Simple and Space-Efficient Minimal Perfect Hash Functions

Fabiano C. Botelho
Department of Computer Science
Federal University of Minas Gerais, Brazil

Rasmus Pagh
Computational Logic and Algorithms Group
IT Univ of Copenhagen, DenMark

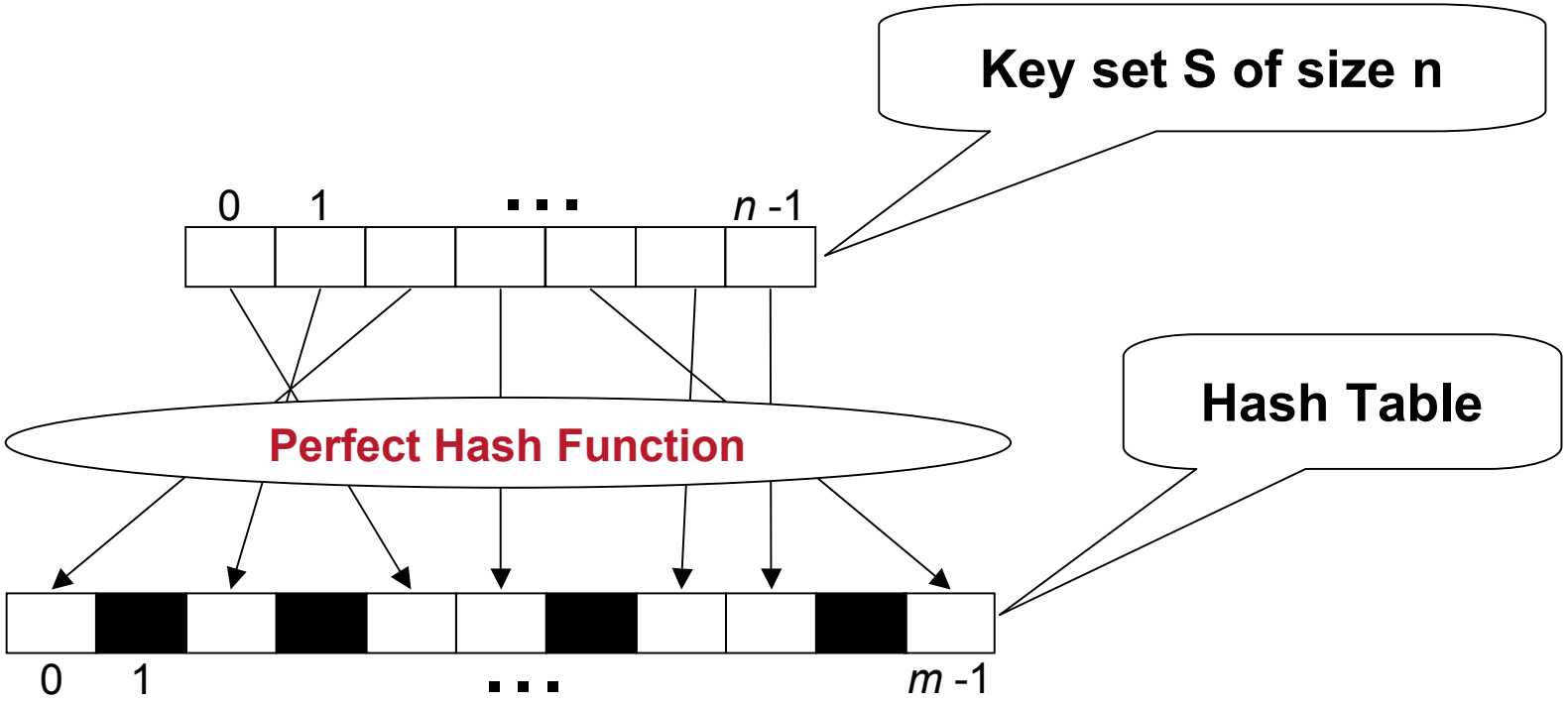
Nivio Ziviani
Department of Computer Science
Federal University of Minas Gerais, Brazil

What Is The Problem to Solve?

Design, analyze and implement MPHFs that:

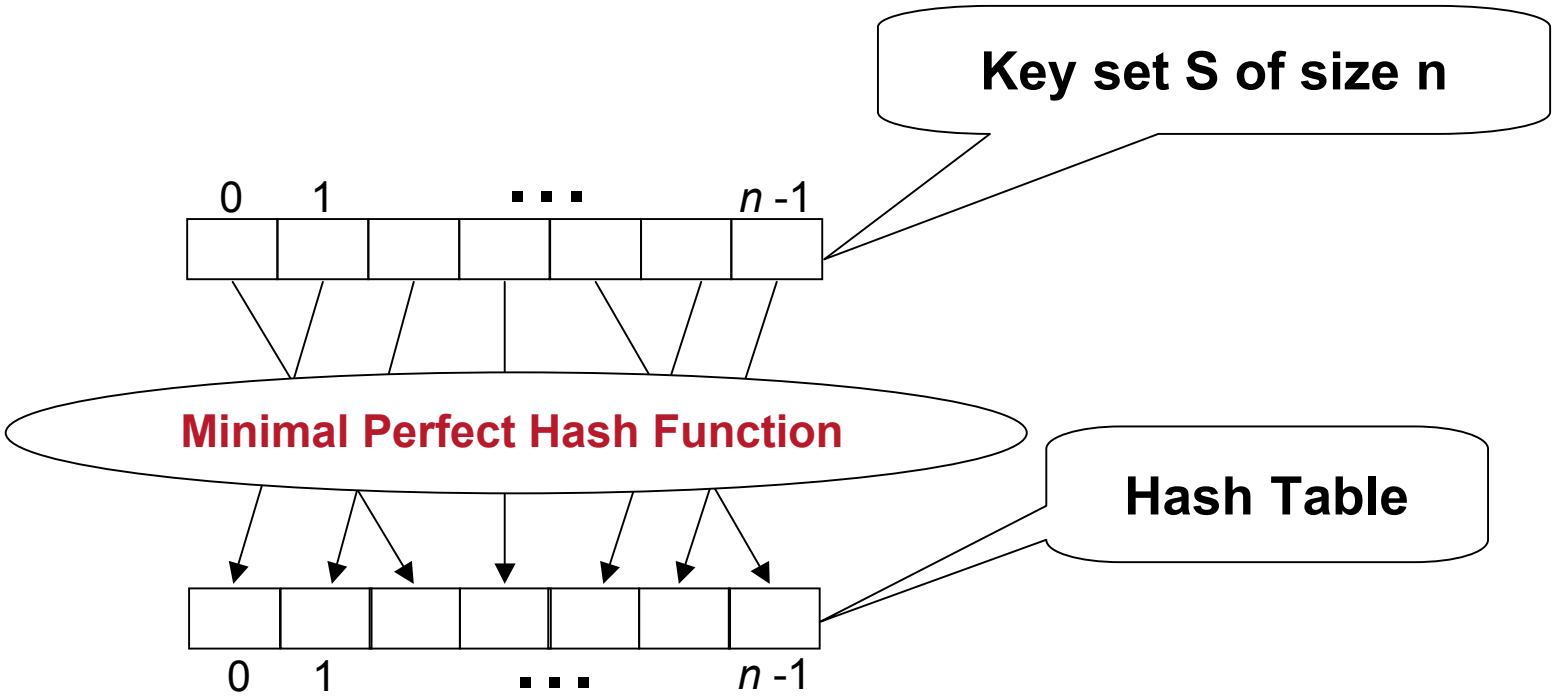
- Use space close to the optimal
- Faster to generate than the ones available in the literature
- Fast to compute
- Small memory to generate the functions

Perfect Hash Function



$$S \subseteq U, \text{where} |U| = u$$

Minimal Perfect Hash Function



$$S \subseteq U, \text{ where } |U| = u$$

Lower Bounds For Storage Space

- PHFs (m ≈ n): $\text{Storage Space} \geq \frac{n^2}{m} \log e$

- MPHFs (m = n): $\text{Storage Space} \geq n \log e$

$$\log e = 1.4427$$

Related Work

- Theoretical Results
- Practical Results
- Heuristics

Theoretical Results

Work	Gen. Time	Eval. Time	Size (bits)
Mehlhorn (1984)	Expon.	Expon.	O(n)
Schmidt and Siegel (1990)	Not analyzed	O(1)	O(n)
Hagerup and Thorup (2001)	O(n+log log u)	O(1)	O(n)

Practical Results

Work	Gen. Time	Eval. Time	Size (bits)
Czech, Havas and Majewski (1992)	O(n)	O(1)	O(n log n)
Majewski, Wormald, Havas and Czech (1996)	O(n)	O(1)	O(n log n)
Pagh (1999)	O(n)	O(1)	O(n log n)

Heuristics

Work	Application	Gen. Time	Eval. Time	Size (bits)
Fox, Chen and Heath (1992)	Index data in CD-ROM	Exp.	O(1)	O(n)
Lefebvre and Hoppe (2006)	Sparse spatial data	O(n)	O(1)	O(n)
Chang, Lin and Chou (2005, 2006)	Data mining	O(n)	O(1)	Not analyzed

Our Family of Algorithms

- Near-optimal space
- Evaluation in constant time
- Function generation in linear time
- Simple to describe and implement
- Algorithms in the literature with near-optimal space either:
  - Require exponential time for construction and evaluation, or
  - Use near-optimal space only asymptotically, for large *n*
- Acyclic random hypergraphs
  - Used before by Majewski et all (1996): O(n log n) bits
- We proceed differently: O(n) bits

  (we changed space complexity, close to theoretical lower bound)

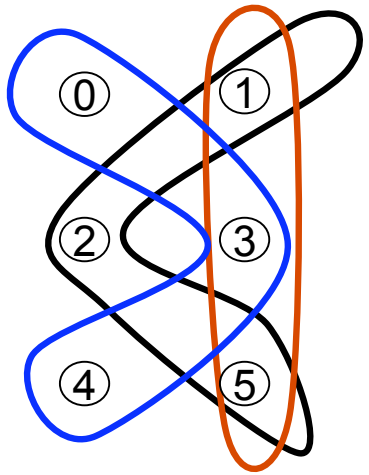
Our Family of Algorithms - Remark

- Chazelle et al (SODA 2004) presented a way of constructing PHFs that is equivalent to ours

- It is explained as a modification of the ``Bloomier Filter'' data structure, but they do not make explicit that a PHF is constructed

Random Hypergraphs (r-graphs)

- 3-graph:



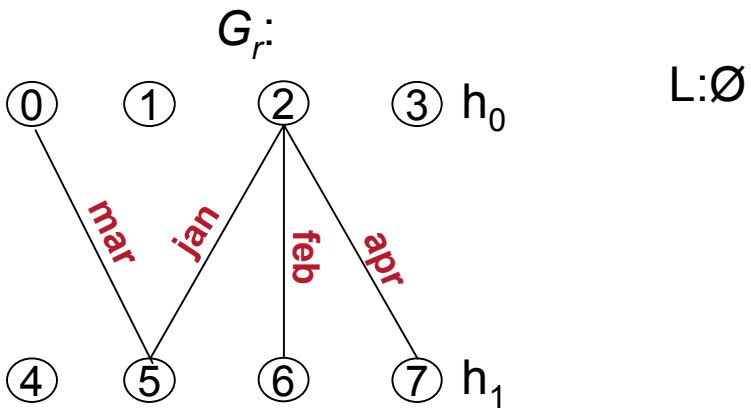
$h_0(jan) = 1 \quad h_1(jan) = 3 \quad h_2(jan) = 5$

$h_0(feb) = 1 \quad h_1(feb) = 2 \quad h_2(feb) = 5$

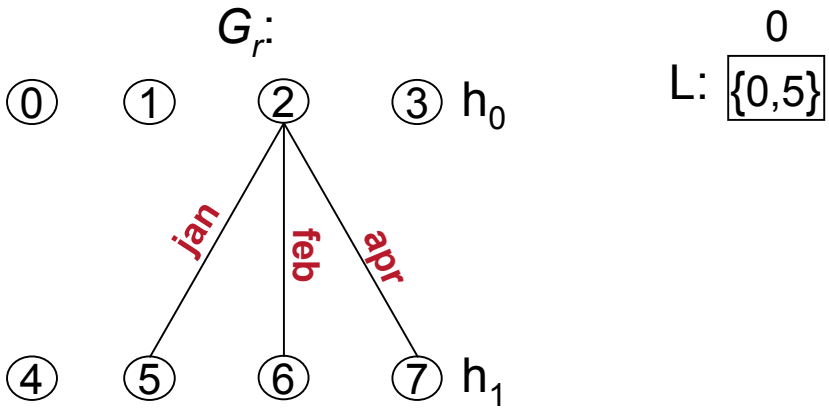
$h_0(mar) = 0 \quad h_1(mar) = 3 \quad h_2(mar) = 4$

- 3-graph is induced by three uniform hash functions
- Our best result uses 3-graphs

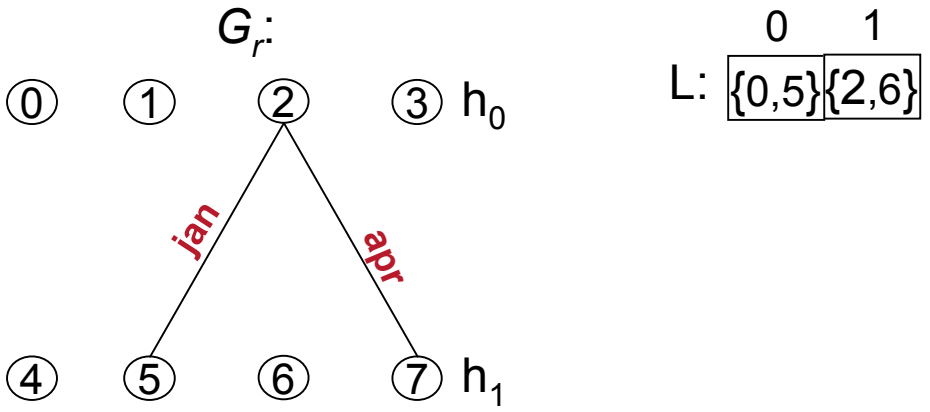
Acyclic 2-graph



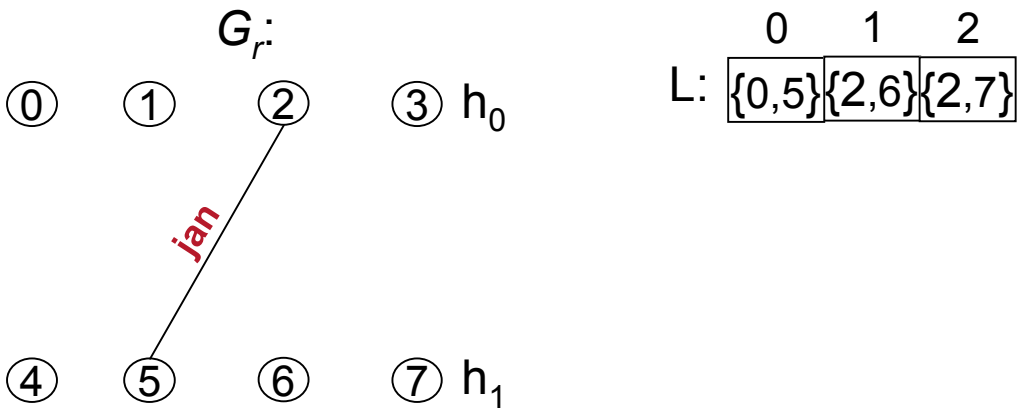
Acyclic 2-graph



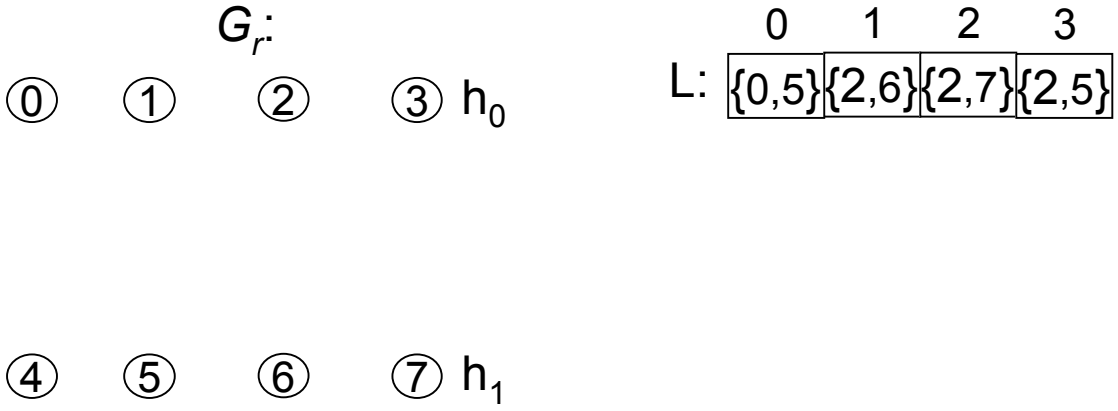
Acyclic 2-graph



Acyclic 2-graph

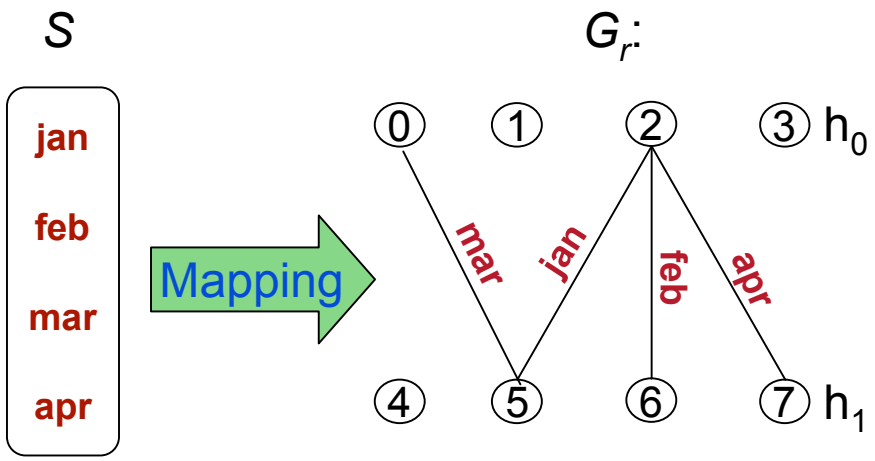


Acyclic 2-graph

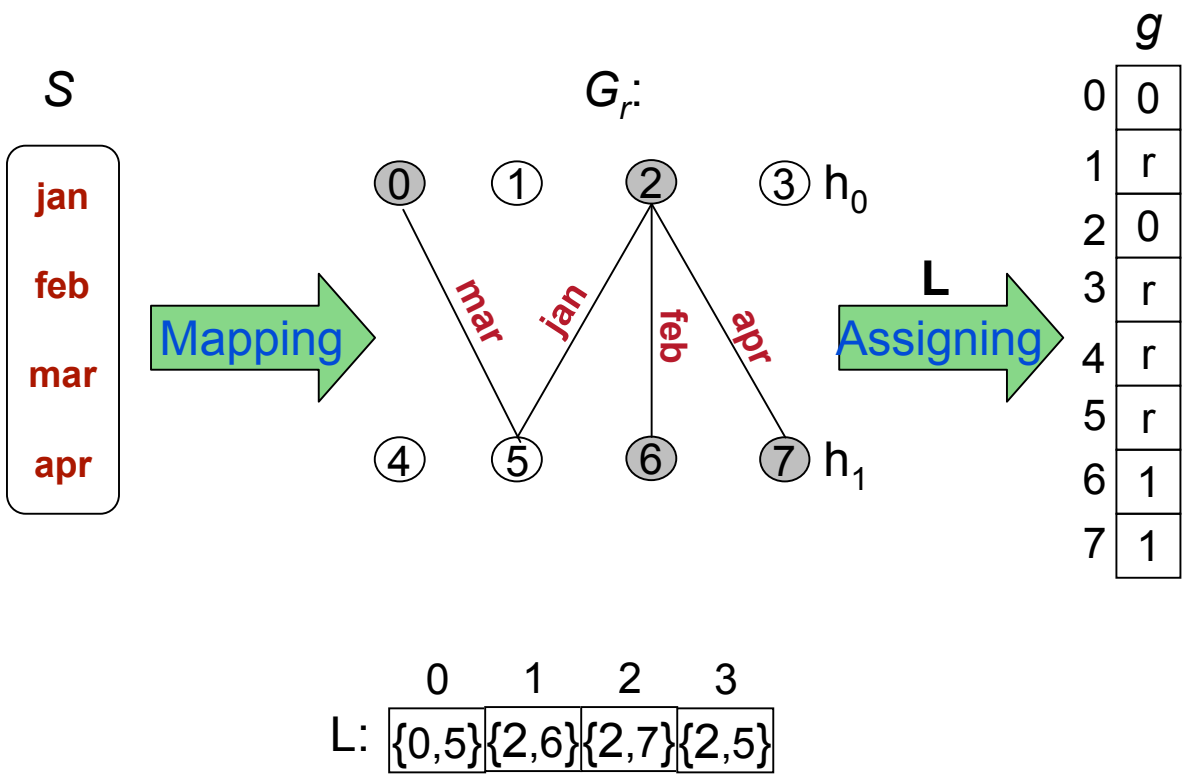


G_r is acyclic

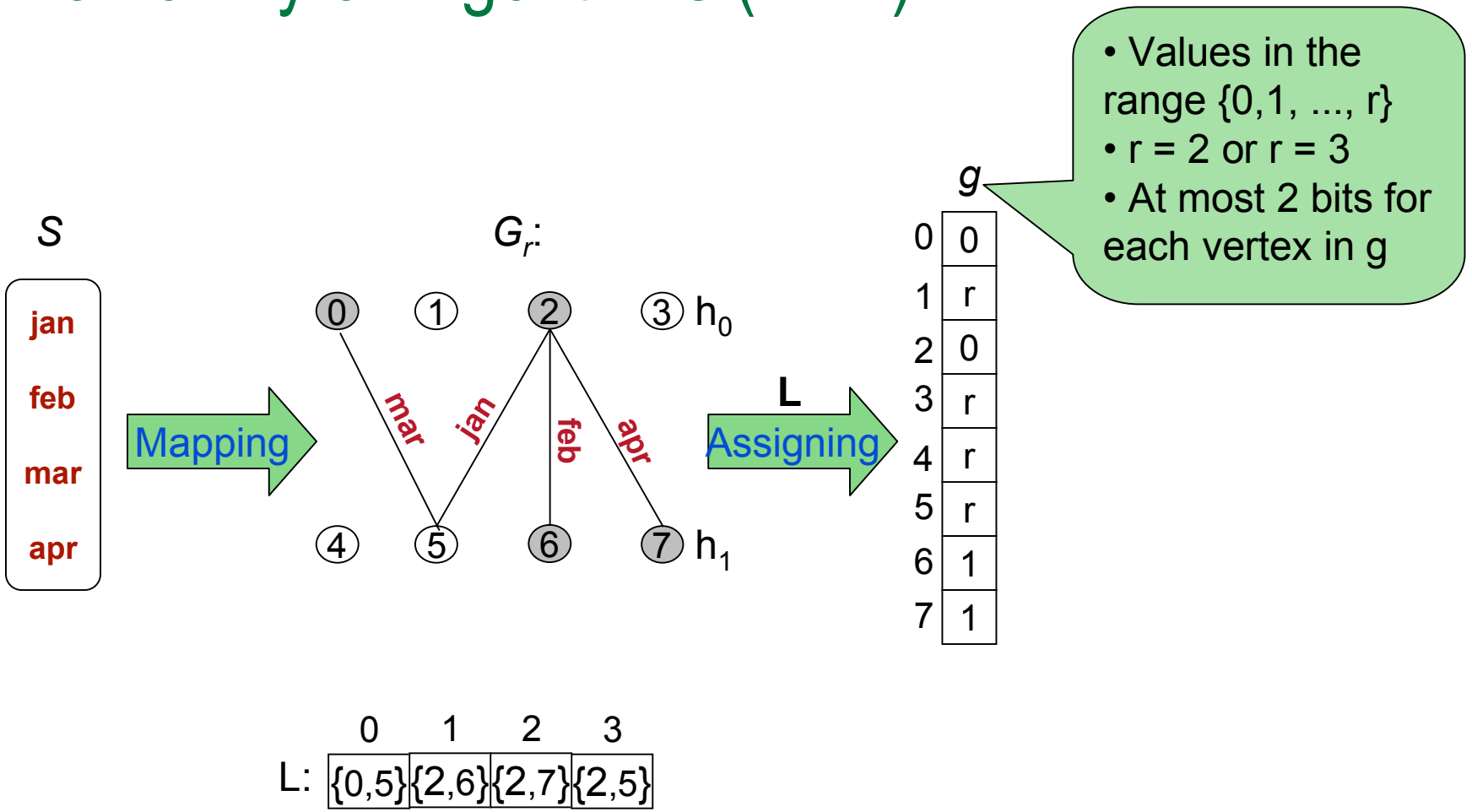
The Family of Algorithms (r = 2)



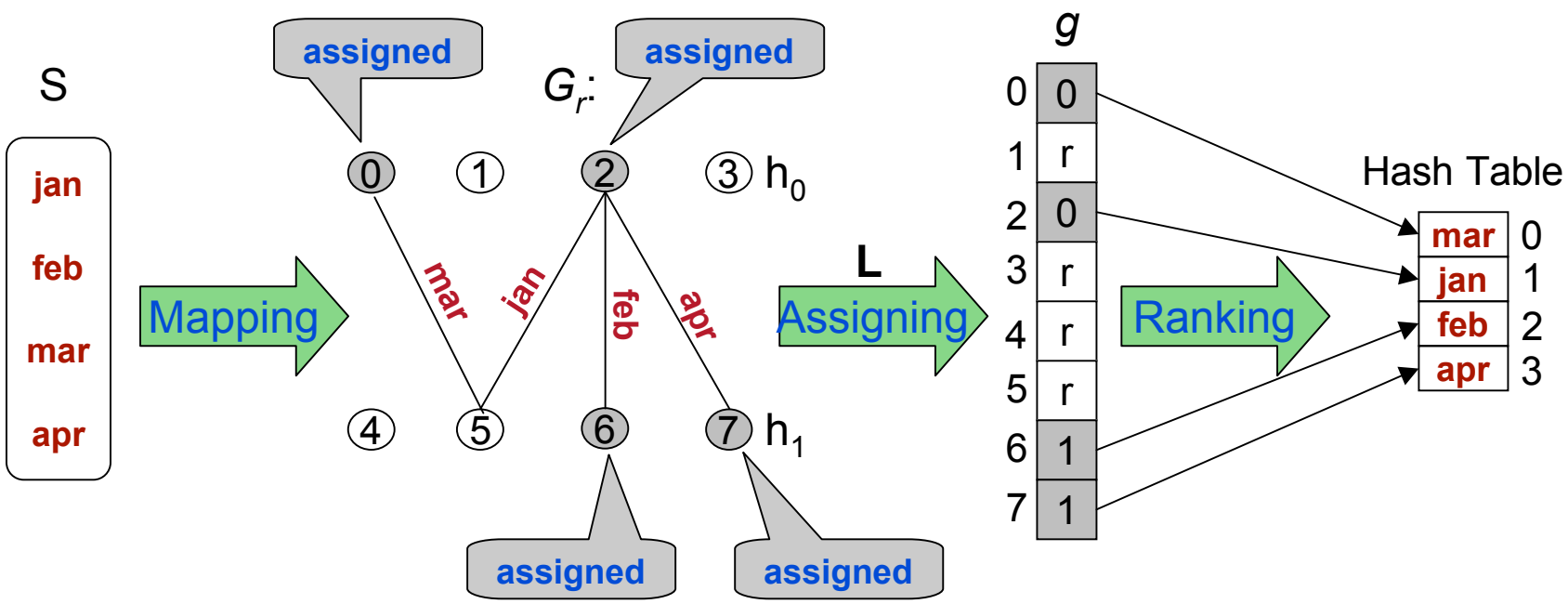
The Family of Algorithms (r = 2)



The Family of Algorithms (r = 2)



The Family of Algorithms (r = 2)



$phf(feb) = h_{i=1}(feb) = 6$

$i = (g(h_0(feb)) + g(h_1(feb))) \bmod \mathbf{r} = (g(2) + g(6)) \bmod \mathbf{2} = 1$

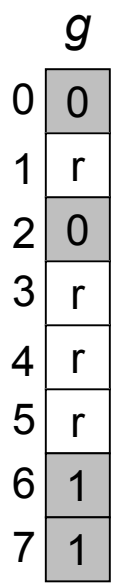
$mphf(feb) = rank(phf(feb)) = rank(6) = 2$

Use of Acyclic Random Hypergraphs

- Sufficient condition for the family of algorithms work (Majewski et al (1996))
- Repeatedly selects $h_0, h_1 \ldots, h_{r-1}$
- For r = 2, m=cn and c>2, $\Pr_a = \sqrt{1-(2/c)^2}$
  - For c = 2.09, $\Pr_a = 0.29$
- For r = 3 and c≥1.23: probability tends to 1
- Number of iterations is $1/\Pr_a$:
  - r = 2: 3.5 iterations
  - r = 3: 1.0 iteration

Space to Represent the Functions (r = 2)

- PHFs (ranking information not required):
  - g: [0,m-1] _ {0,1}
  - m = cn bits, c = 2.09 _ 2.09 n bits
- MPHFs (ranking information required):
  - g: [0,m-1] _ {0,1,2}
  - 2m + _m = (2+ _)cn bits
  - For c = 2.09 and _ = 0.125 _ 4.44 n bits
- Packed MPHFs (Range of size 3):
  - log 3 bits for each entry of g (arithmetic coding)
  - (log 3 + _)cn bits.
  - For c = 2.09 and _ = 0.125 _ 3.6 n bits.



Space to Represent the Functions (r = 3)

- PHFs (ranking information not required):
  - g: [0,m-1] _ {0,1,2}
  - m = cn bits, c = 1.23 _ 2.46 n bits
- Packed PHFs (Range of size 3):
  - log 3 bits for each entry of g (arithmetic coding)
  - (log 3) cn bits, c = 1.23 _ **1.95 n** bits
  - Optimal: **1.17n** bits
- MPHFs (ranking information required):
  - g: [0,m-1] _ {0,1,2,3}
  - 2m + _m = (2+ _)cn bits
  - For c = 1.23 and _ = 0.125 _ **2.62 n** bits
  - Optimal: **1.4427n** bits.

Experimental Results

- Metrics:
  - Generation time
  - Storage space
  - Evaluation time
- Collection:
  - 64 bytes long on average (URLs collected from the web)
- Experiments
  - Commodity PC with a cache of 2 Mbytes

Related Algorithms

- Botelho, Kohayakawa, Ziviani (2005) - BKZ
- Fox, Chen and Heath (1992) – FCH
- Czech, Havas and Majewski (1992) – CHM
- Majewski, Wormald, Havas and Czech (1996) – MWHC
- Pagh (1999) - PAGH

Generation Time and Storage Space

Algorithms		Generation Time (sec)	Storage Space	
			Bits/Key	Size (MB)
Ours	r = 2	19.49 ± 3.750	3.60	1.52
	r = 3	9.80 ± 0.007	2.62	1.11
BKZ		16.85 ± 1.85	21.76	9.19
FCH		5901.9 ± 1489.6	3.66	1.55
MWHC		10. 63 ± 0.09	26.76	11.30
PAGH		52.55 ± 2.66	44.16	18.65

n=3,541,615 keys

Evaluation Time

Algorithms		Evaluation Time (sec)
Ours	r = 2	2.63
	r = 3	2.73
BKZ		2.81
FCH		2.14
MWHC		2.85
PAGH		2.78

n=3,541,615 keys

Comparison of the Resulting PHFs and MPHFs

r	Packed	m	Generation Time (sec)	Evaluation Time (sec)	Storage Space	
					Bits/Key	Size (MB)
2	No	2.09n	19.41 ± 3.736	1.83	2.09	0.88
2	Yes	n	19.49 ± 3.750	2.63	3.60	1.52
3	No	1.23n	9.73 ± 0.009	2.16	2.46	1.04
3	Yes	1.23n	9.95 ± 0.009	2.14	1.95	0.82
3	No	n	9.80 ± 0.007	2.73	2.62	1.11

n=3,541,615 keys

Conclusions

- We have presented an efficient family of algorithms
  - Near space-optimal PHFs and MPHFs
- The algorithms are simpler and has much lower constant factors than existing theoretical results
- Outperforms the main practical general purpose algorithms found in the literature considering
  - generation time
  - storage space
- Implementation available at http://cmph.sf.net
  - LGPL free software license