АРХИТЕКТУРА СОВРЕМЕННЫХ КОМПЬЮТЕРОВ

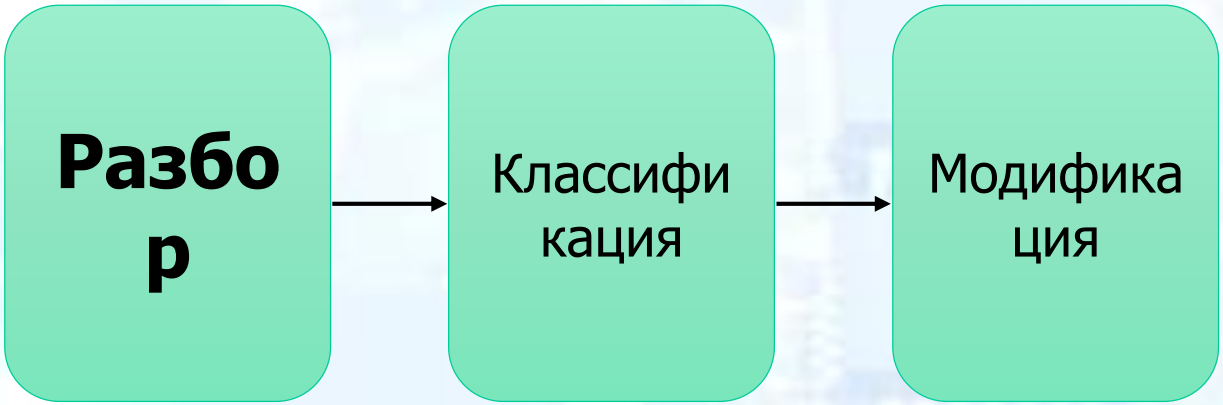
Лекция 9:
Обработка заголовков пакетов

ВМК МГУ им. М.В. Ломоносова, Кафедра АСВК
Доцент, к.ф.-м.н. Волканов Д.Ю.

План лекции

- Подходы к разбору заголовка пакета
- Постановка задачи классификации пакетов
- Подходы к решению задачи и требования к ним
- Подходы на основе декомпозиции задачи
- Подходы на основе деревьев поиска

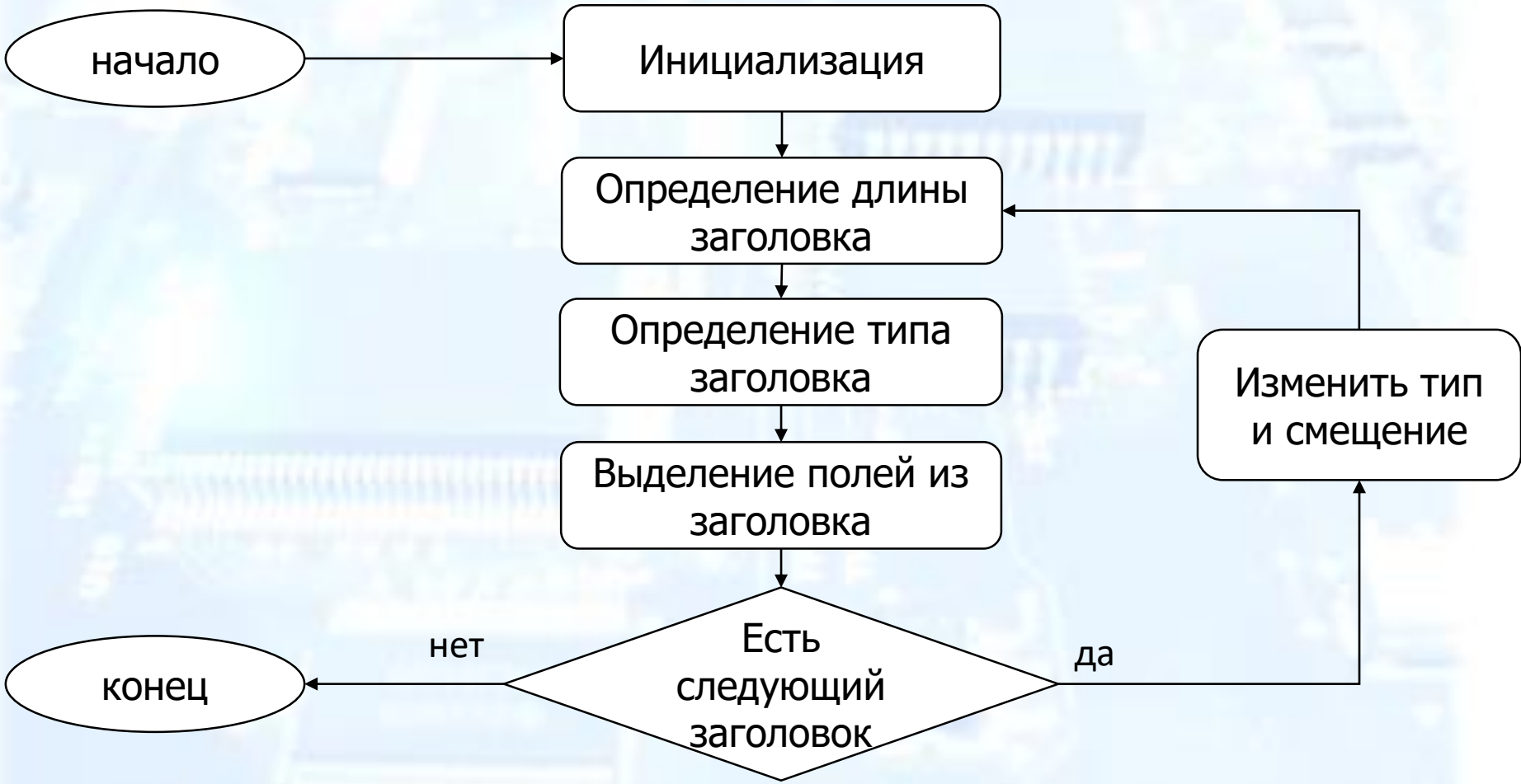
ОБРАБОТКА ПАКЕТА



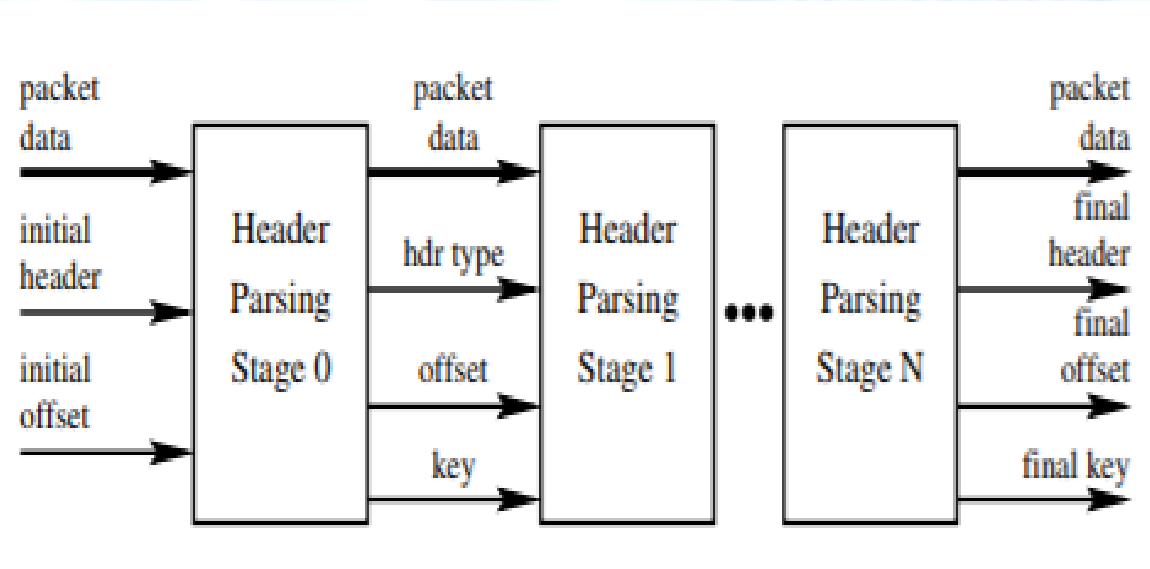
Задачи анализатора заголовков:

- Определение последовательности заголовков в пакете
- Выделение полей из заголовков
- Передача выделенных полей на стадию классификации

Базовый алгоритм обработки пакетов



Конвейерная схема анализа заголовка



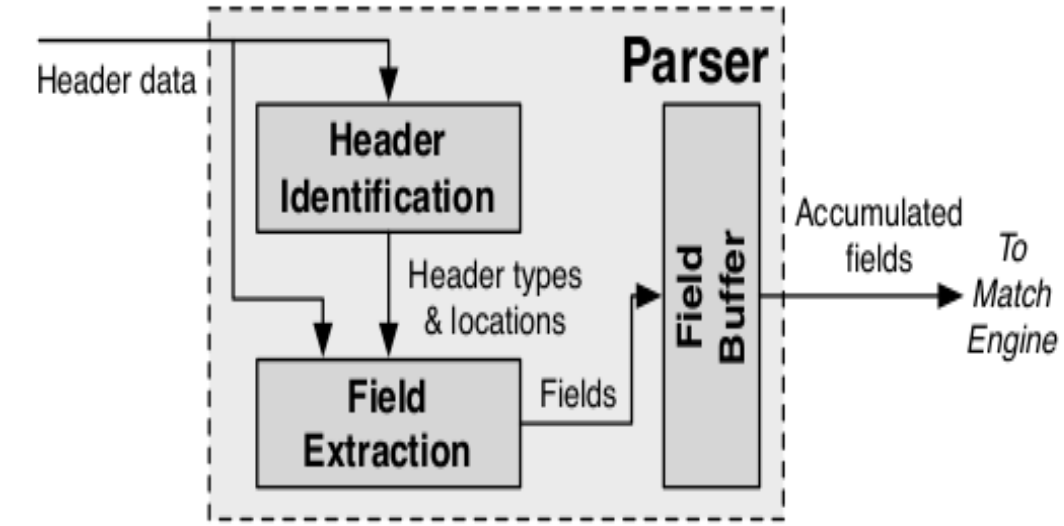
Анализатор представляет из себя конвейер. На каждой стадии конвейера обрабатывается один заголовок.

5

Модульная схема анализатора заголовка

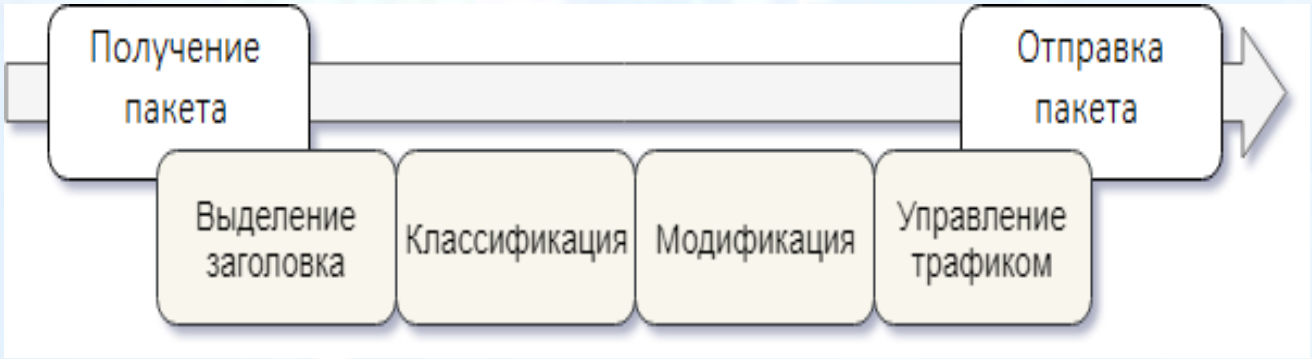
Анализатор состоит из 3 модулей:

- Модуль идентификации заголовков
- Модуль выделения полей
- Буфер



Классификация пакетов

- Классификация пакета — это определение номера порта (портов), на который необходимо отправить пакет, и действий над заголовком пакета, которые необходимо выполнить перед отправкой пакета



7

Виды таблиц классификации

- Классификация по одному полю:
  - Таблица MAC адресов
  - Таблица IP префиксов
  - Таблица MPLS меток
- **Классификация по нескольким полям:**
  - Таблица ACL (Access Control List)
  - Таблица потоков OpenFlow

Таблица потоков OpenFlow

- Таблица потоков – набор правил $R = \{r_1, r_2, \ldots, r_n\}$
- Правило $r_i = (p_i, f_i, a_i)$:
  - p_i – приоритет
  - f_i – вектор значений полей
  - a_i – список действий

- Правило $r_i \in R$ *идентифицирует* заголовок пакета h, если для $\forall\ v \in f_i$ и соответствующего $w \in h$:
  - $w = v$

  или
  - w = v по битам маски

  или
  - TCP = * («пробой»)

Rule id	Ingress port	Ether type	IP src	IP dst	IP proto	IP ToS bits	TCP/UDP Src Port	TCP/UDP Dst Port	Action
R1	3	2048	206.159.213.125/32	101.152.182.8/30	0x06f/0xff	0	1024 : 65535	*	drop
R2	3	2048	15.25.70.8/30	*	*	0	*	0:1599	forward
R3	5	2048	*	18.152.125.32/30	0x11/0xff	1	1024 : 65535	1024 : 65535	enqueue
R4	5	2048	206.159.213.125/32	*	0x06f/0xff	1	*	80	forward
R5	*	*	*	*	*	*	*	*	drop

Постановка задачи классификации пакетов

Дано:

- Множество полей F
- Набор правил $R = \{r_1, r_2, \ldots, r_n\}$

Для любого вектора значений полей заголовка h **найти**:

- Правило $r_i \in R$ такое, что:
  1. r_i идентифицирует h
  2. $\forall\ r_j \in R: j \neq i$ и r_j идентифицирует h, тогда $p_i > p_j$

Rule id	Ingress port	Ether type	IP src	IP dst	IP proto	IP ToS bits	TCP/UDP Src Port	TCP/UDP Dst Port	Action
R1	3	2048	206.159.213.125/32	101.152.182.8/30	0x06f/0xff	0	1024 : 65535	*	drop
R2	3	2048	15.25.70.8/30	*	*	0	*	0:1599	forward
R3	5	2048	*	18.152.125.32/30	0x11/0xff	1	1024 : 65535	1024 : 65535	enqueue
R4	5	2048	206.159.213.125/32	*	0x06f/0xff	1	*	80	forward
R5	*	*	*	*	*	*	*	*	drop

Требования к структурам данных

- **Высокая скорость поиска** достигается
  - Уменьшением занимаемого объема памяти
  - Уменьшением количества доступов к памяти
- **Высокая скорость обновления таблицы** достигается
  - Уменьшением времени построения структуры данных

11

Подходы к классификации пакетов

- Архитектурные
- Алгоритмические:
  - **Декомпозиция задачи**
  - **Деревья поиска (decision tree)**
- Смешанные

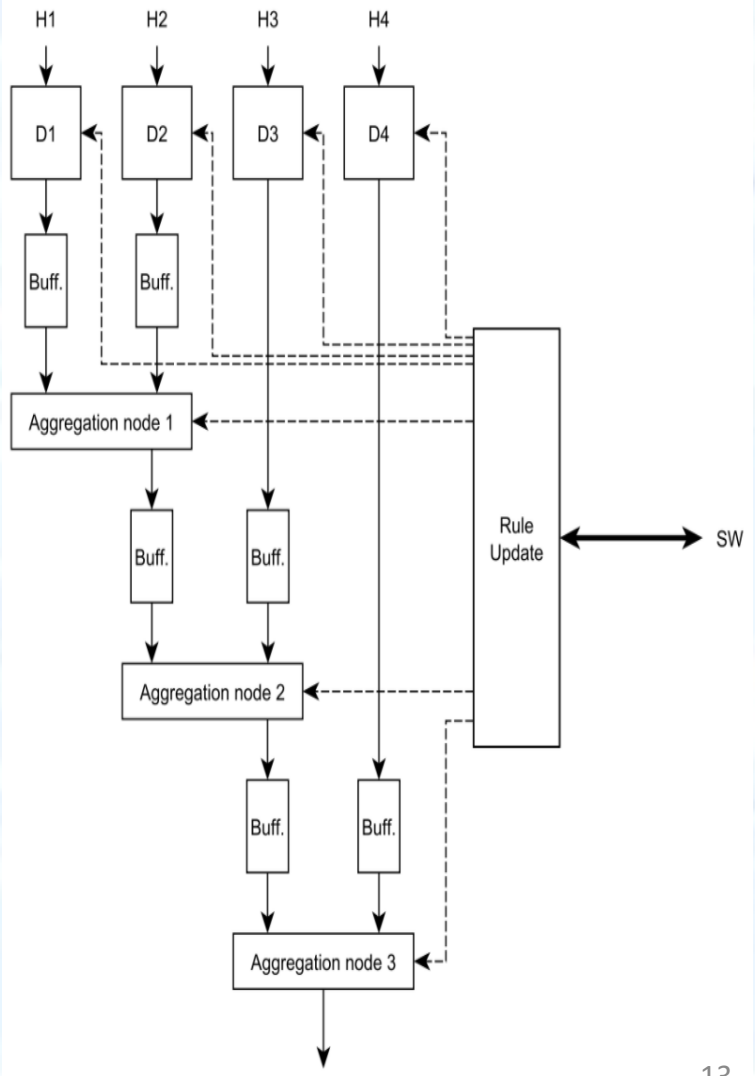
12

Distributed Crossproducting of Field Labels (DCFL)

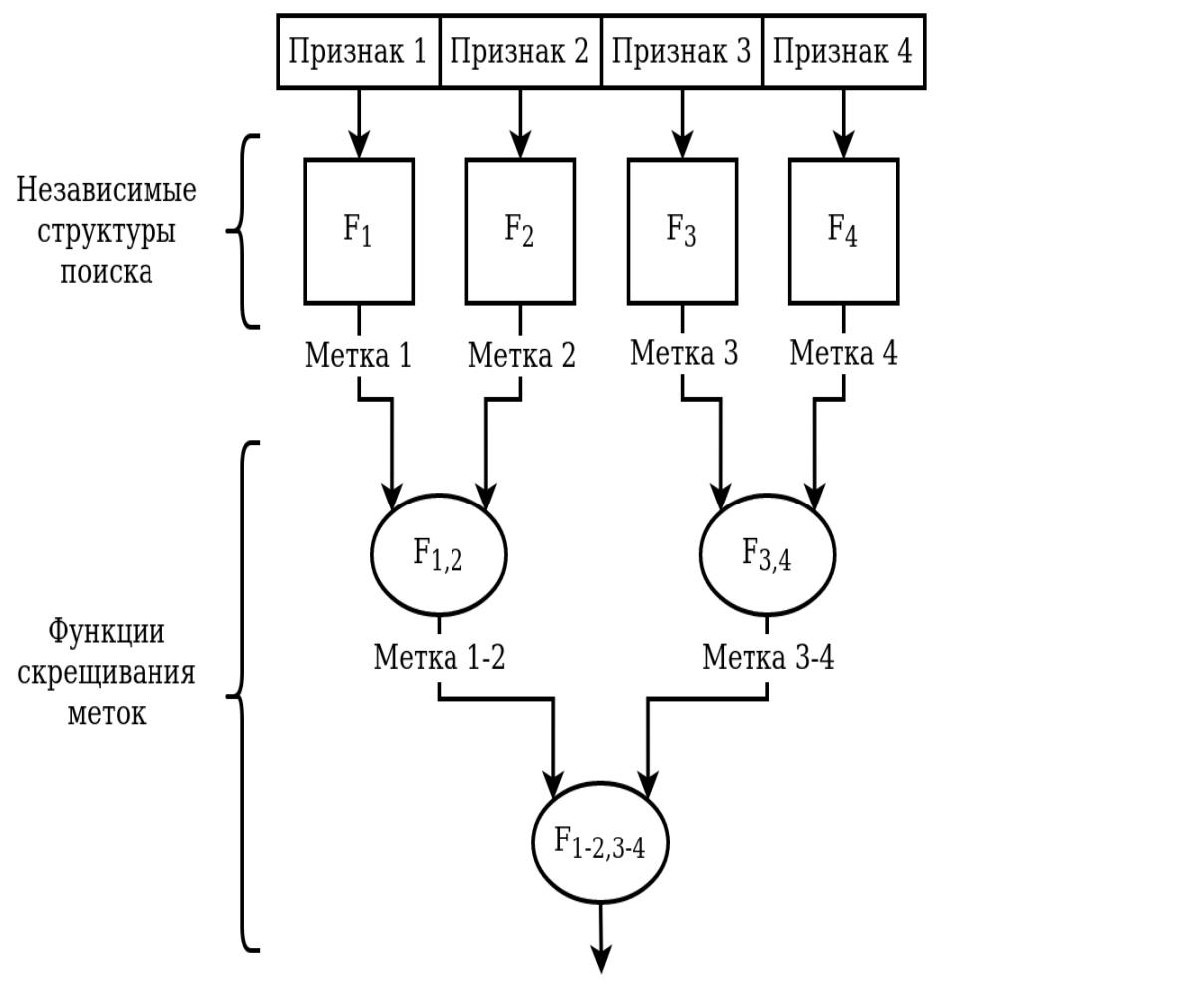
Задача классификации по нескольким полям

↓

Несколько задач классификации по одному полю



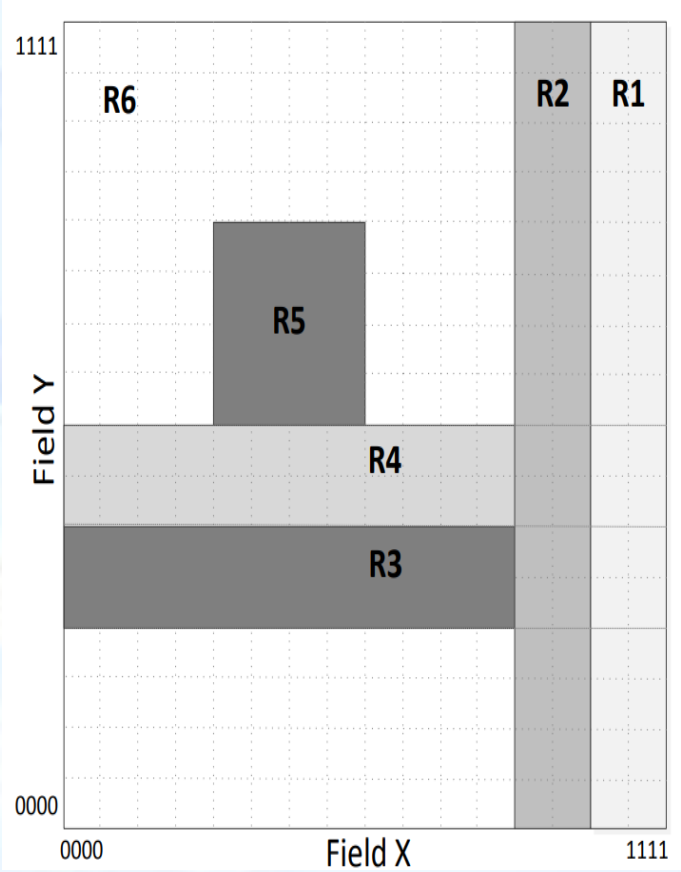
DCFL


Признак 1
Признак 2
Признак 3
Признак 4
Независимые структуры поиска
F1
F2
F3
F4
Метка 1
Метка 2
Метка 3
Метка 4
Функции скрещивания меток
F1,2
F3,4
Метка 1-2
Метка 3-4
F1-2,3-4

14

Геометрическое представление таблицы

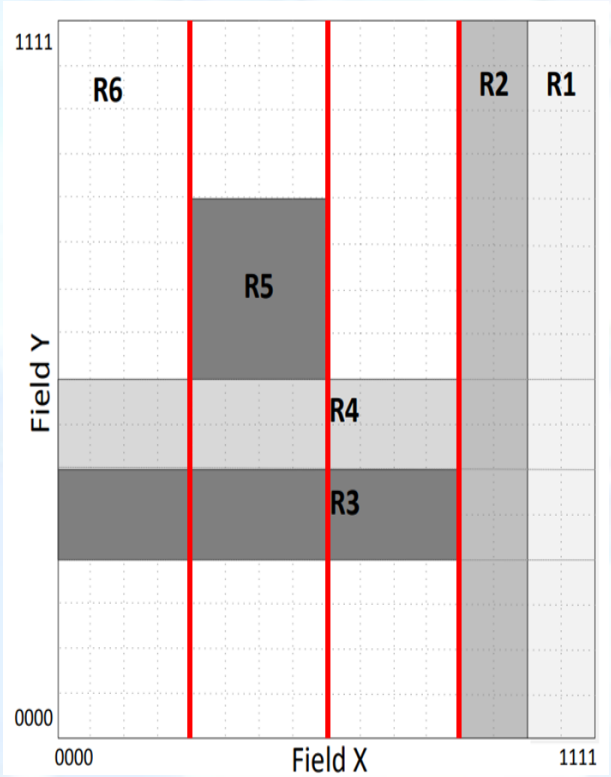
Rule #	Priority	Field X	Field Y	*Action*
R1	*1*	*111**	*	*drop*
R2	*2*	*110**	*	*forward*
R3	*3*	*	*010**	*enqueue*
R4	*4*	*	*011**	*modify*
R5	*5*	*01***	*10***	*forward*
R6	*6*	*	*	*drop*



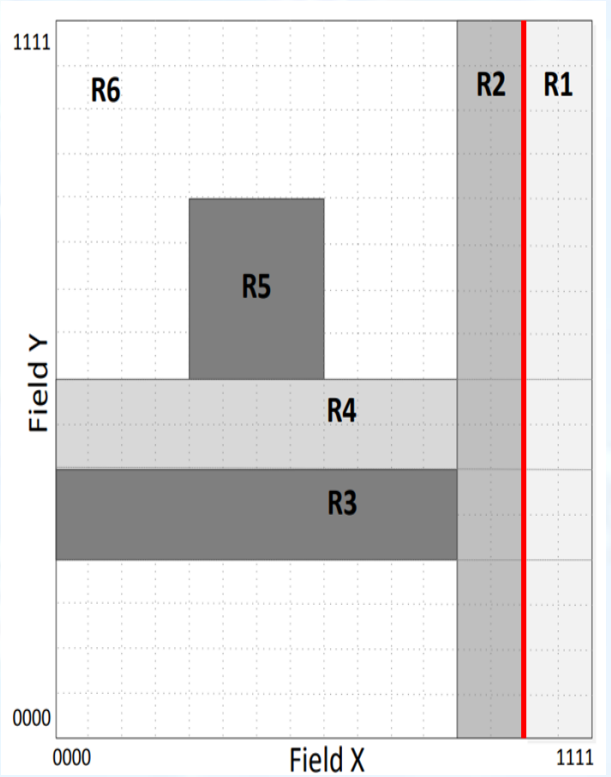
15

Подходы с деревьями поиска

Equal-sized cutting

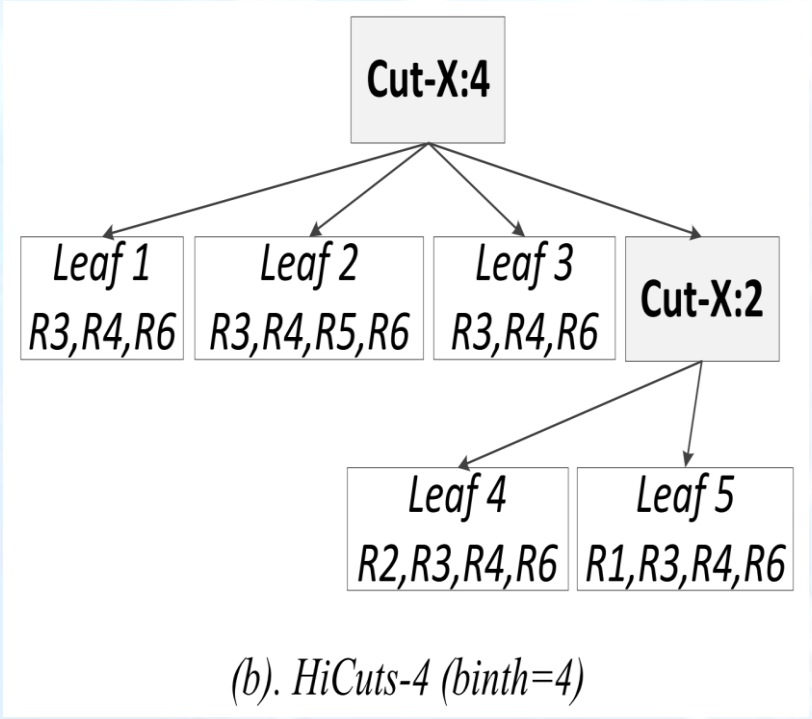


Equal-dense splitting

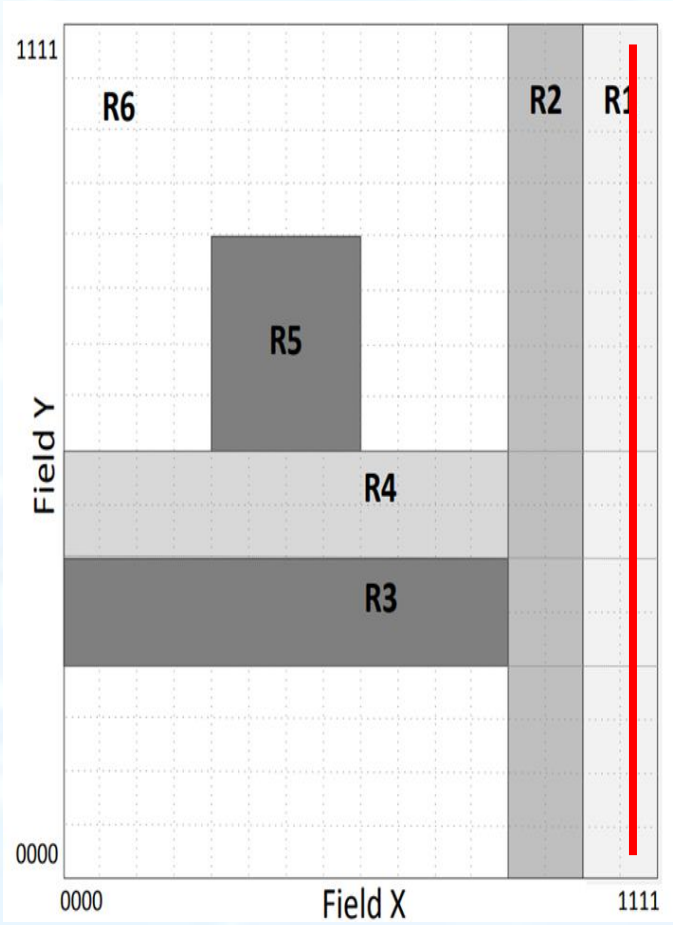


16

HiCuts



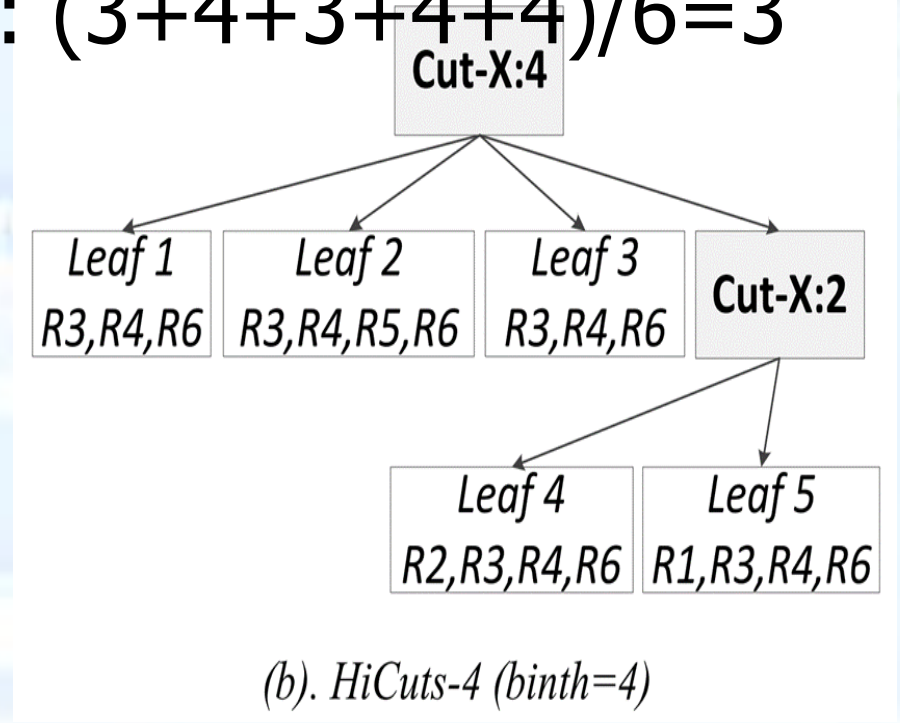
(b). HiCuts-4 (binth=4)



17

Дублирование правил (rule replication)

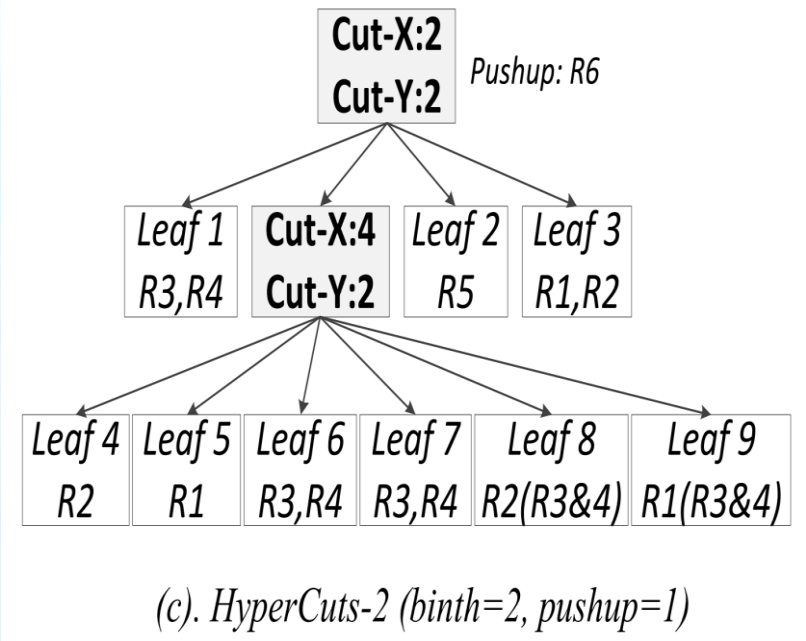
- Коэффициент дублирования правил (rule replication factor) = число правил в листьях / число правил в таблице
- Пример: (3+4+3+4+4)/6=3



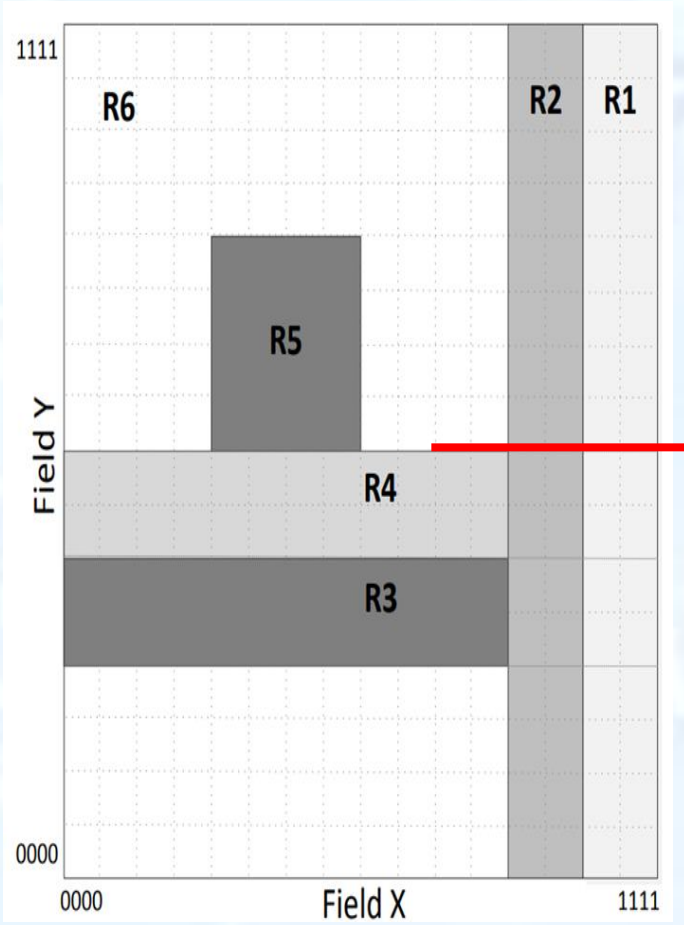
(b). HiCuts-4 (binth=4)

18

HyperCuts

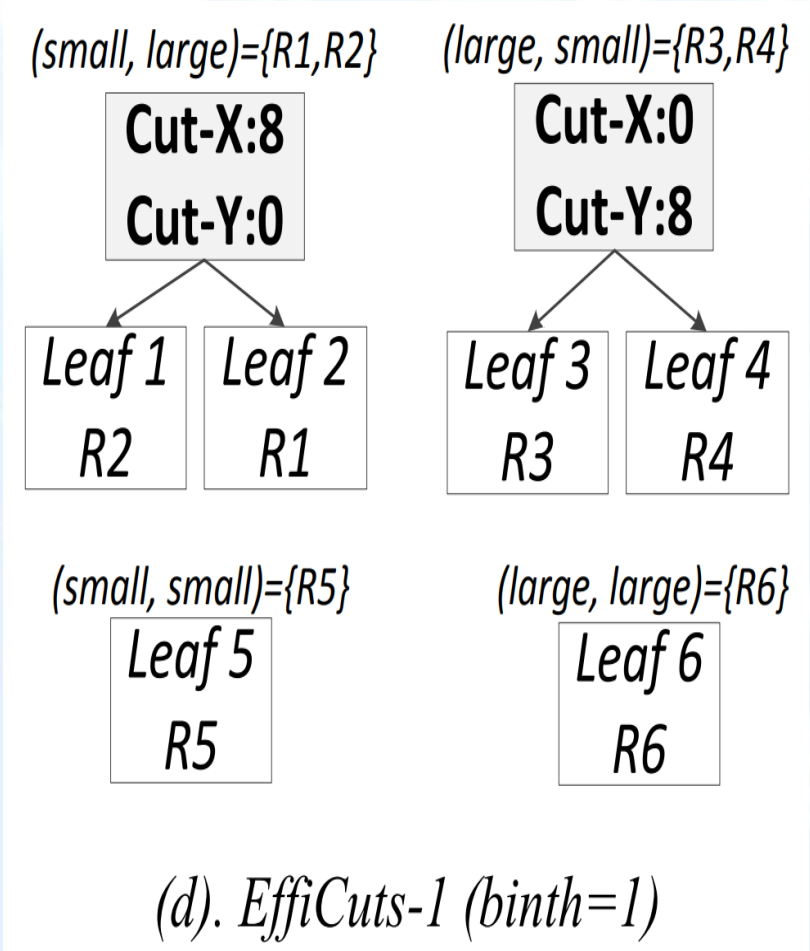


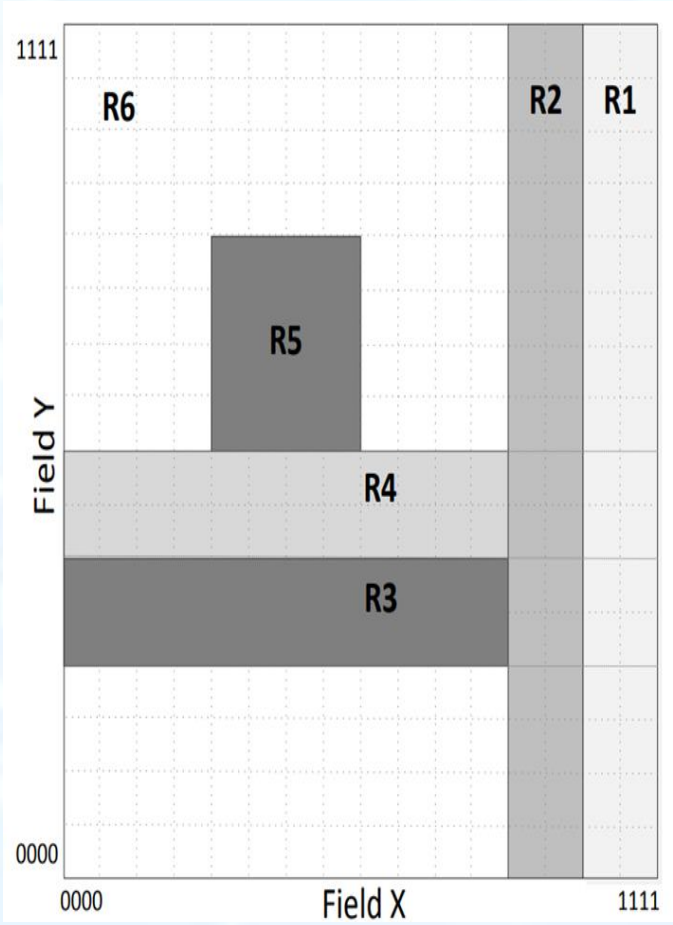
(c). HyperCuts-2 (binth=2, pushup=1)



19

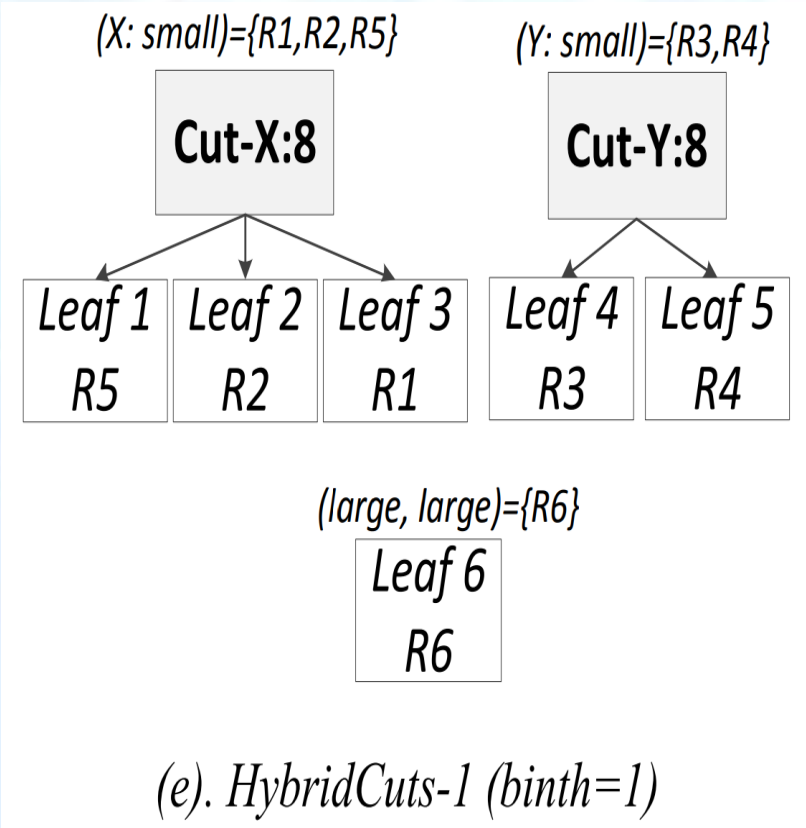
EffiCuts


(small, large)={R1,R2}
Cut-X:8
Cut-Y:0
Leaf 1
R2
Leaf 2
R1
(large, small)={R3,R4}
Cut-X:0
Cut-Y:8
Leaf 3
R3
Leaf 4
R4
(small, small)={R5}
Leaf 5
R5
(large, large)={R6}
Leaf 6
R6
(d). EffiCuts-1 (binth=1)

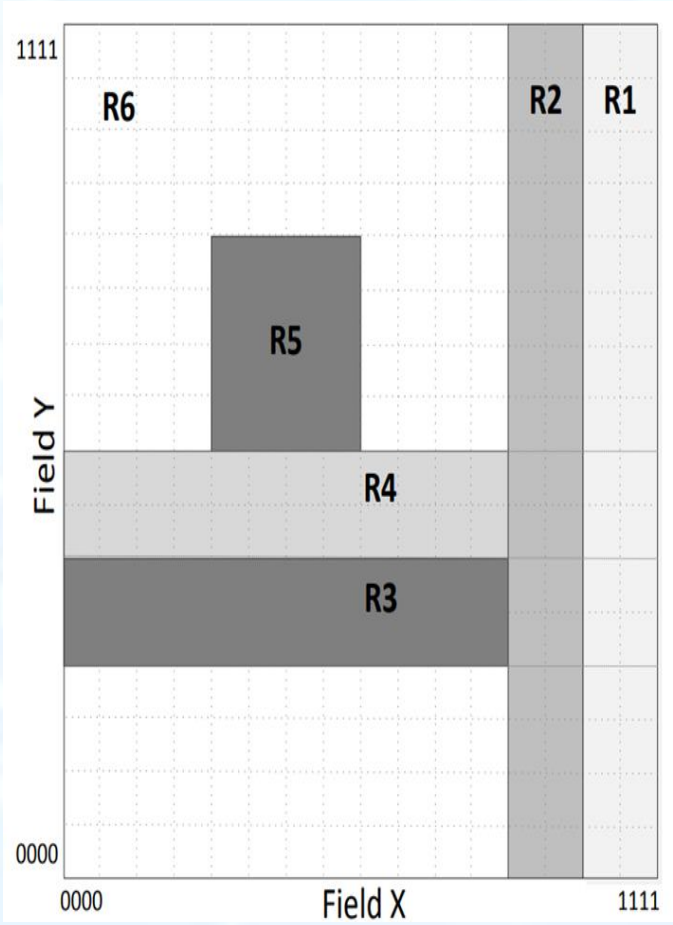

1111
R6
R2
R1
R5
R4
R3
Field Y
0000
0000
Field X
1111

20

HybridCuts

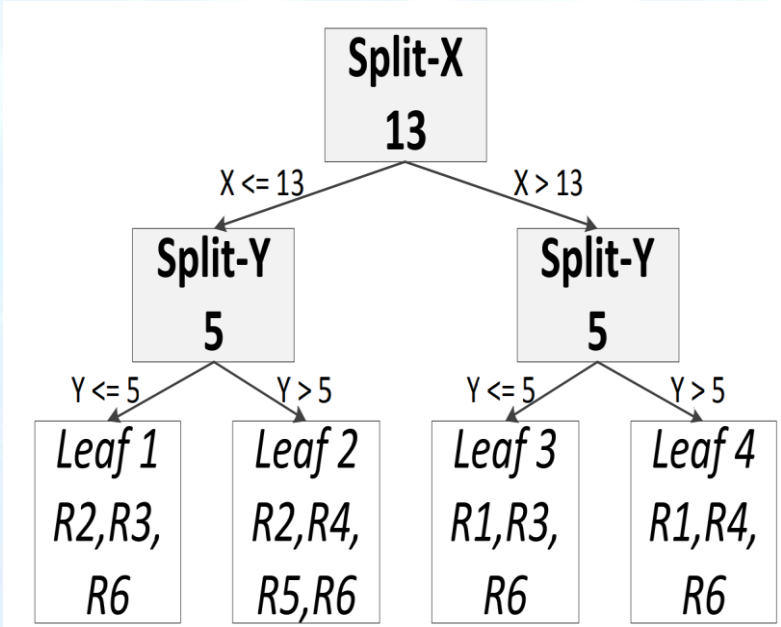


(e). HybridCuts-1 (binth=1)

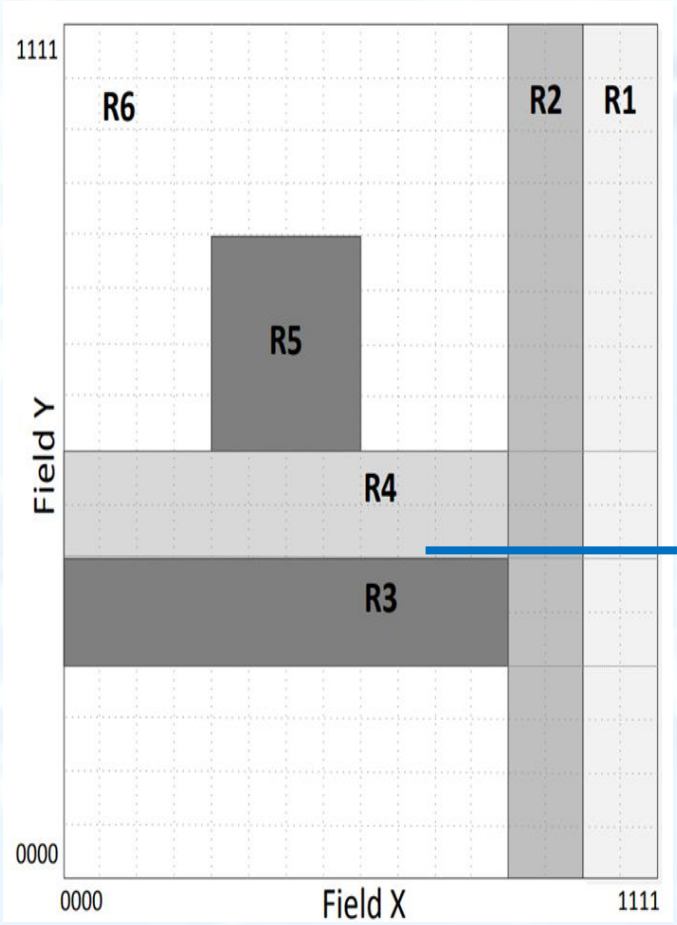


21

HyperSplit



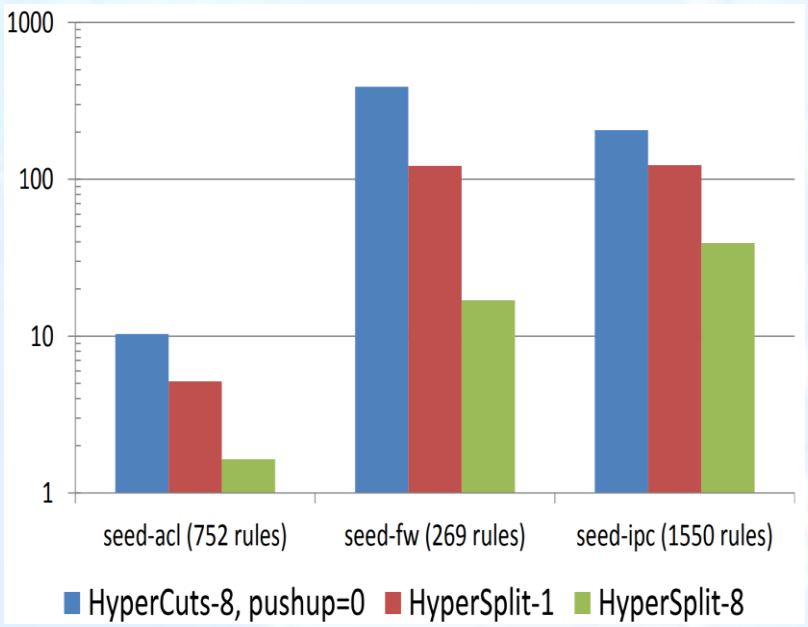
(f). HyperSplit-4 (binth=4)

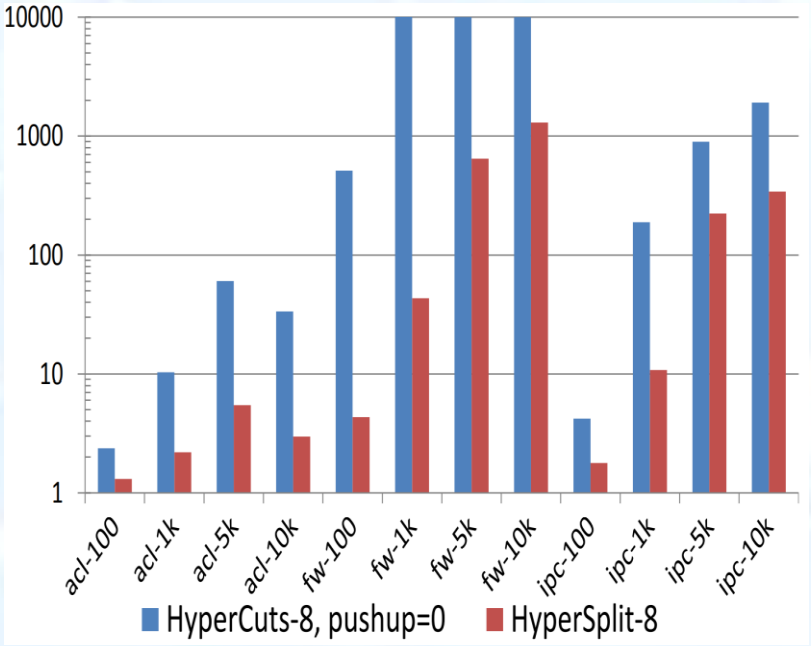


22

Дублирование правил

Коэффициент дублирования правил (rule replication factor) = число правил в листьях / число правил в таблице





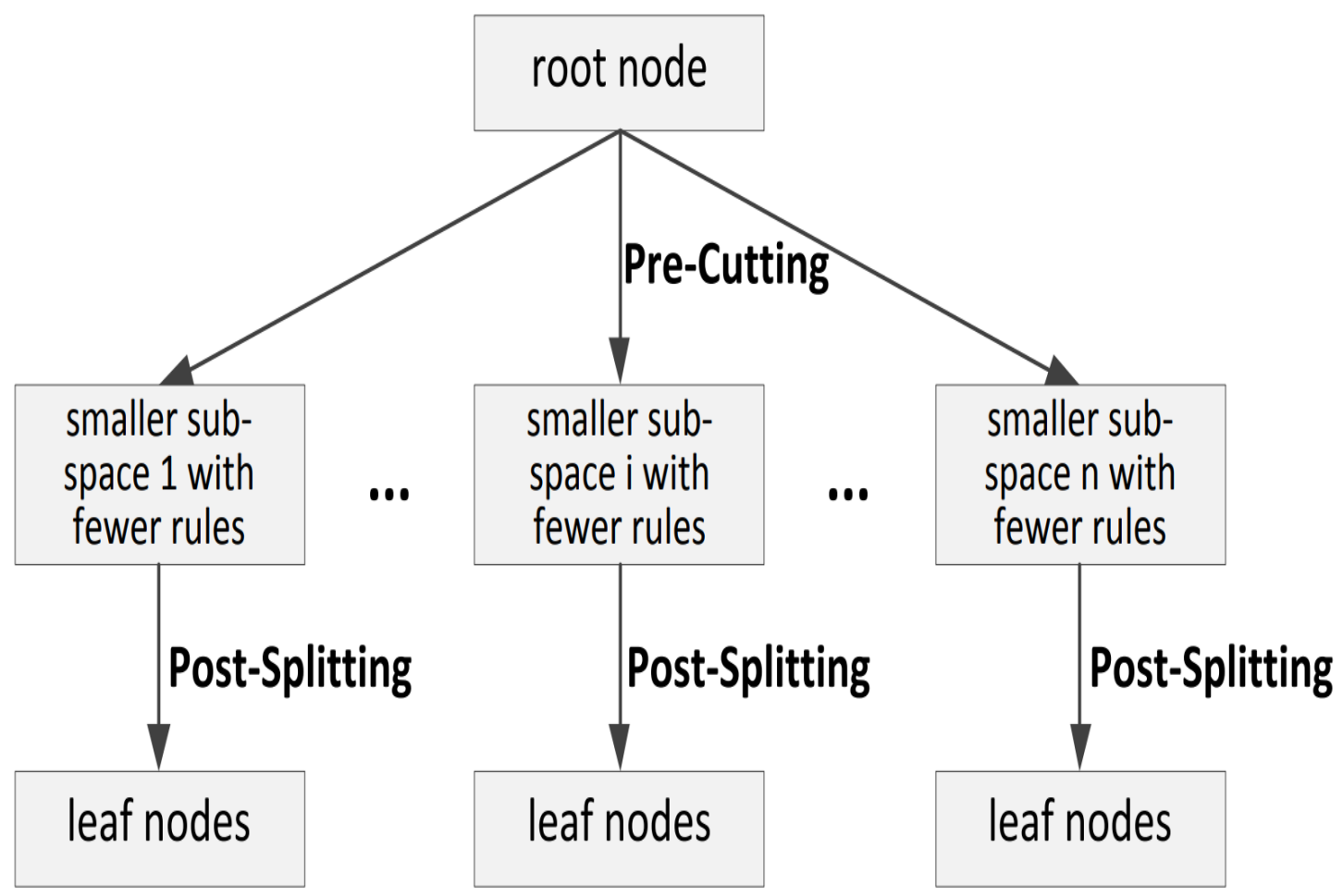
23

CutSplit

1. *Rule Set Partitioning:* разбиение множества правил:
   1. По «большим» полям
   2. По лучшим «маленьким» полям
2. *Faster Pre-Cutting:* пространство поиска для каждого подмножества разделяется по выбранным «маленьким» полям
3. *Explicit Post-Splitting*: поиск по оставшимся правилам

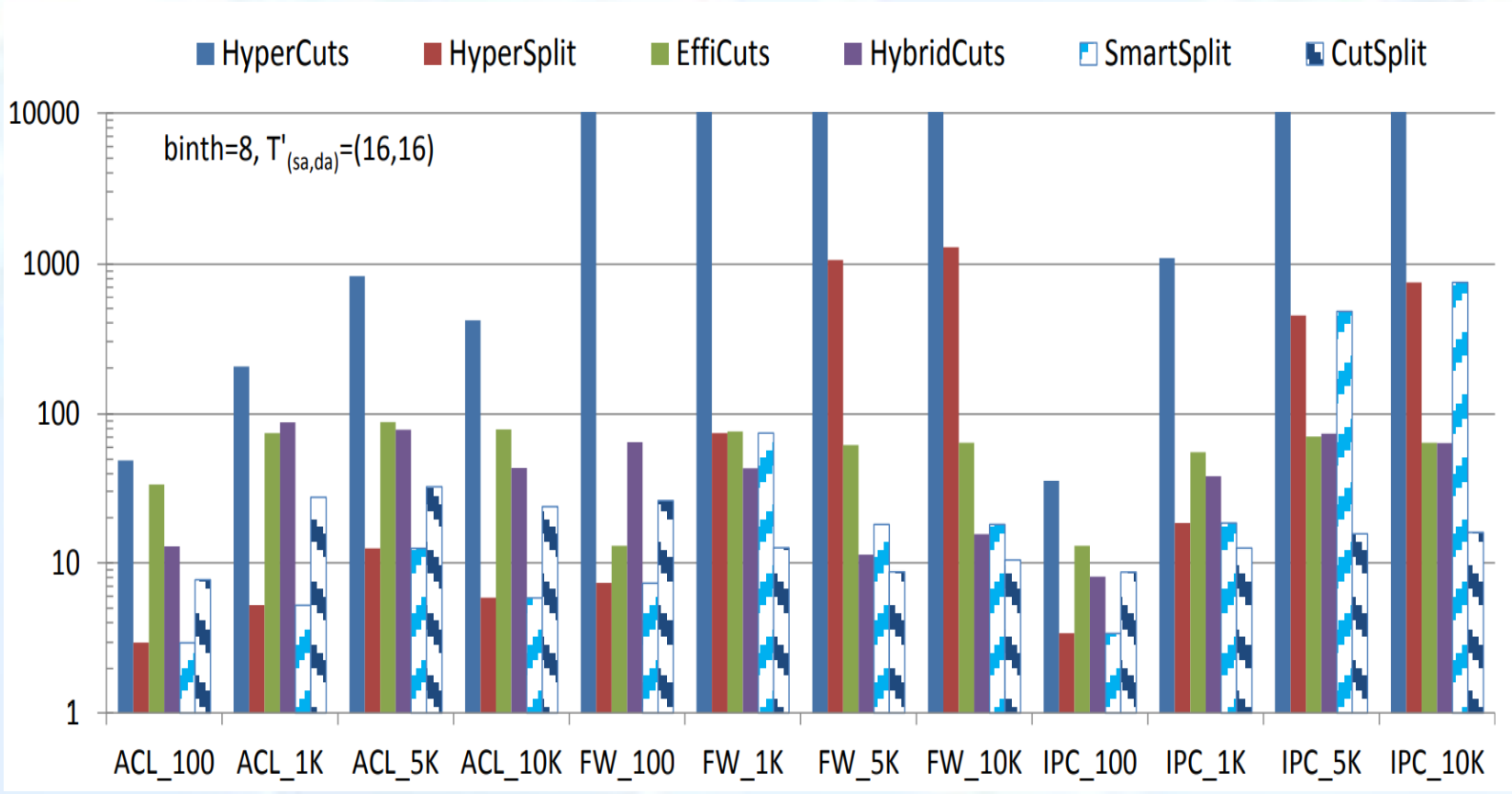
24

CutSplit


root node
Pre-Cutting
smaller sub-space 1 with fewer rules
...
smaller sub-space i with fewer rules
...
smaller sub-space n with fewer rules
Post-Splitting
Post-Splitting
Post-Splitting
leaf nodes
leaf nodes
leaf nodes

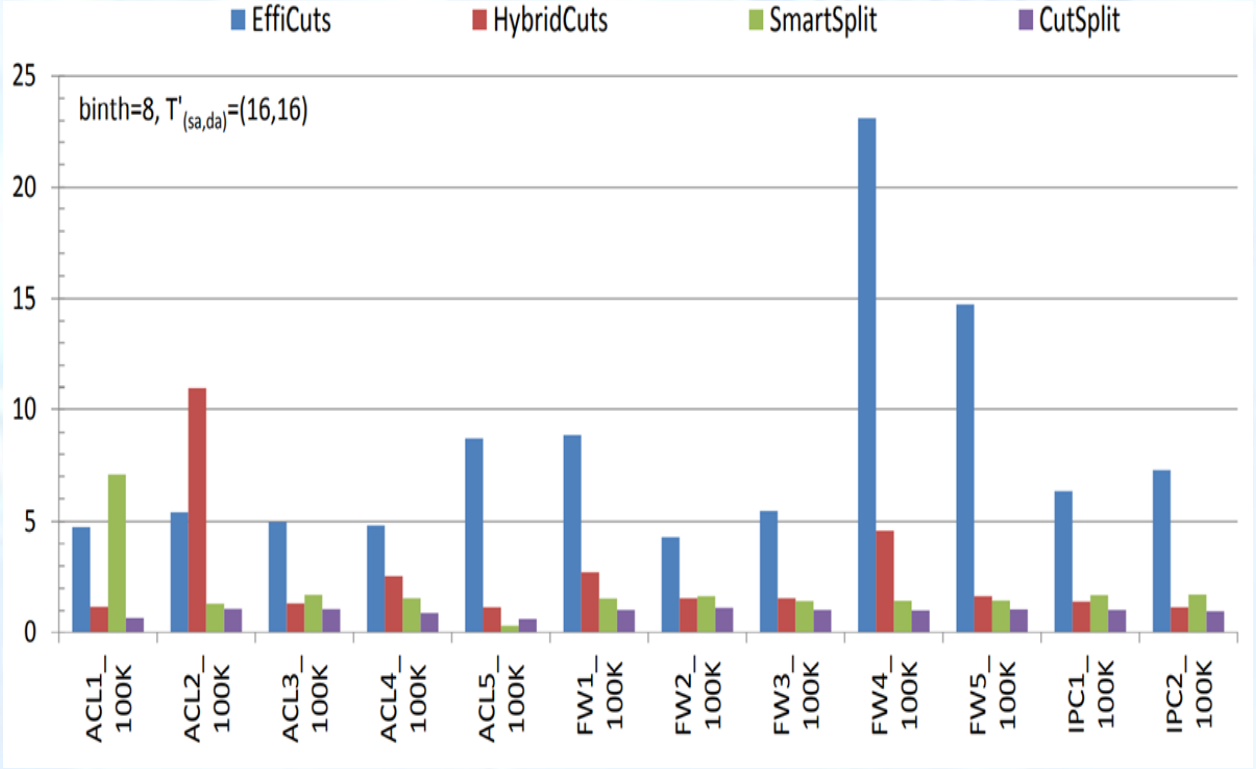
25

Потребление памяти, байт/правило



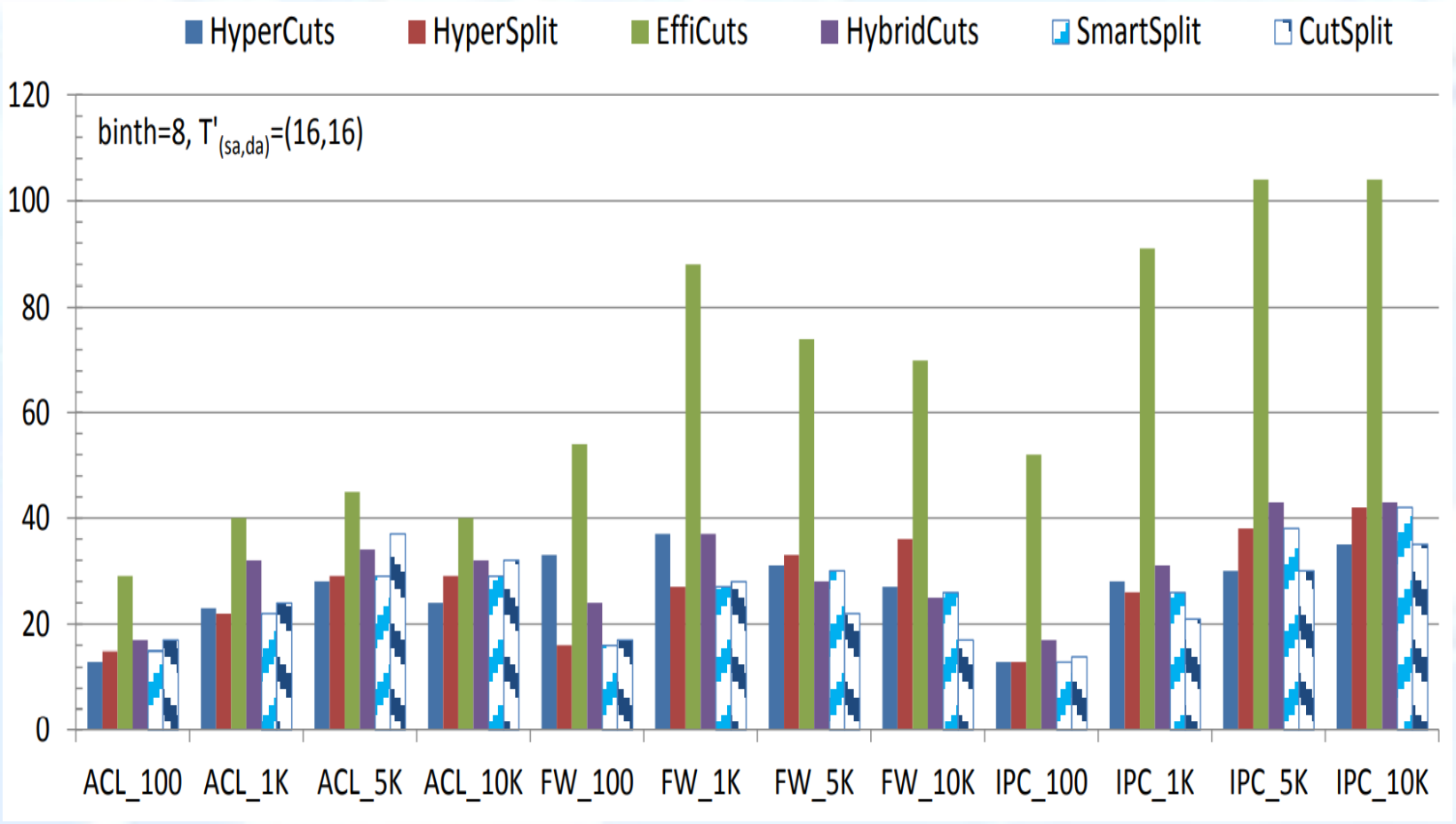
26

Потребление памяти, МБ (много правил)



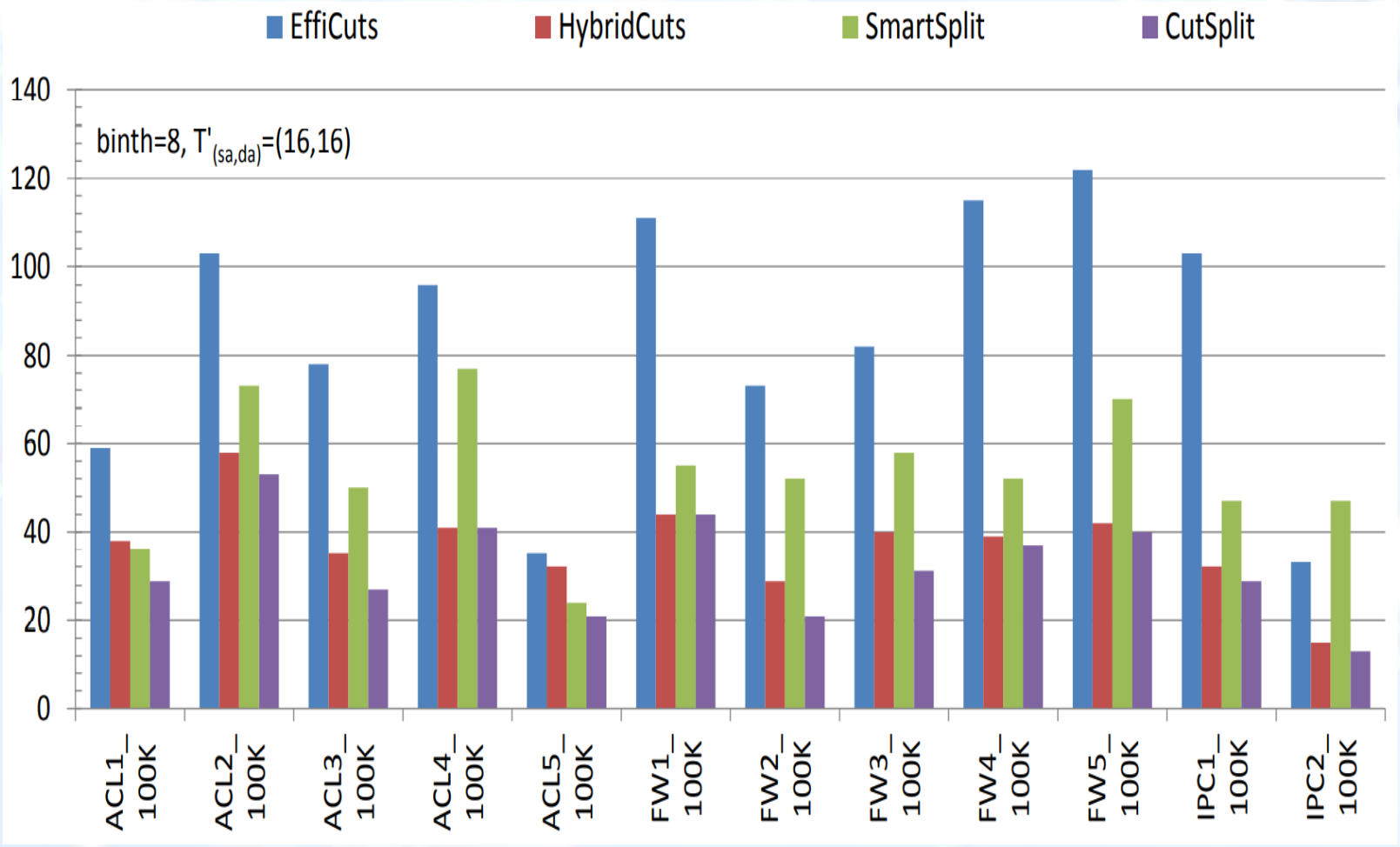
27

Число доступов в память (мало правил)



28

Число доступов в память (много правил)



29

Время построения структуры данных

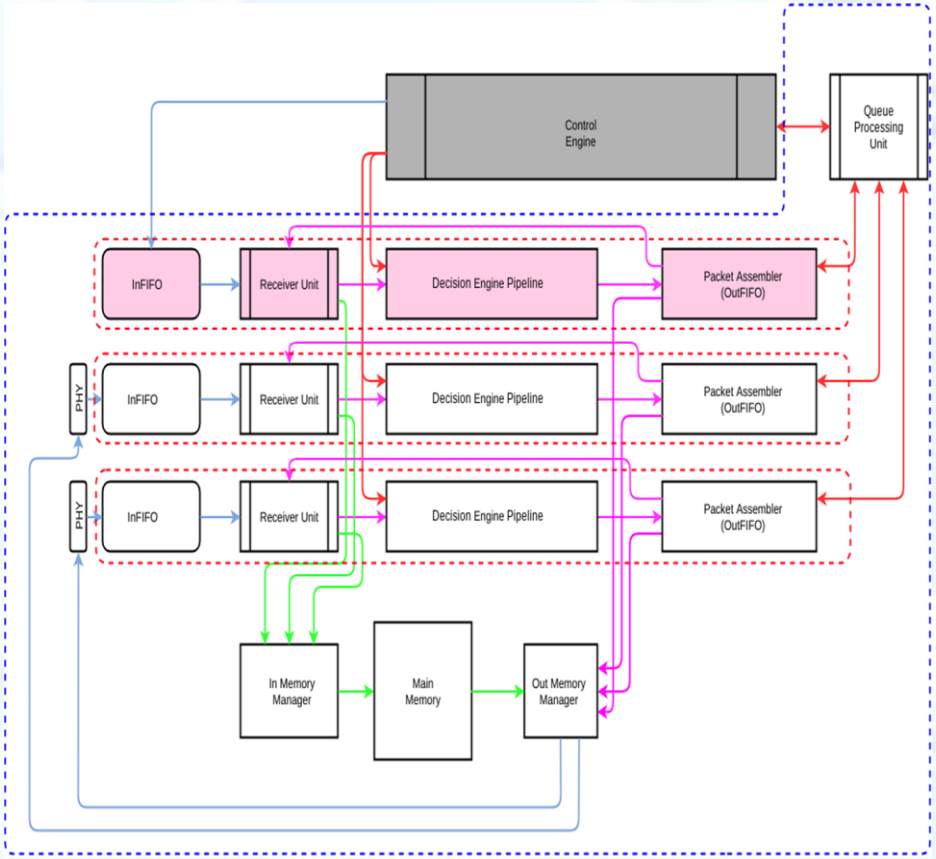
Rule set	EffiCuts	HybridCuts	SmartSplit	CutSplit
ACL1_100K	4784.4	183.1	632.5	11.7
ACL2_100K	8338.4	91.0	427.4	4.1
ACL3_100K	8453.6	148.6	6403.7	2.6
ACL4_100K	8232.6	161.8	3336.1	3.4
ACL5_100K	8905.3	138.5	2695.9	3.0
FW1_100K	4250.7	165.1	1392.1	3.0
FW2_100K	2842.2	161.9	1652.9	2.5
FW3_100K	4281.2	187.8	3855.4	3.0
FW4_100K	1662.1	280.3	4553.6	3.5
FW5_100K	3778.4	179.2	3212.7	2.7
IPC1_100K	8615.0	151.5	3133.4	2.6
IPC2_100K	6070.4	229.6	3187.9	2.6
MEAN	5851	173	2874	3.7

30

Применение в отечественном СПУ

Ограничения:

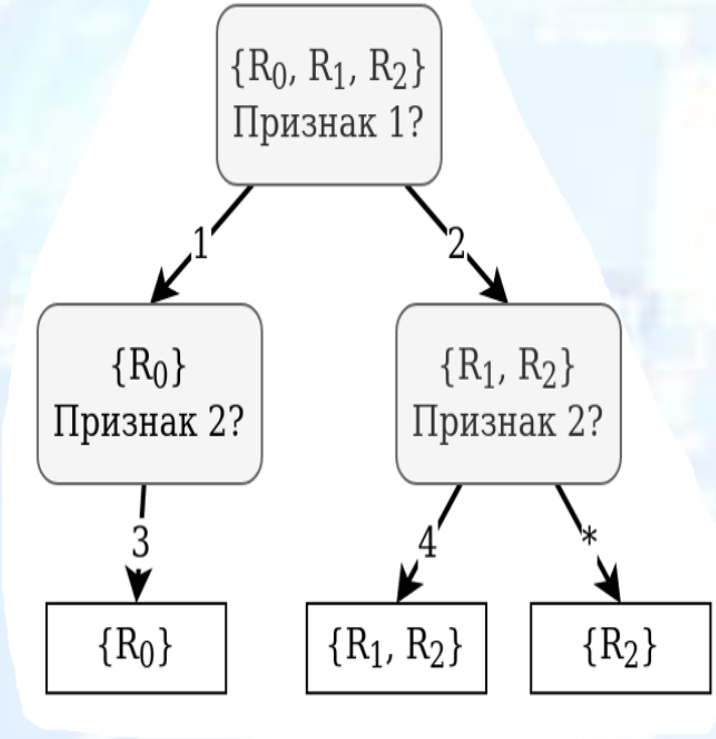
- Объем доступной памяти
- Поддержка хэш-функций
- Дополнительный объём памяти в процессе поиска



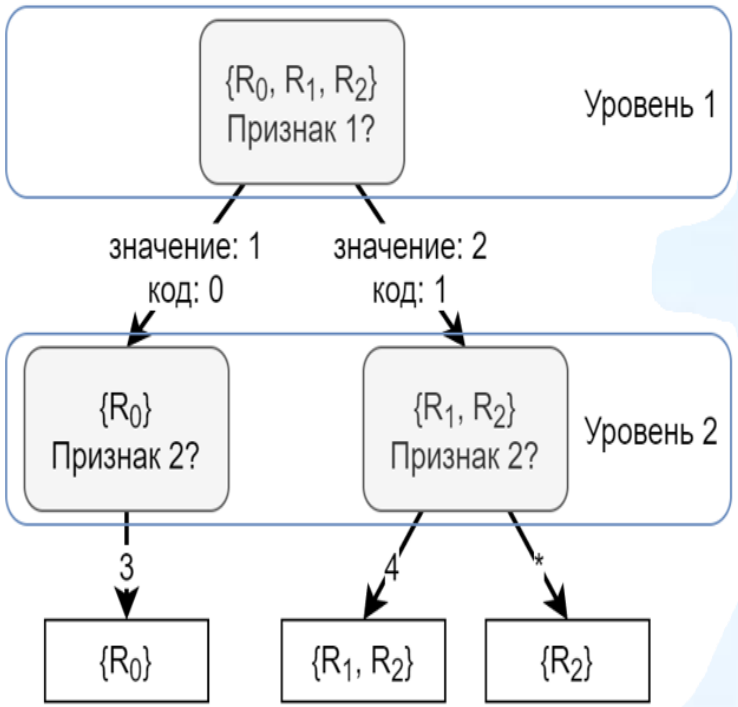
31

Подходы для отечественного СПУ

Дерево поиска



Дерево поиска с кодированием дуг



32

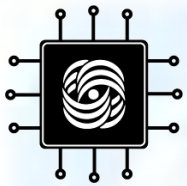
Заключение

- Рассмотрены требования к подходам для задачи классификации пакетов
- Рассмотрена схема декомпозиции задачи классификации пакетов по нескольким полям
- Рассмотрены различные подходы на основе деревьев поиска

33

Список литературы

1. W. Li, X. Li, H. Li and G. Xie, "CutSplit: A Decision-Tree Combining Cutting and Splitting for Scalable Packet Classification," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, USA, 2018, pp. 2645-2653.
2. D. E. Taylor and J. S. Turner, "Scalable packet classification using distributed crossproducing of field labels,"



Спасибо за внимание!