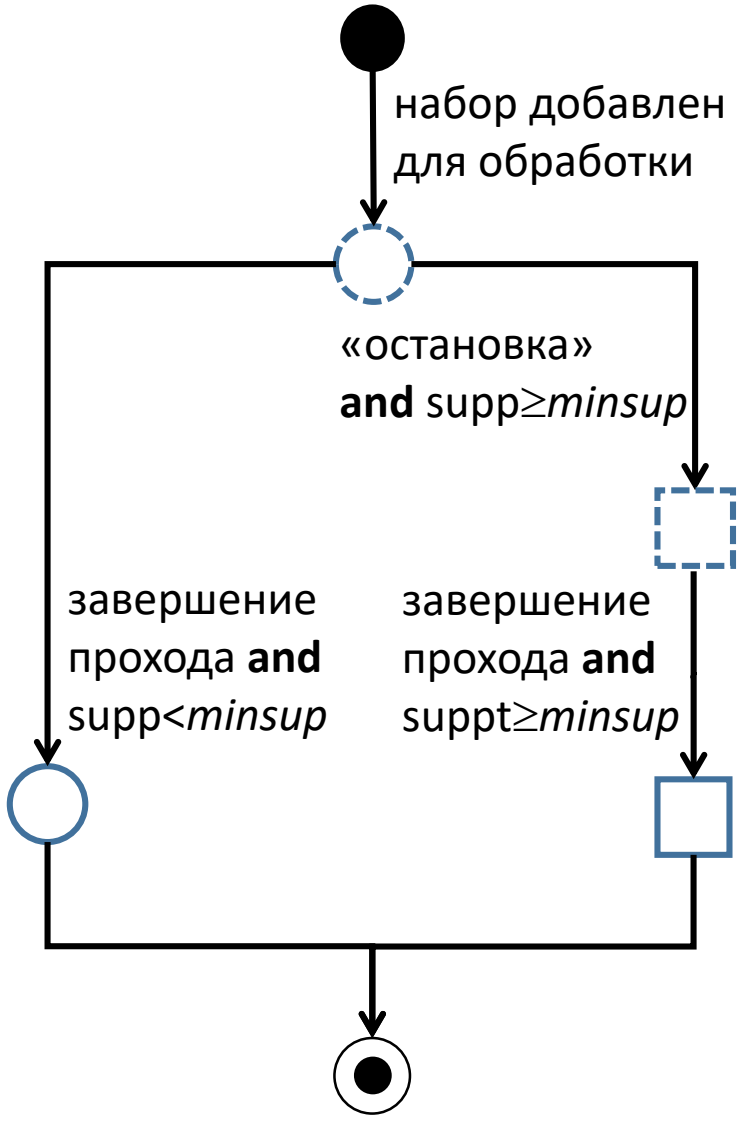


DIC: менее 2 проходов



DIC: разметка наборов

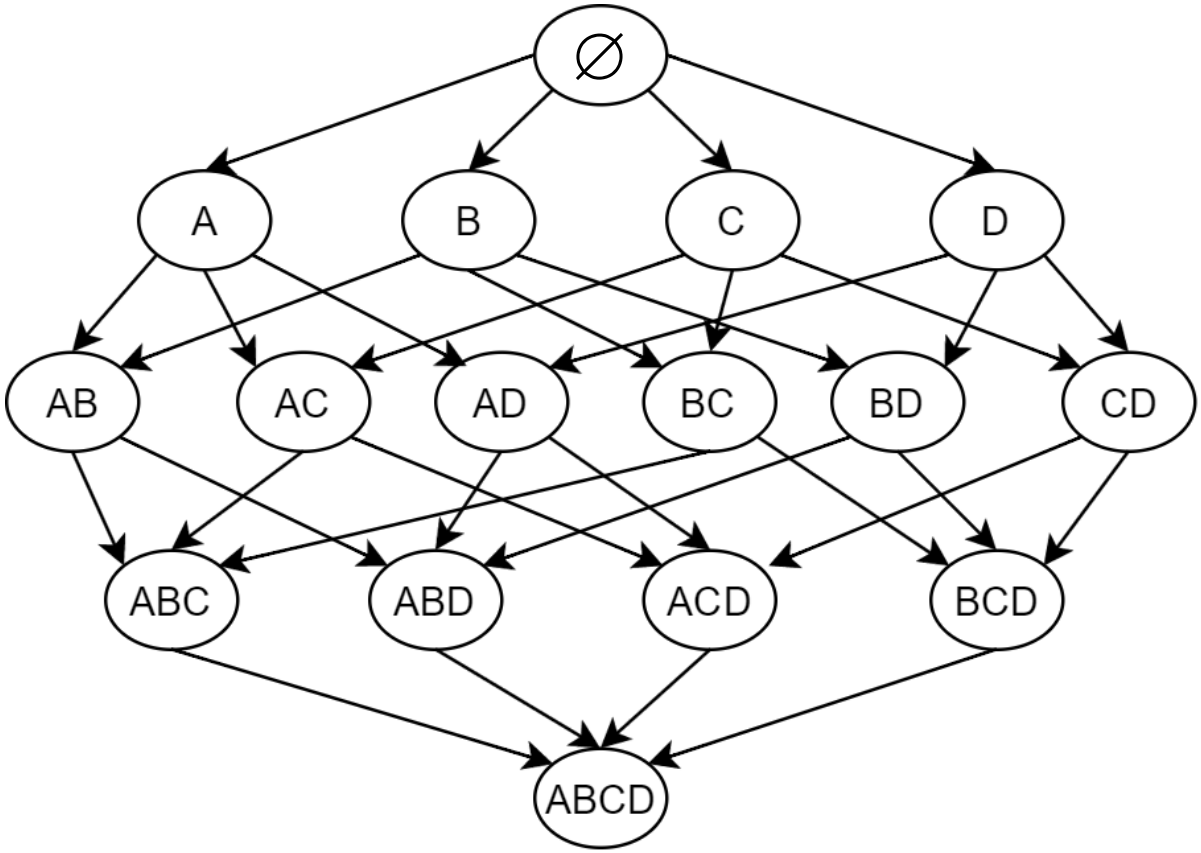
- **подтвержденные частые**
  - конечный статус, результат
  - подсчет поддержки не нужен
- **подтвержденные редкие**
  - конечный статус, не результат
  - подсчет поддержки не нужен
- **неподтвержденные частые**
  - промежуточный статус
  - нужен подсчет поддержки
- **неподтвержденные редкие**
  - начальный статус («кандидат»)
  - нужен подсчет поддержки



DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle ∅

Dashed Box ∅

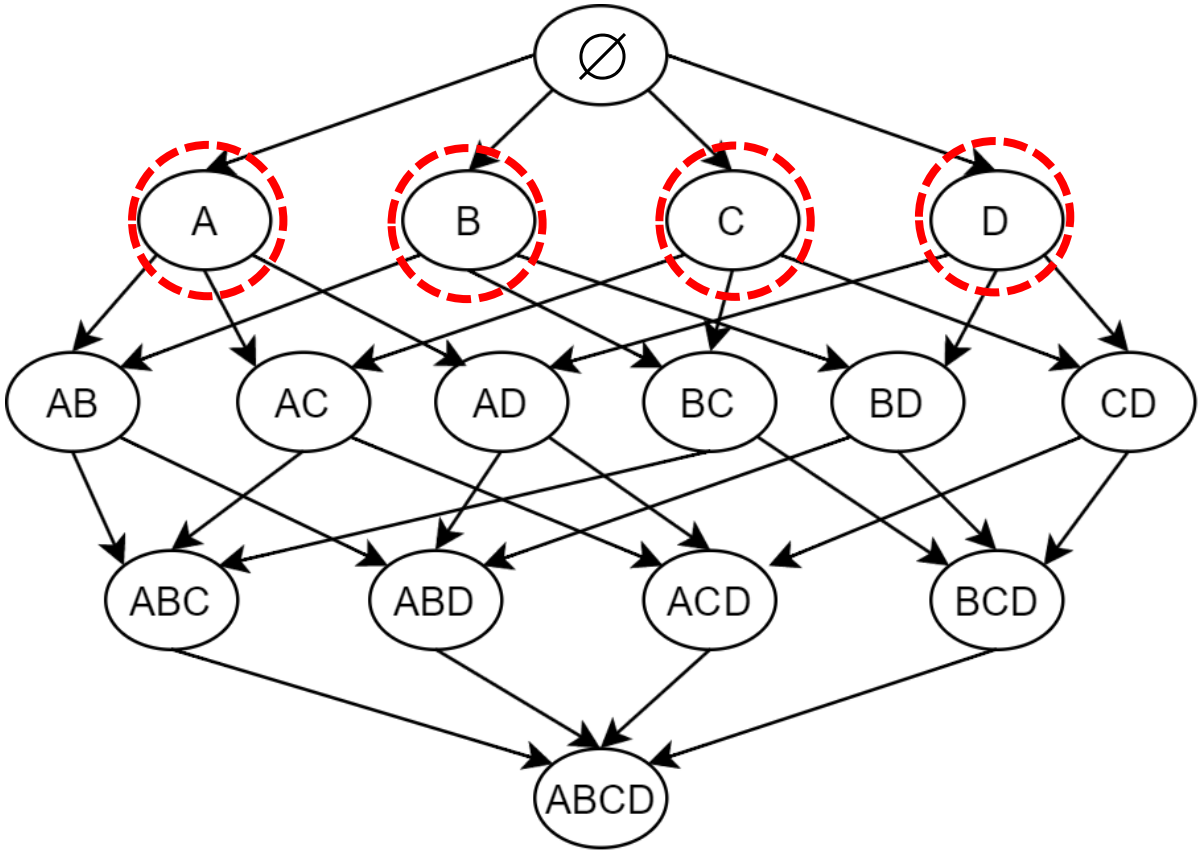
Solid Circle ∅

Solid Box ∅

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
A: 0; B: 0;
C: 0; D: 0

Dashed Box
∅

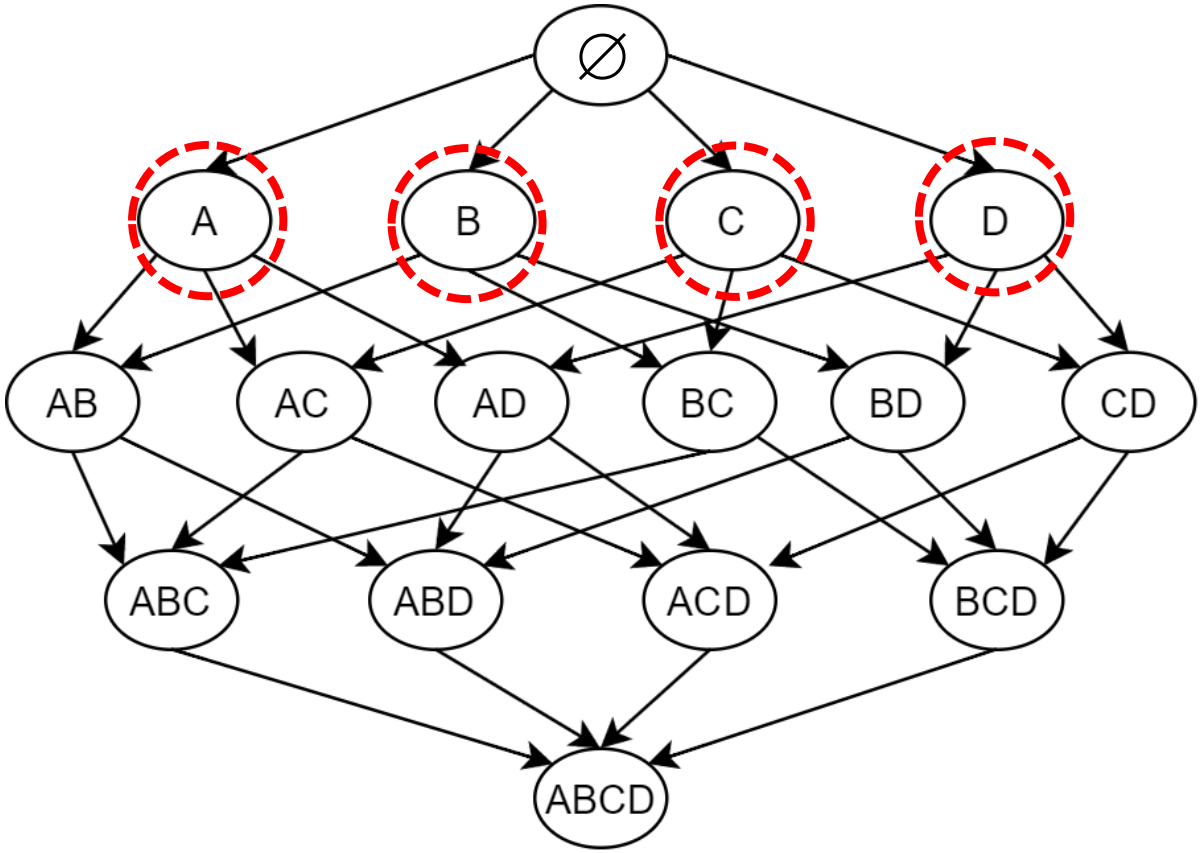
Solid Circle
∅

Solid Box
∅

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
A: 4; B: 4;
C: 3; D: 4

Dashed Box
∅

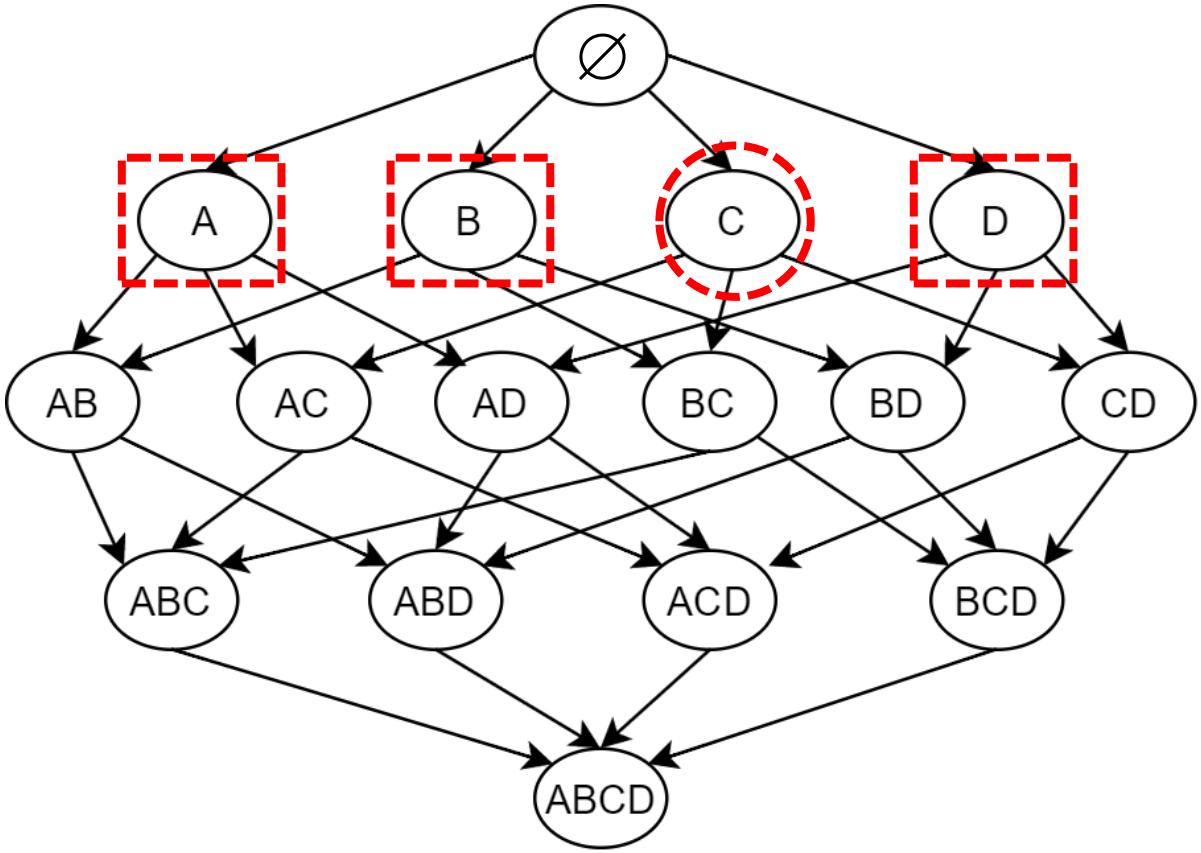
Solid Circle
∅

Solid Box
∅

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
C: 3

Dashed Box
A: 4;
B: 4;
D: 4

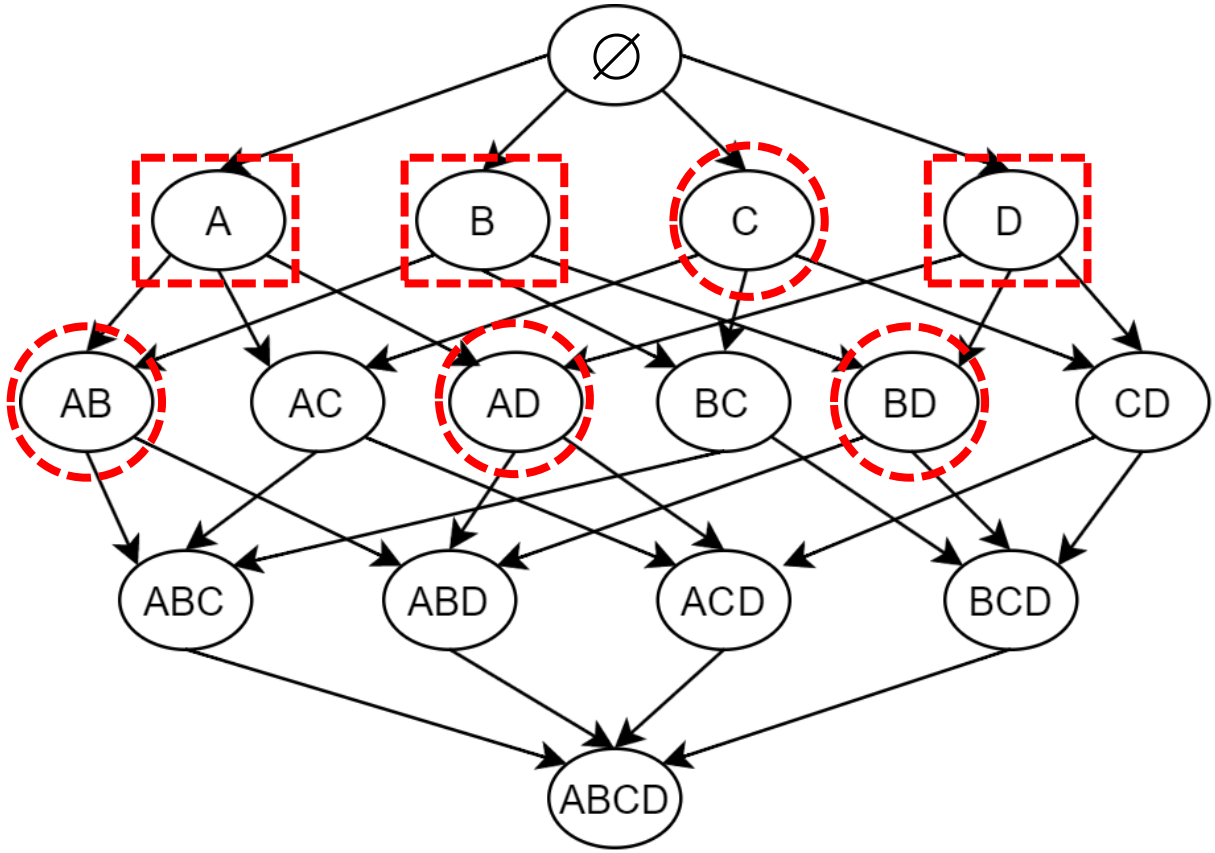
Solid Circle
∅

Solid Box
∅

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
C: 3; AB: 0;
AD: 0; BD: 0

Dashed Box
A: 4;
B: 4;
D: 4

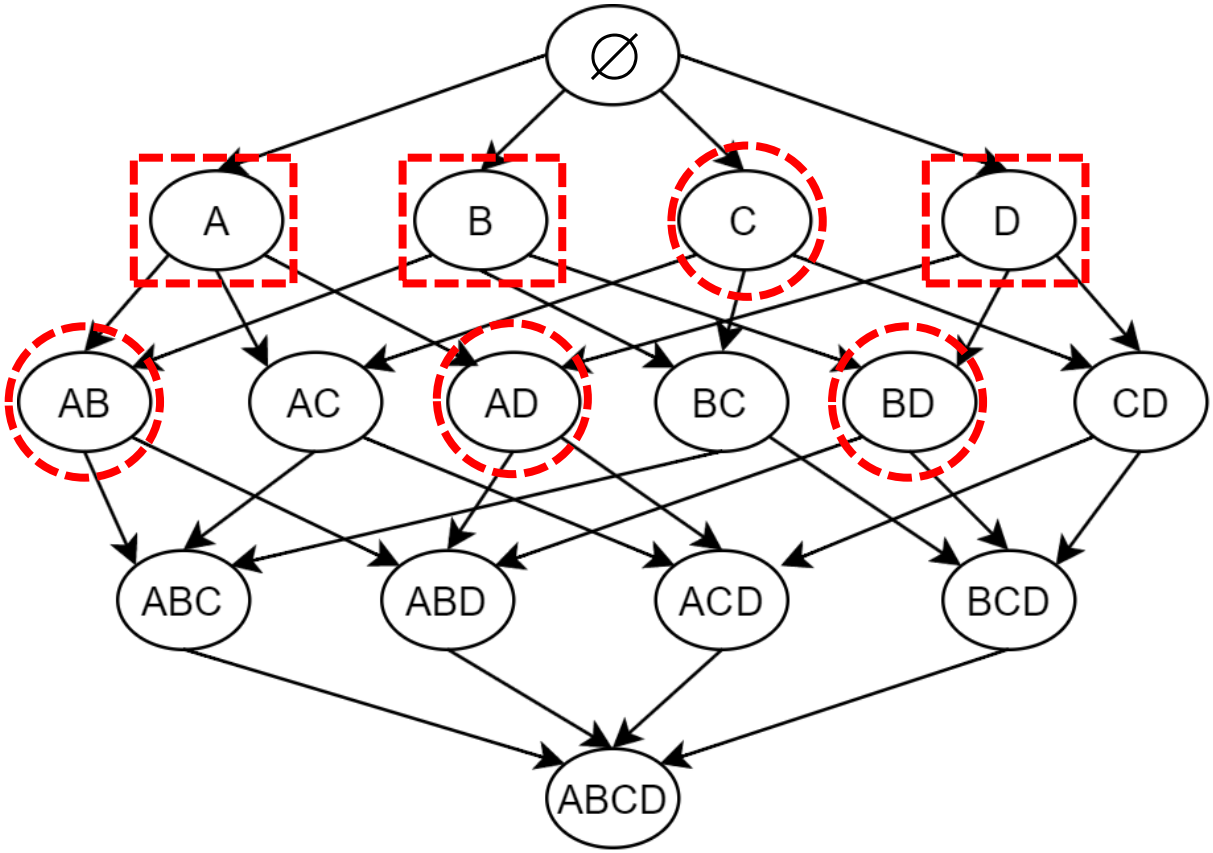
Solid Circle
∅

Solid Box
∅

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
C: 3; AB: 2;
AD: 0; BD: 0

Dashed Box
A: 7;
B: 8;
D: 4

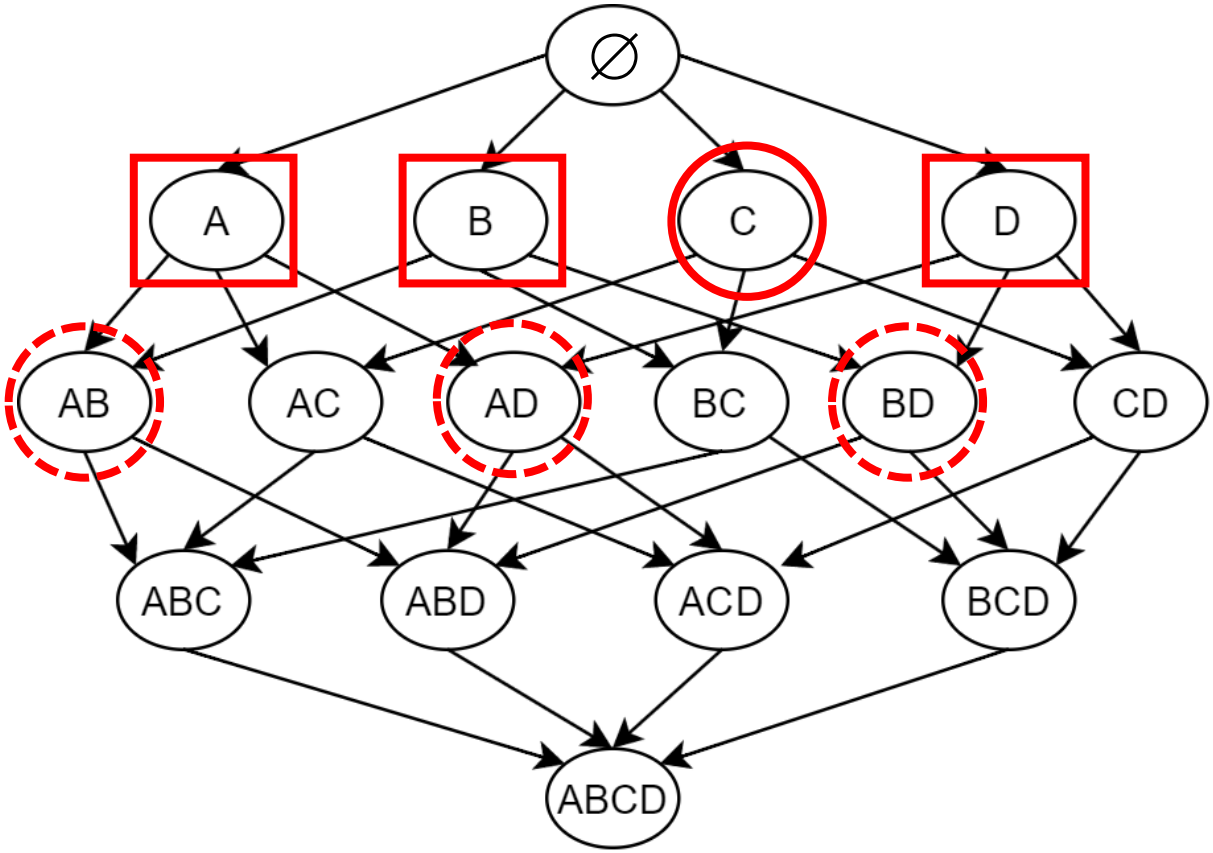
Solid Circle
∅

Solid Box
∅

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
AB: 2; AD: 0;
BD: 0

Dashed Box
∅

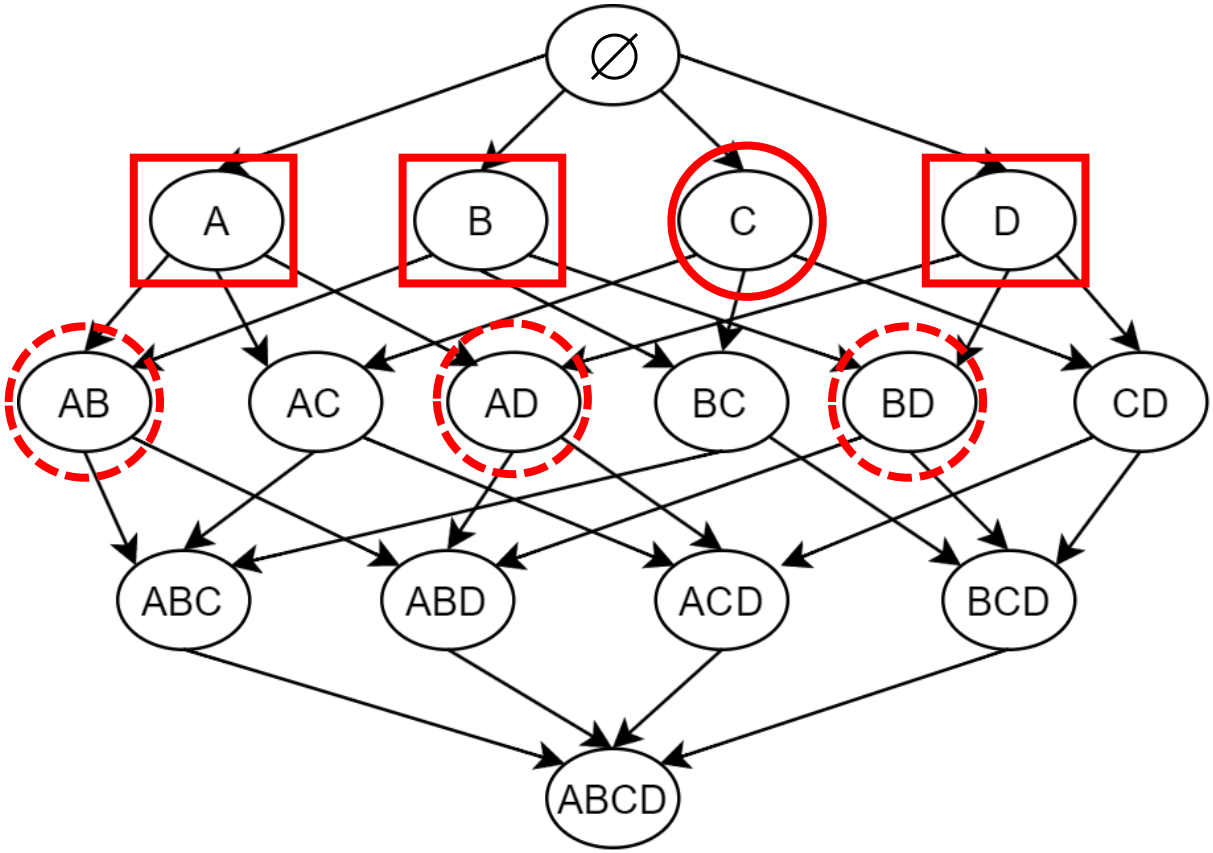
Solid Circle
C: 3

Solid Box
A: 7;
B: 8;
D: 4

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
AB: 5; AD: 3;
BD: 4

Dashed Box
∅

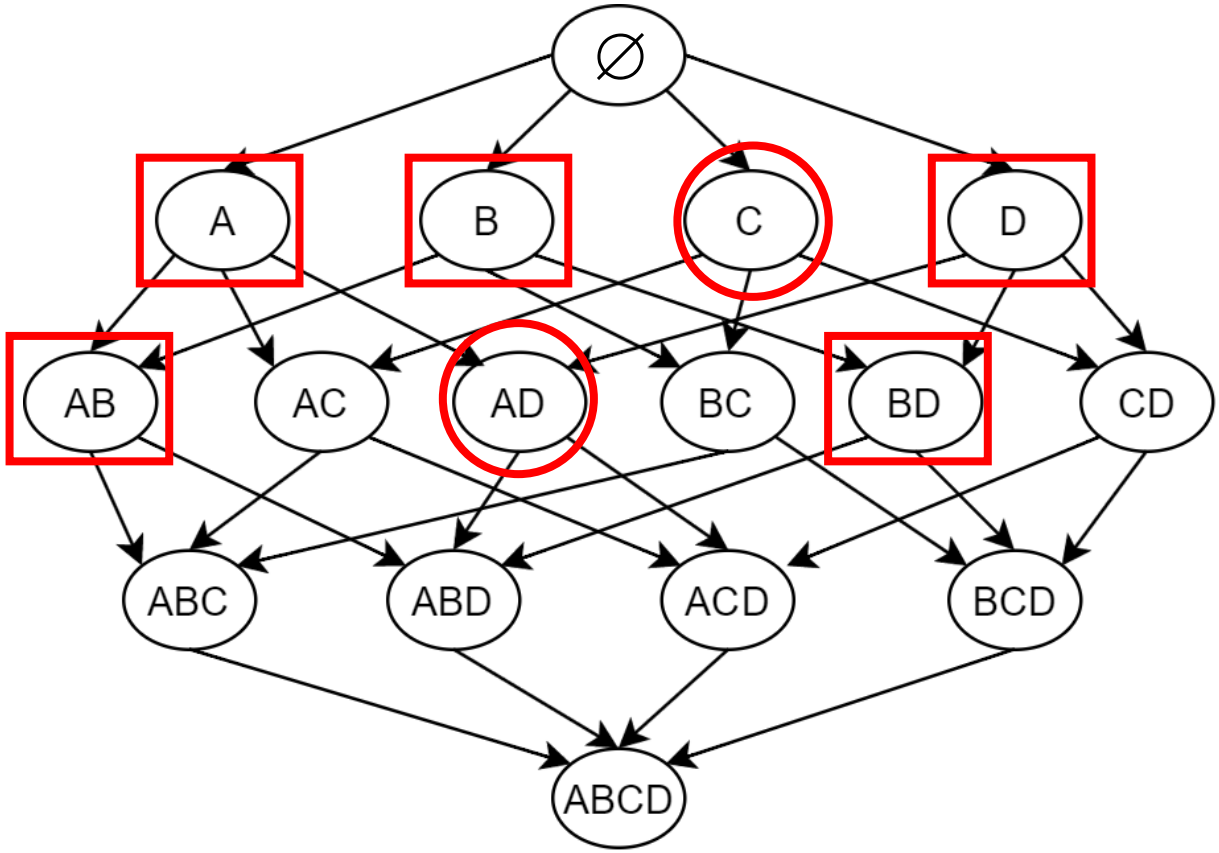
Solid Circle
C: 3

Solid Box
A: 7;
B: 8;
D: 4

DIC: пример

D *M*=5
minsup=4

1	A C
2	B C D
3	A B D
4	A B D
5	A B C D
6	A
7	B
8	B
9	A B
10	A B



Dashed Circle
∅

Dashed Box
∅

Solid Circle
C: 3;
AD: 3

Solid Box
A: 7; B: 8;
D: 4; AB: 5;
BD: 4

Алгоритм DIC

Algorithm 1. DIC(in $\mathcal{D}$, in $minsup$, in M, out $\mathcal{L}$)

```
                                          ▷ Initialize sets of itemsets
2: SolidBox ← ∅; SolidCircle ← ∅; DashedBox ← ∅
   DashedCircle ← ℐ
4: while DashedCircle ∪ DashedBox ≠ ∅ do
                          ▷ Scan database and rewind if necessary
6:     Read(𝒟, M, Chunk)
       if EOF(𝒟) then
8:         Rewind(𝒟)
       for all T ∈ Chunk do
10:                                   ▷ Count support of itemsets
           for all I ∈ DashedCircle ∪ DashedBox do
12:            if I ⊆ T then
                   support(I) ← support(I) + 1
14:                           ▷ Generate candidate itemsets
           for all I ∈ DashedCircle do
16:            if support(I) ≥ minsup then
                   MoveItemset(I, DashedBox)
18:                for all i ∈ ℐ do
                       C ← I ∪ i
20:                    if ∀s ⊆ C s ∈ SolidBox∪DashedBox then
                           MoveItemset(C, DashedCircle)
22:            ▷ Check full pass completion for itemsets
           for all I ∈ DashedCircle do
24:            if IsPassCompleted(I) then
                   MoveItemset(I, DashedBox)
26:        for all I ∈ DashedBox do
               if IsPassCompleted(I) then
28:                MoveItemset(I, SolidBox)
   ℒ ← SolidBox
```

Работы по тематике DIC

Год	Работа	Вклад
1997	Brin S., Motwani R., Ullman J.D., Tsur S. Dynamic Itemset Counting and Implication Rules for Market Basket Data. Proc. of the SIGMOD'1997, ACM Press, 1997, pp. 255–264.	Оригинальный последовательный алгоритм.
2001	Cheung D.W., Hu K., Xia S. An Adaptive Algorithm for Mining Association Rules on Shared-memory Parallel Machines. Distributed and Parallel Databases, vol. 9, no. 2, pp. 99–132, 2001.	Параллельный алгоритм • Для SMP-систем • Техника полного первого прохода • Техника отбрасывания кандидатов на основе виртуального разбиения • Эксперименты на 12 CPU с БД до 10^8 транзакций; ускорение сублинейное (до 8 на 12 CPU) *без учета I/O*
2011	Paranjape-Voditel P., Deshpande U., A DIC-based Distributed Algorithm for Frequent Itemset Generation. Journal of Software SW, vol. 6, no. 2, pp. 306–313, 2011.	Распределенный алгоритм • На базе MPI • Обмены возможно частыми наборами между узлами на каждой «остановке»; старт подсчета поддержки наборов, возможно частых на других узлах, но не являющихся таковыми на данном узле • Эксперименты на 6, 8, 12 узлах с БД до 10^5 транзакций; ускорение не измерялось; *exp* рост количества обменов
2015	Kumar P., Bhatt P., Choudhury R. Bitwise Dynamic Itemset Counting Algorithm. Proc. of the IEEE ICCIC'2015, IEEE, 2015, pp. 1–4.	Последовательный алгоритм • Вертикальное битовое представление (битовая карта транзакций, в которые входит набор) • Техника отбрасывания заведомо редких наборов • Слабые эксперименты

Параллельный DIC

- Прямое битовое представление данных
- Реализация структур данных
- Оптимизация до распараллеливания
- Распараллеливание
  - Порядок обработки данных
  - Балансировка загрузки

Битовое представление данных

- Объекты: $\tilde{I}=(i_1, i_2, \ldots, i_m)$
- Транзакция (набор): $T \subseteq \tilde{I}$ ($I \subseteq \tilde{I}$)
- База данных транзакций: $D=(T_1, T_2, \ldots, T_n)$, $T_i \subseteq \tilde{I}$
- Битовая маска транзакции: $Bitmask(T)$
  - i_p-й бит установлен $\Leftrightarrow i_p \in T$ ($1 \le p \le m$)
  - иначе i_p-й бит сброшен
- Битовая карта базы данных: $Bitmap(D)$
  - $Bitmap(D)=\{Bitmask(T_j): 1 \le j \le n\}$

D
3
2, 6, 7, 9

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													1		
							1		1	1				1	

$Bitmap(D)$
4
354

Битовое представление данных

- Сокращение размера базы данных
  - D можно целиком поместить в оперативную память
- Ускорение вычисления поддержки
  - $I \sqsubseteq T \Leftrightarrow Bitmask(I)$ **AND** $Bitmask(T) = Bitmask(I)$
  -

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I										1				1		
T								1		1	1				1	

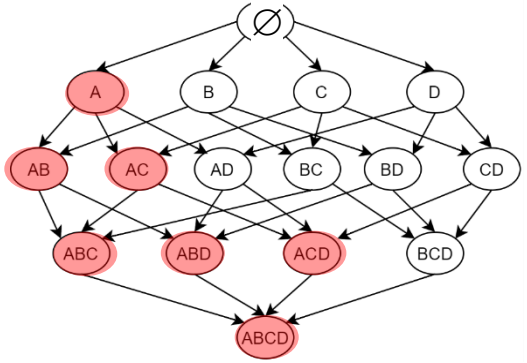
- Усложнение генерации кандидата
  - $c \leftarrow Bitmask(a)$ **OR** $Bitmask(b) \Leftrightarrow a$ и b отличны в одном бите
  - (1,2,3,4)←(1,2,3) **OR** (1,2,4), но (1,2,3) и (1,3,5)→Ø
- Случай m>64
  - объектов больше sizeof(unsigned long long int)
  - The GNU Multiple Precision Arithmetic Library, https://gmplib.org/

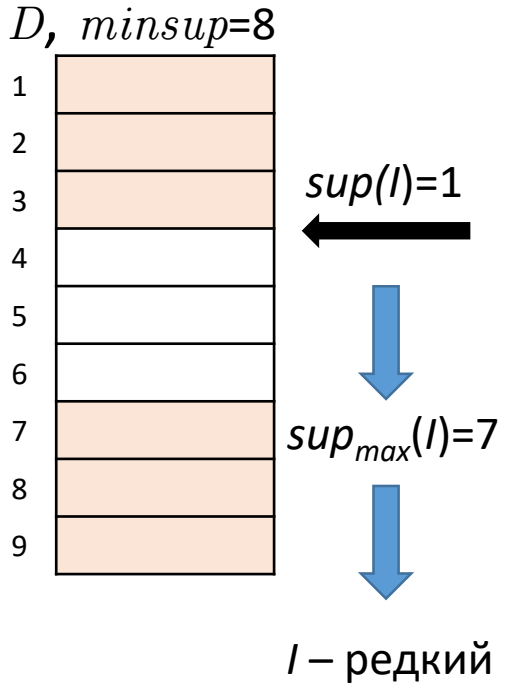
Реализация структур данных

- База данных транзакций (битовая карта)
  - unsigned long long int * **BITMAP**;
  - **BITMAP** = _mm_malloc(n*sizeof(unsigned long long int)) // Выравнивание
- Набор – struct {
  - mask // Битовая маска
  - k // Количество объектов в наборе
  - stopNo // Номер «остановки»
  - supp // Поддержка набора
  - fig // Вид: одно из  } **itemset**;
- Множества наборов
  - C++ STL (Standard Template Library), класс вектор
  - vector<struct itemset> **DASHED** // Наборы 
  - vector<struct itemset> **SOLID** // Наборы 
  - vector<struct itemset> **CAND** // Наборы-кандидаты

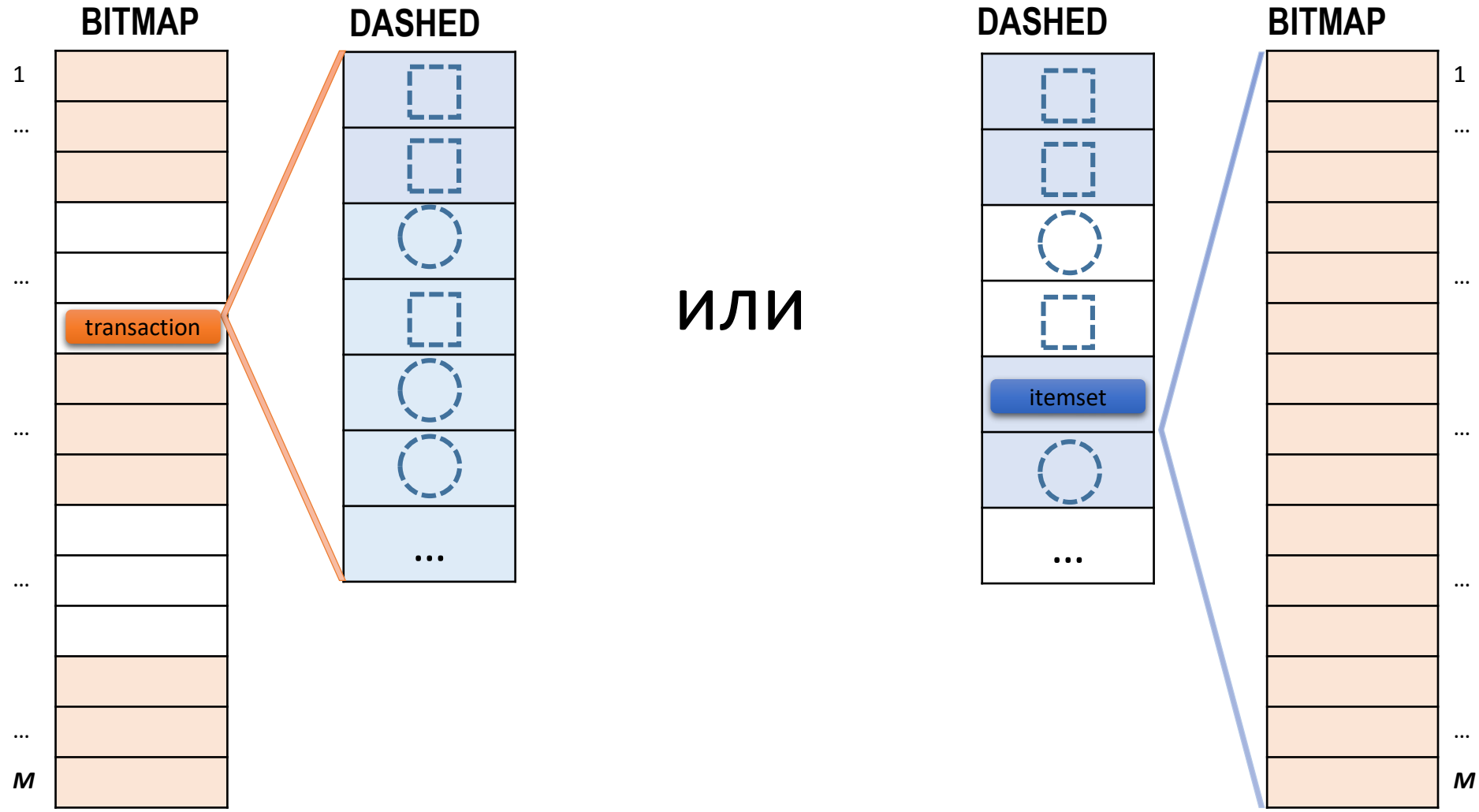
Оптимизация до распараллеливания

- Первый проход по D – полный
  - Найдем L_1
  - Отбросим редкие 1-наборы
  - Получим меньше 2-наборов кандидатов
- Отсечение заведомо редких наборов
  - После обработки очередных M транзакций вычисляем максимально возможную поддержку набора sup_{max}
    - предполагаем, что набор входит во все остальные транзакции
  - Если $sup_{max} < minsup$, то набор редкий
  - Отбрасываем также наборы, в которые входит данный редкий набор

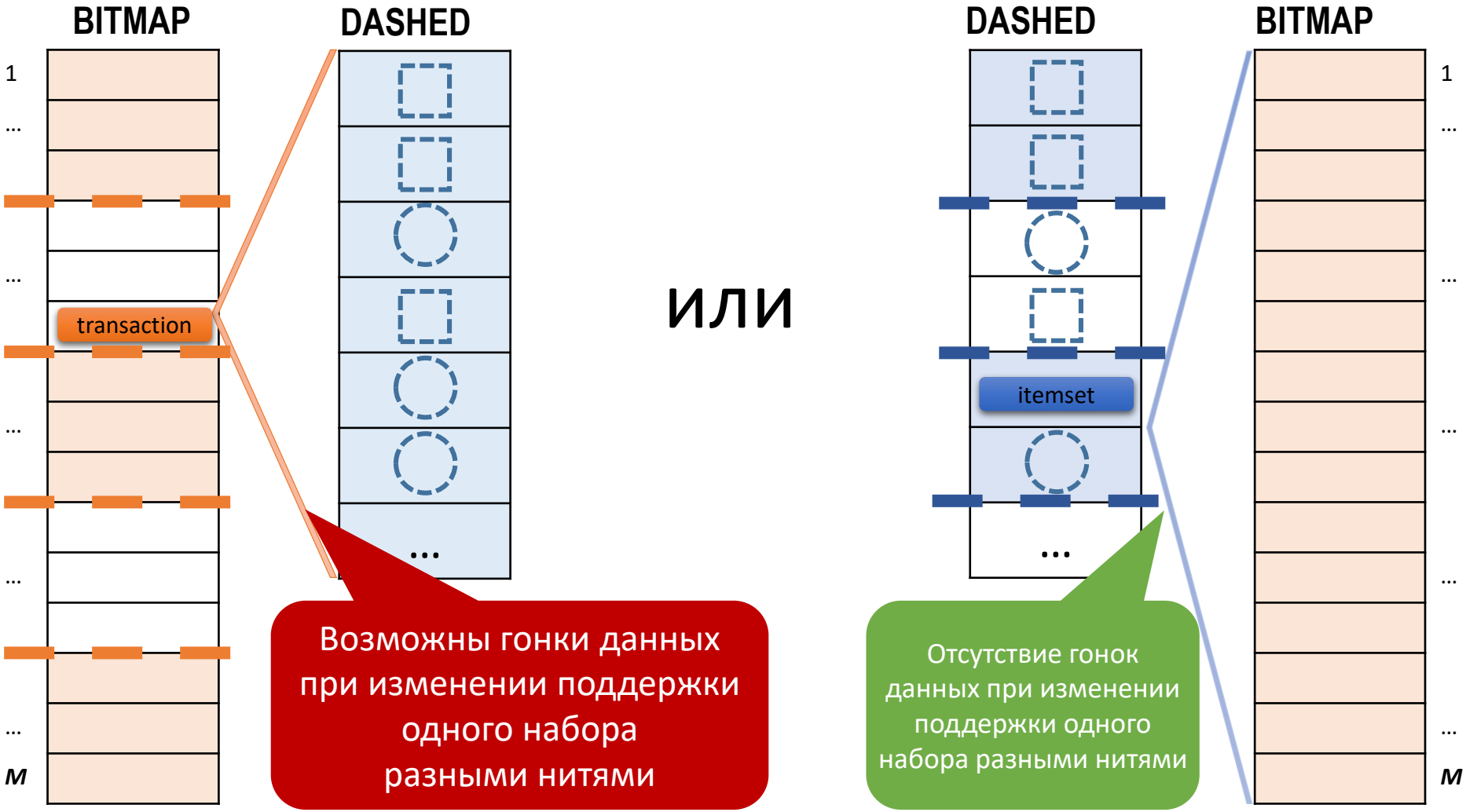




Распараллеливание: порядок обработки

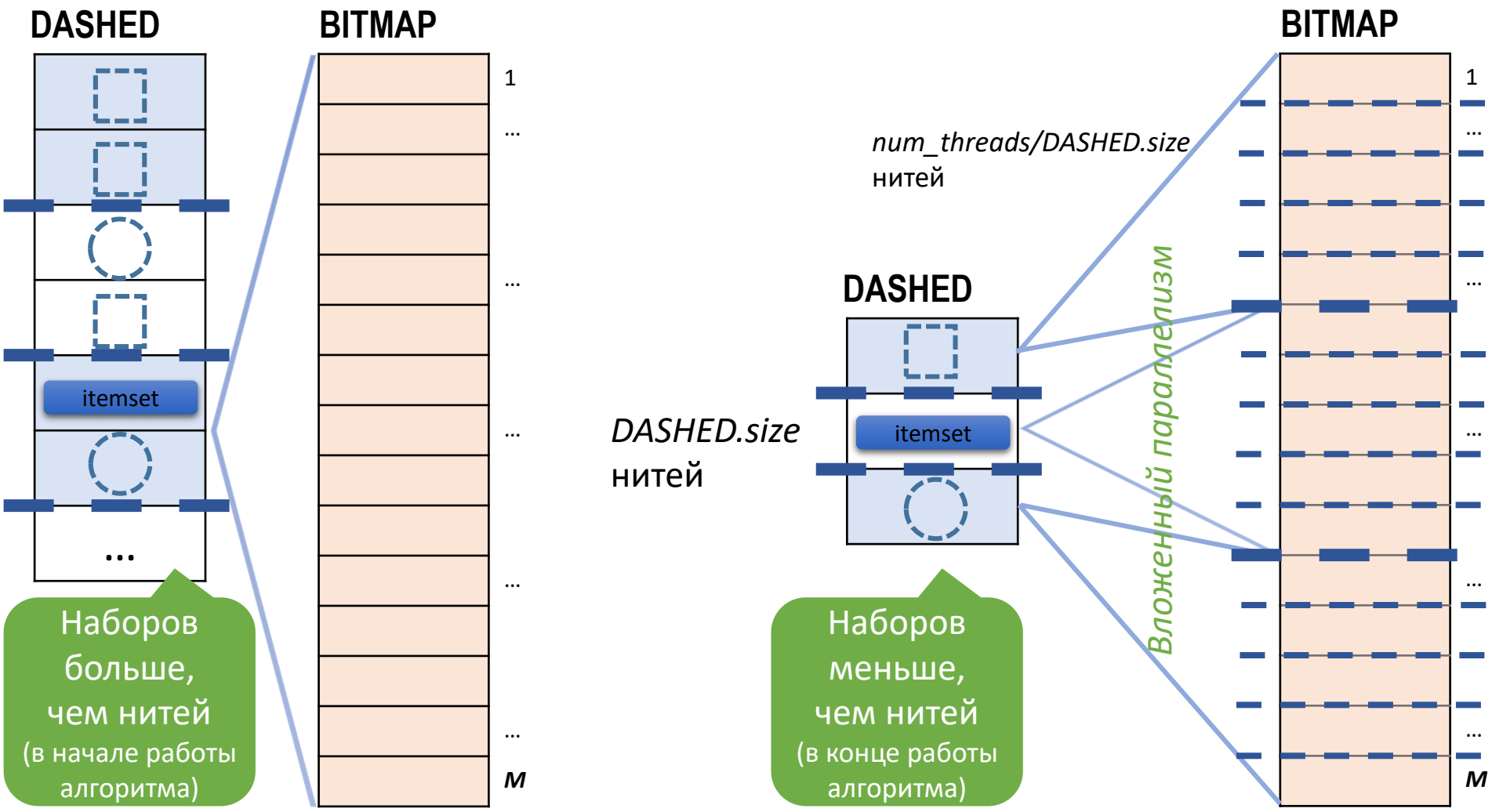


Распараллеливание: порядок обработки



Распараллеливание: балансировка

- Количество наборов в **DASHED** изменяется после очередной «остановки»



Параллельный DIC: подсчет поддержки

Algorithm 3. `CountSupport(in out` $DASHED$`)`

```
    if DASHED.size() ≥ num_of_threads then
 2:     #pragma omp parallel for
        for all I ∈ DASHED do
 4:         I.stop ← I.stop + 1
            for all T ∈ B[first] .. B[last] do
 6:             if I.mask AND T = I.mask then
                    I.supp ← I.supp + 1
 8: else
        omp_set_nested(true)
10:     #pragma omp parallel for
         num_threads(DASHED.size())
12:     for all I ∈ DASHED do
            I.stop ← I.stop + 1
14:         #pragma omp parallel for reduction(+:I.supp)
             num_threads(⌈num_of_threads / DASHED.size()⌉)
16:         for all T ∈ B[first] .. B[last] do
                if I.mask AND T = I.mask then
18:                 I.supp ← I.supp + 1
```

Параллельный DIC

Algorithm 2. ParalDIC(in $\mathcal{B}$, in $minsup$, in M, out $\mathcal{L}$)

▷ Initialize sets of itemsets
SOLID.$init()$; DASHED.$init()$
$k \leftarrow 1$
for all $i \in 0..m-1$ **do**
  $I.fig \leftarrow$ NIL; $I.bitmask \leftarrow 0$
  $I.mask \leftarrow$ SetBit$(I.mask, i)$
  $I.stop \leftarrow 0$; $I.supp \leftarrow 0$; $I.k \leftarrow k$
  SOLID.$push\_back(I)$
$stop_{max} \leftarrow \lceil \frac{n}{M} \rceil$; $stop \leftarrow 0$
FirstPass(SOLID, DASHED)
while not DASHED.$empty()$ **do**
  ▷ Scan database and rewind if necessary
  $stop \leftarrow stop + 1$
  if $stop > stop_{max}$ **then**
    $stop \leftarrow 1$
  $first \leftarrow (stop-1) \cdot M$; $last \leftarrow stop \cdot M - 1$
  $k \leftarrow k + 1$
  CountSupport(DASHED)
  CutDashedCircle(DASHED)
  GenCandidates(DASHED)
  CheckFullPass(DASHED)
$\mathcal{L} \leftarrow \{I \in$ SOLID$, I.fig =$ BOX$\}$

Параллельный DIC: отсечение кандидатов

Algorithm 4. `CutDashedCircle(in out` $DASHED$`)`

```
    #pragma omp parallel for
 2: for all I ∈ DASHED and I.fig = CIRCLE do
        if I.supp ≥ minsup then
 4:             ▷ Move appropriate itemsets to Dashed Box set
            I.fig ← BOX
 6:     else
                                ▷ Prune clearly infrequent itemset
 8:         supp_max ← I.supp + M · (stop_max − I.stop)
            if supp_max < minsup then
10:             I.fig ← NIL
                        ▷ Prune supersets of infrequent itemset
12:             for all J ∈ DASHED and J.fig = CIRCLE do
                    if I.mask AND J.mask = I.mask then
14:                     J.fig ← NIL
    DASHED.erase(∀I, I.fig = NIL)
```

Параллельный DIC: проверка полного прохода для кандидатов

Algorithm 5. `CheckFullPass(in out` $DASHED$`)`

```
    #pragma omp parallel for
2:  for all I ∈ DASHED do
        if I.stop = stop_max then
4:          if I.supp ≥ minsup then
                I.fig ← BOX
6:          SOLID.push_back(I)
            I.fig ← NIL
8:  DASHED.erase(∀I, I.fig = NIL)
```

Эксперименты

- Платформы

Спецификация	Процессор	Сопроцессор
Модель, Intel Xeon	X5680	Phi SE10X
Кол-во ядер	6	60
Тактовая частота, GGz	3.33	1.1
Гиперпоточность	2	4
Производительность, TFLOPS	0.371	1.076
Память, Gb	24	8

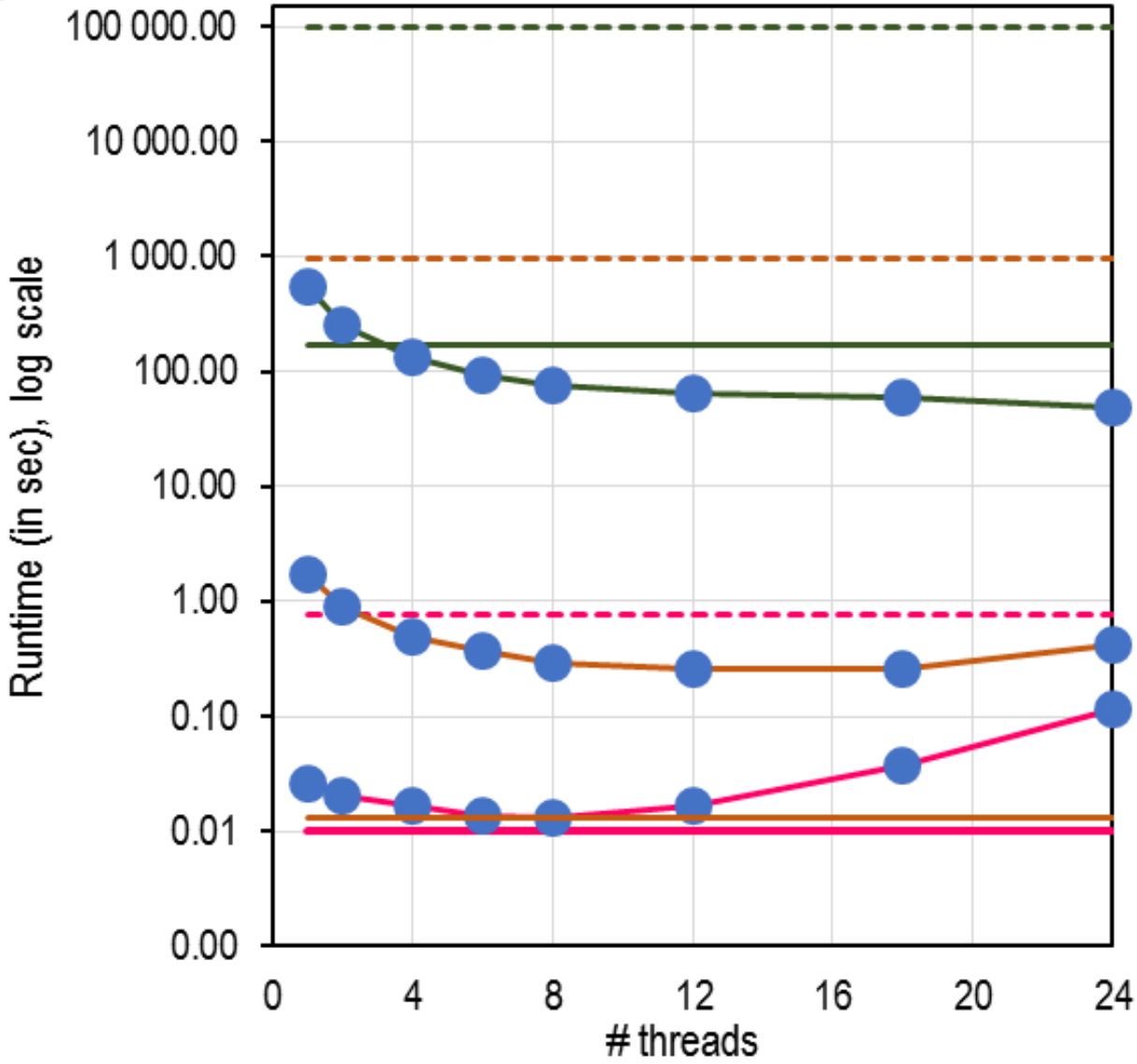
- Тестовые наборы

Набор	Кол-во транзакций	Кол-во объектов	Источник
SKIN	245 057	11	Dhall A., et. al. Adaptive Digital Makeup // ISVC 2009. LNCS, vol. 5876. Springer, 2009, pp. 728–736.
RECORDLINK	574 912	29	Sariyar M., et al. Controlling False Match Rates in Record Linkage Using Extreme Value Theory // Journal of Biomedical Informatics, vol. 44, no. 4, pp. 648–654, 2011.
20M	$2*10^7$	64	Синтез с помощью генератора IBM Quest (как у авторов DIC). Ср. длина транзакции 20, ср. длина частого набора 10.

Производительность

[1] By Cristian Borgelt

[2] By Bart Goethals



Набор *SKIN*

Serial Apriori [1]

Serial DIC (M=120 000) [2]

Parallel DIC (M=120 000)

Набор *RECORDLINK*

Serial Apriori [1]

Serial DIC (M=287 456) [2]

Parallel DIC (M=287 456)

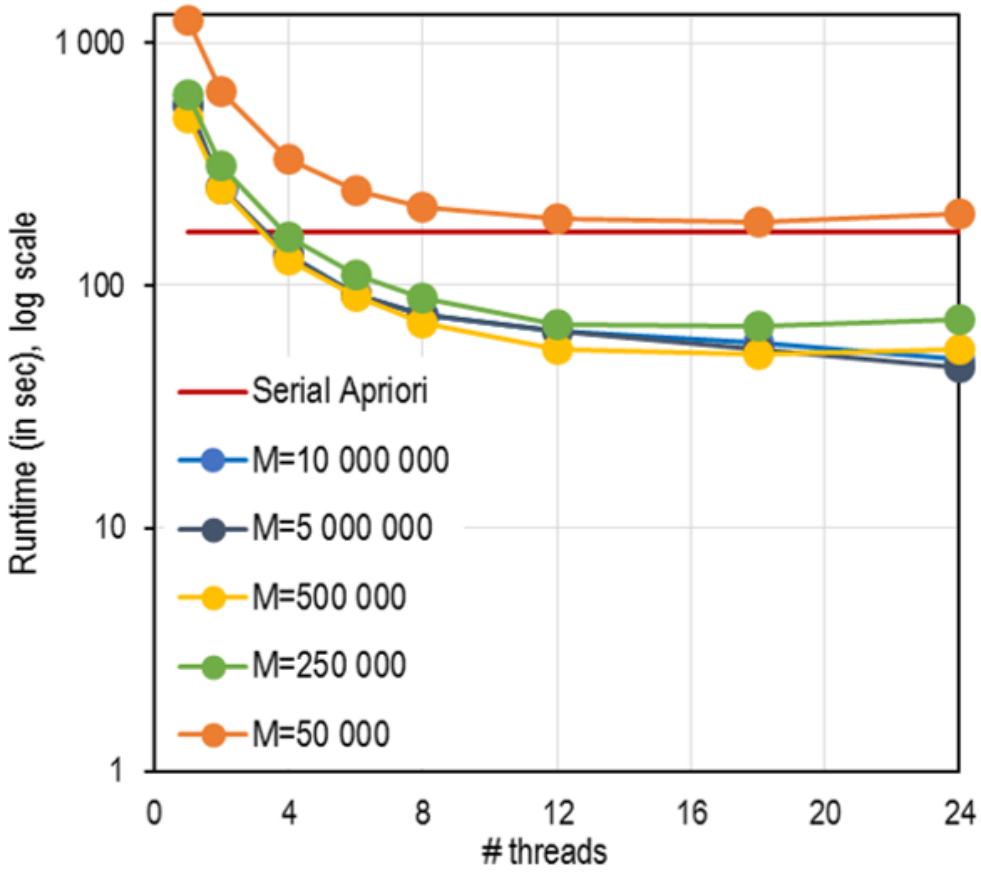
Набор *20M*

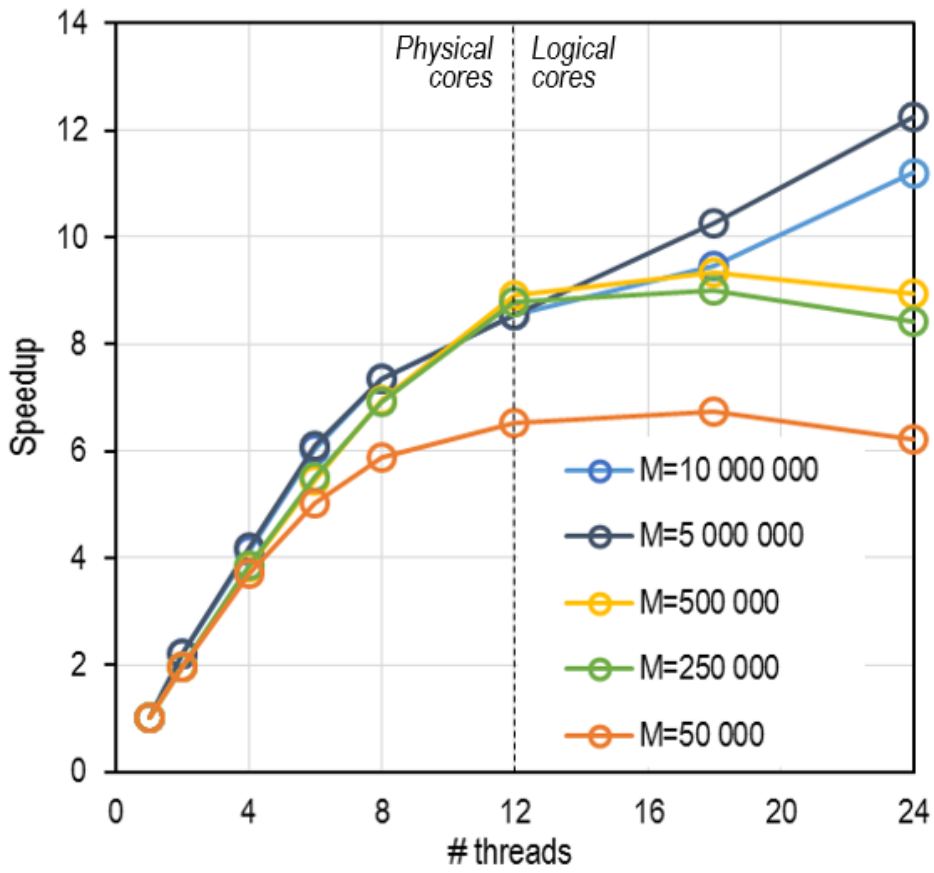
Serial Apriori [1]

Serial DIC (M=10 000 000) [2]

Parallel DIC (M=10 000 000)

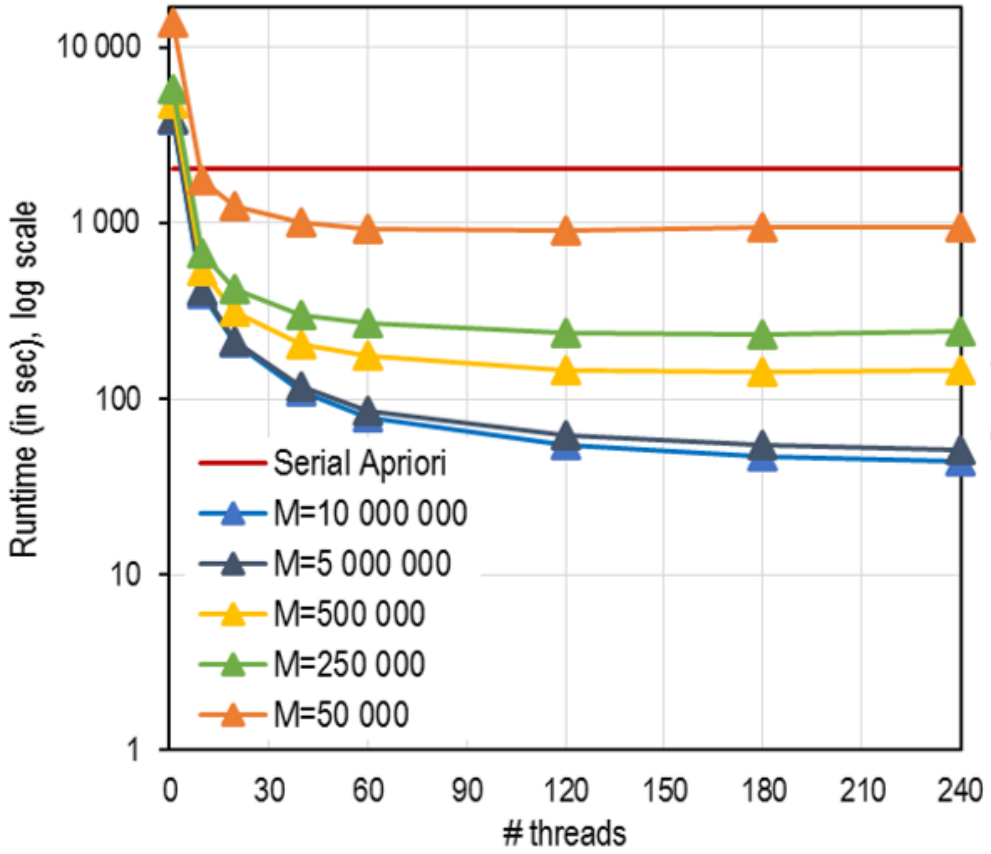
Масштабируемость по M (CPU)

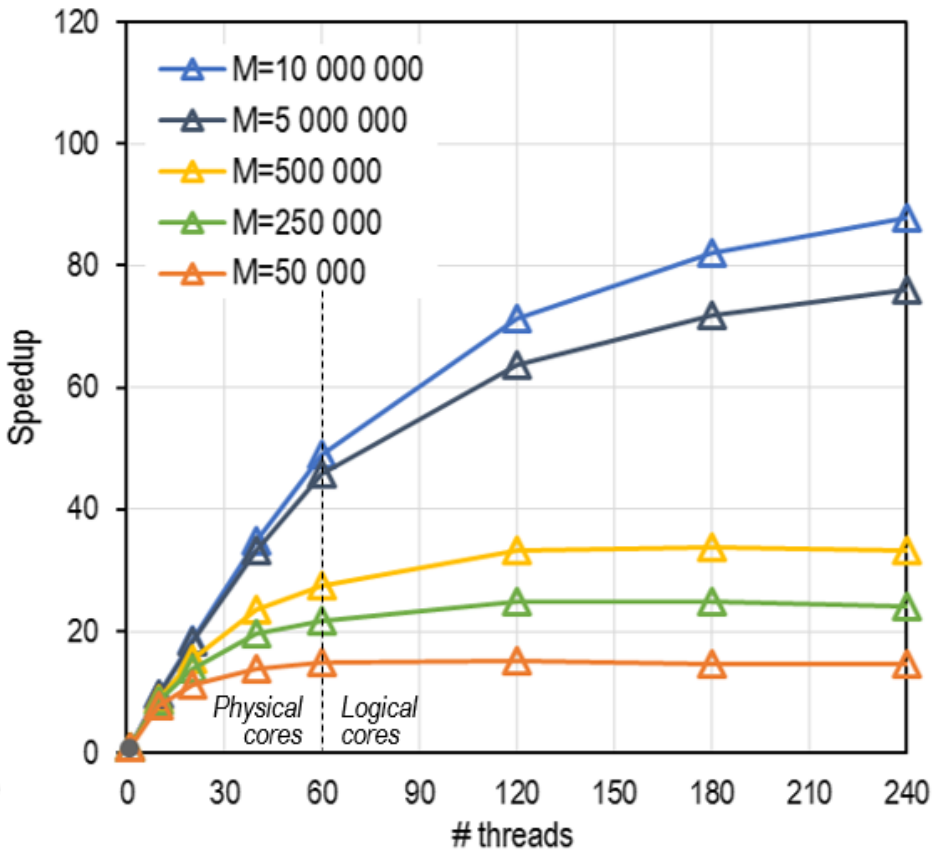




Набор: *20M*
minsup=0.1

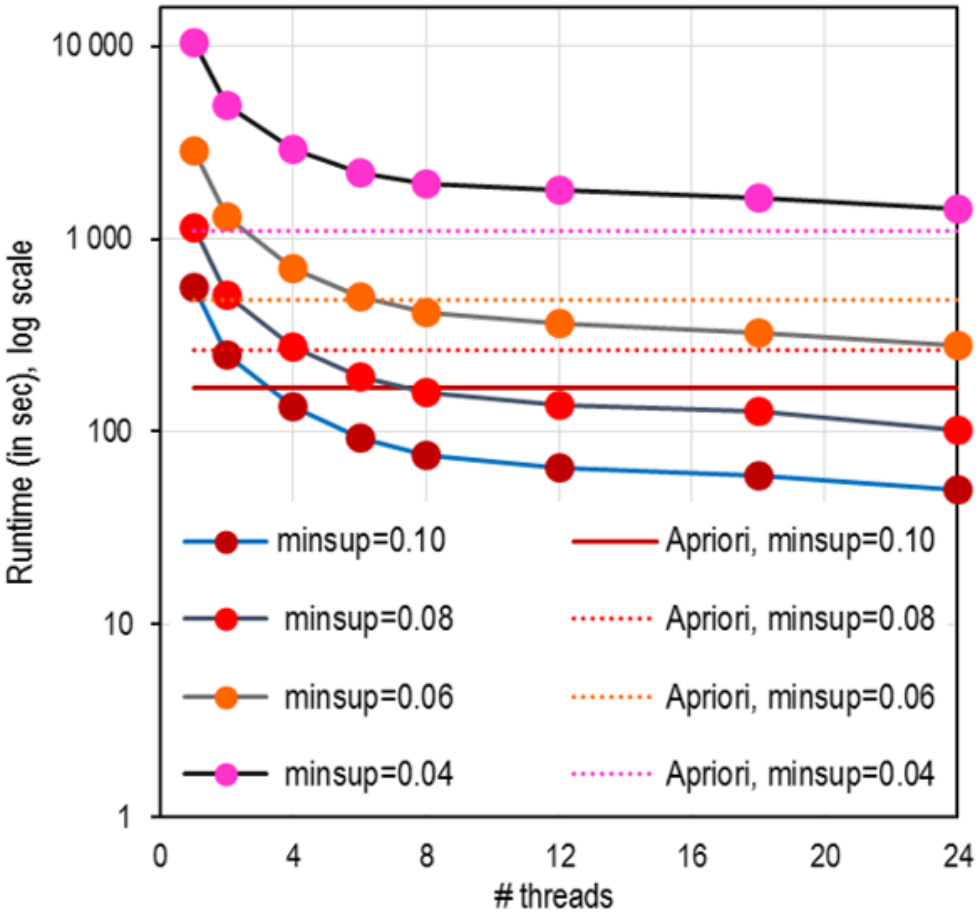
Масштабируемость по M (Phi)

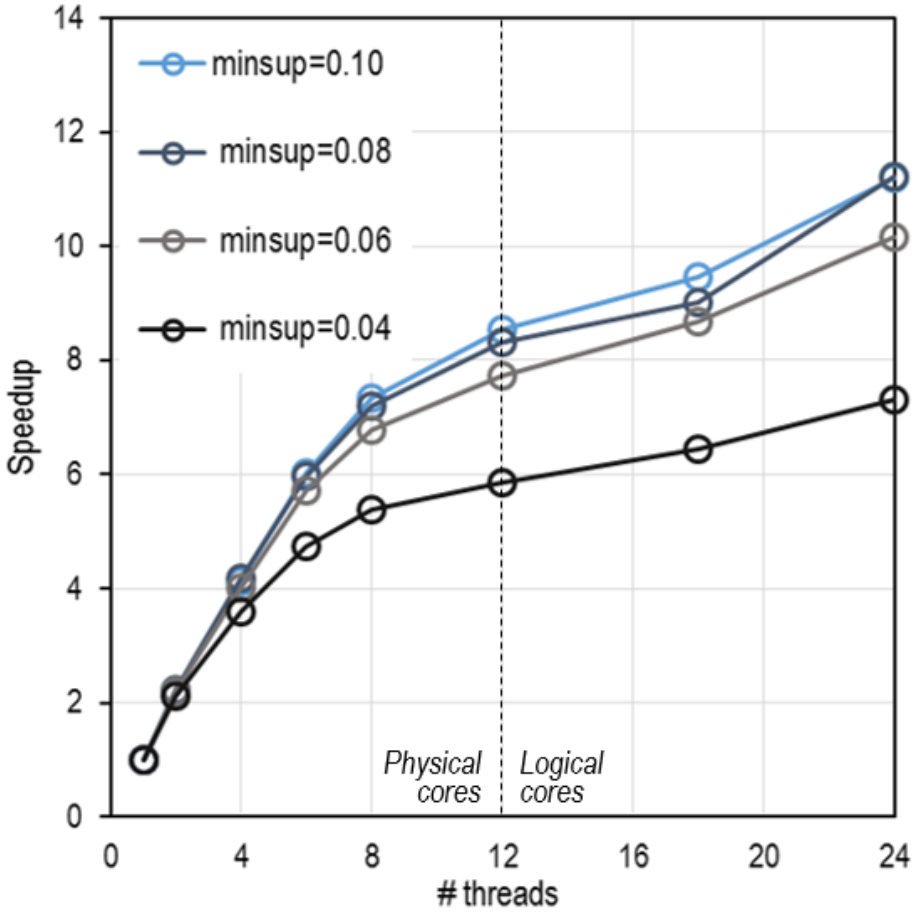




Набор: *20M*
minsup=0.1

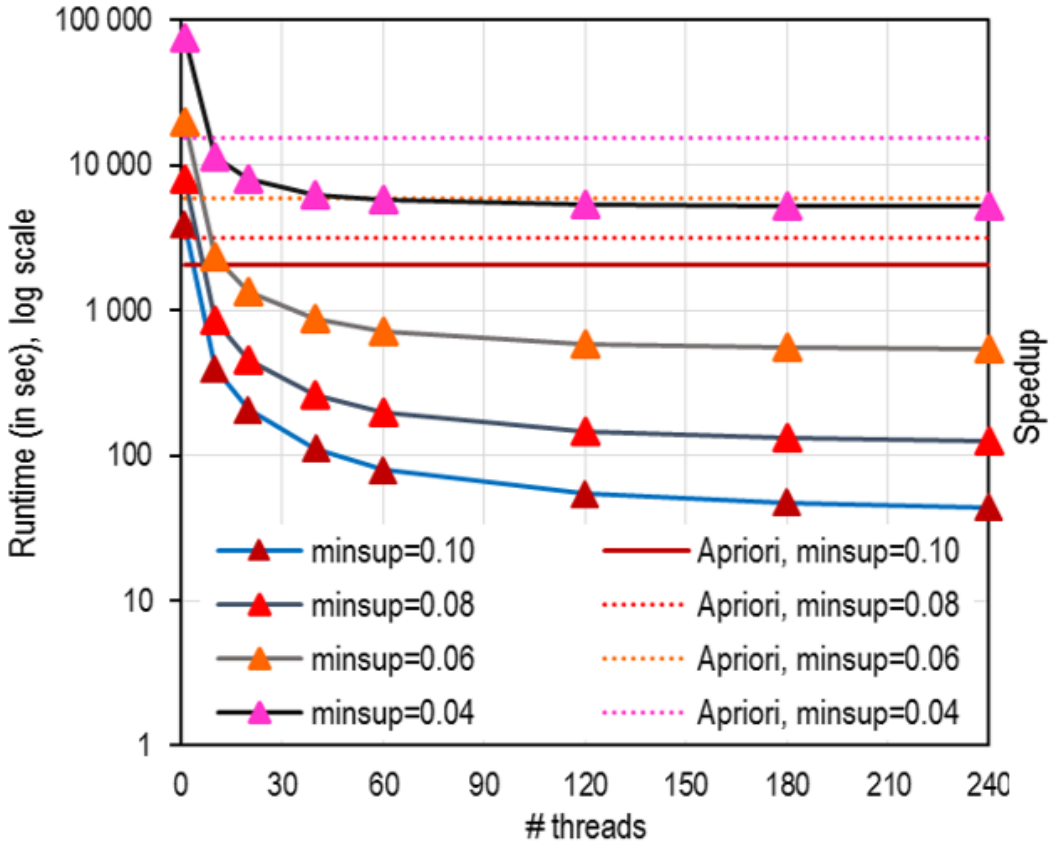
Масштабируемость по *minsup* (CPU)

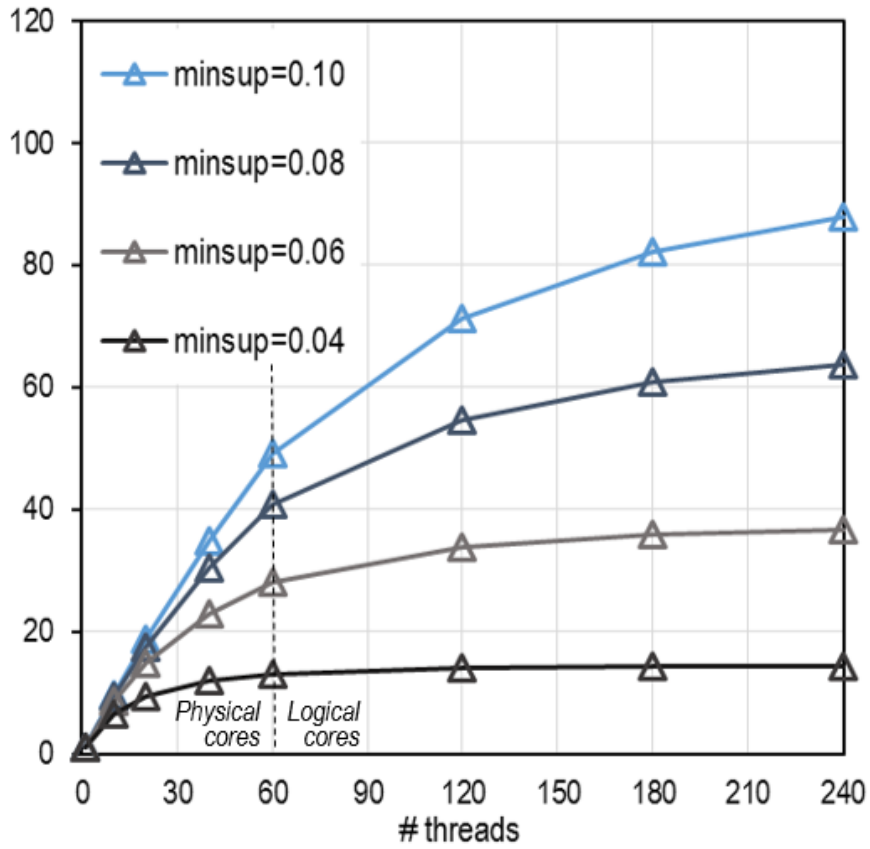




Набор: *20M*
M=10^7

Масштабируемость по *minsup* (Phi)





Набор: *20M*
M=10^7