

15-853: Algorithms in the Real World

Data Compression 4

15-853

Page 1

Compression Outline

Introduction: Lossy vs. Lossless, Benchmarks, ...

Information Theory: Entropy, etc.

Probability Coding: Huffman + Arithmetic Coding

Applications of Probability Coding: PPM + others

Lempel-Ziv Algorithms: LZ77, gzip, compress, ...

Other Lossless Algorithms: Burrows-Wheeler

➔ **Lossy algorithms for images:** JPEG, MPEG, ...

- Scalar and vector quantization

- JPEG and MPEG

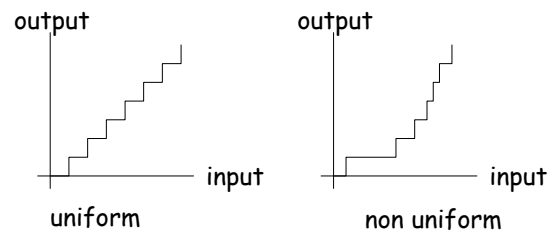
Compressing graphs and meshes: BBK

15-853

Page 2

Scalar Quantization

Quantize regions of values into a single value:

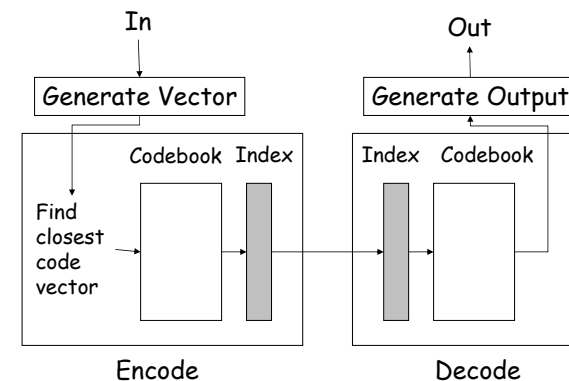


Can be used to reduce # of bits for a pixel

15-853

Page 3

Vector Quantization



15-853

Page 4

Vector Quantization

What do we use as vectors?

- Color (Red, Green, Blue)
 - Can be used, for example to reduce 24bits/pixel to 8bits/pixel
 - Used in some terminals to reduce data rate from the CPU (colormaps)
- K consecutive samples in audio
- Block of K pixels in an image

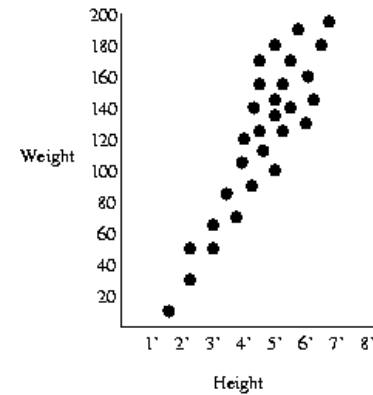
How do we decide on a codebook

- Typically done with **clustering**

15-853

Page 5

Vector Quantization: Example



15-853

Page 6

Linear Transform Coding

Want to encode values over a region of time or space

- Typically used for images or audio

Select a set of linear basis functions ϕ_i that span the space

- sin, cos, spherical harmonics, wavelets, ...
- Defined at discrete points

15-853

Page 7

Linear Transform Coding

$$\text{Coefficients: } \Theta_i = \sum_j x_j \phi_i(j) = \sum_j x_j a_{ij}$$

$$\Theta_i = i^{\text{th}} \text{ resulting coefficient}$$

$$x_j = j^{\text{th}} \text{ input value}$$

$$a_{ij} = ij^{\text{th}} \text{ transform coefficient} = \phi_i(j)$$

$$\Theta = A x$$

In matrix notation:

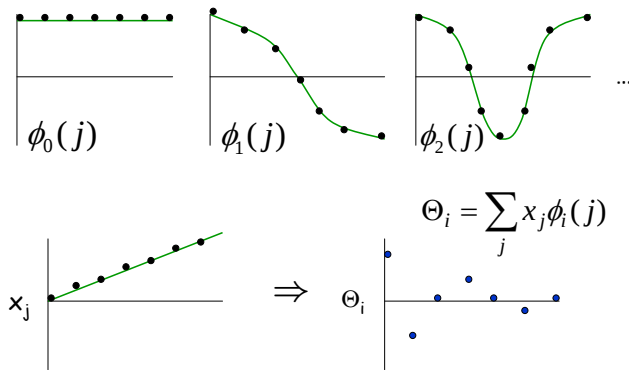
$$x = A^{-1} \Theta$$

Where A is an $n \times n$ matrix, and each row defines a basis function

15-853

Page 8

Example: Cosine Transform

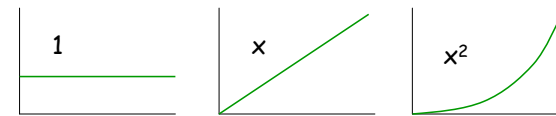


15-853

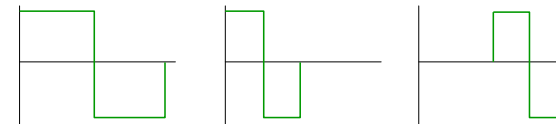
Page 9

Other Transforms

Polynomial:



Wavelet (Haar):



15-853

Page 10

How to Pick a Transform

Goals:

- Decorrelate
- Low coefficients for many terms
- Basis functions that can be ignored by perception

Why is using a Cosine or Fourier transform across a whole image bad?

How might we fix this?

15-853

Page 11

Usefulness of Transform

Typically transforms A are **orthonormal**: $A^{-1} = A^T$

Properties of orthonormal transforms:

$$\square \sum x^2 = \sum \Theta^2 \text{ (energy conservation)}$$

Would like to compact energy into as few coefficients as possible

$$G_{TC} = \frac{\frac{1}{n} \sum \sigma_i^2}{\left(\prod \sigma_i^2 \right)^{1/n}} \quad \text{(the **transform coding gain**)}$$

arithmetic mean/geometric mean

$$\sigma_i = (\Theta_i - \Theta_{av})$$

The higher the gain, the better the compression

15-853

Page 12

Case Study: JPEG

A nice example since it uses many techniques:

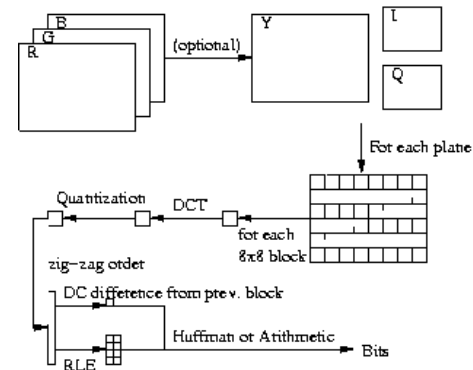
- Transform coding (Cosine transform)
- Scalar quantization
- Difference coding
- Run-length coding
- Huffman or arithmetic coding

JPEG (Joint Photographic Experts Group) was designed in **1991** for **lossy** and **lossless** compression of **color** or **grayscale images**. The lossless version is rarely used.
Can be adjusted for compression ratio (typically 10:1)

15-853

Page 13

JPEG in a Nutshell



15-853

Page 14

JPEG: Quantization Table

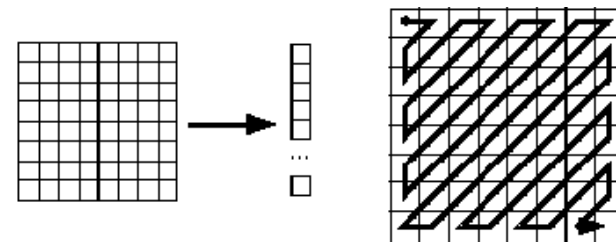
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Also divided through uniformly by a quality factor which is under control.

15-853

Page 15

JPEG: Block scanning order



Uses run-length coding for sequences of zeros

15-853

Page 16

JPEG: example



.125 bits/pixel (factor of 200)

15-853

Page 17

Case Study: MPEG

Pretty much JPEG with **interframe coding**

Three types of frames

- **I** = intra frame (aprox. JPEG) anchors
- **P** = predictive coded frames
- **B** = bidirectionally predictive coded frames

Example:

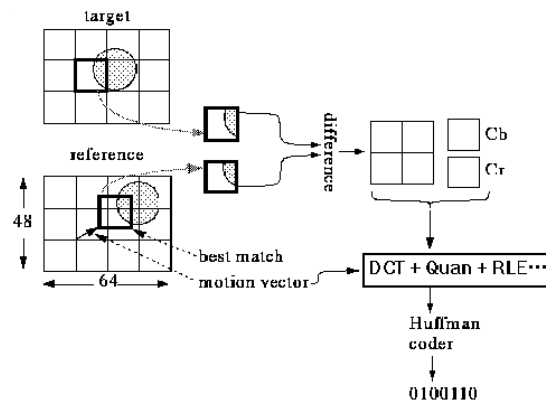
Type: I B B P B B P B B P B B I
Order: 1 3 4 2 6 7 5 9 10 8 12 13 11

I frames are used for random access.

15-853

Page 18

MPEG matching between frames



15-853

Page 19

MPEG: Compression Ratio

356 x 240 image

Type	Size	Compression
I	18KB	7/1
P	6KB	20/1
B	2.5KB	50/1
Average	4.8KB	27/1

30 frames/sec x 4.8KB/frame x 8 bits/byte
= 1.2 Mbits/sec + .25 Mbits/sec (stereo audio)

HDTV has 15x more pixels

= 18 Mbits/sec

15-853

Page 20

MPEG in the "real world"

- DVDs
 - Adds "encryption" and error correcting codes
- Direct broadcast satellite
- HDTV standard
 - Adds error correcting code on top
- Storage Tech "Media Vault"
 - Stores 25,000 movies

Encoding is much more expensive than encoding.
Still requires special purpose hardware for high resolution and good compression.

15-853

Page 21

Wavelet Compression

- A set of localized basis functions
- Avoids the need to block

"mother function" $\varphi(x)$

$$\varphi_{sl}(x) = \varphi(2^s x - l)$$

s = scale l = location

Requirements

$$\int_{-\infty}^{\infty} \varphi(x) dx = 0 \quad \text{and} \quad \int_{-\infty}^{\infty} |\varphi(x)|^2 dx < \infty$$

Many **mother** functions have been suggested.

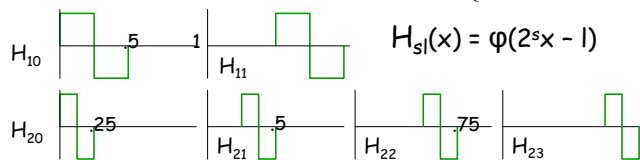
15-853

Page 22

Haar Wavelets

Most described, least used.

$$\varphi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

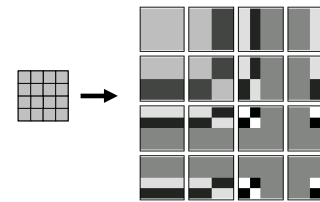


+ DC component = 2^{k+1} components

15-853

Page 23

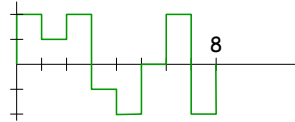
Haar Wavelet in 2d



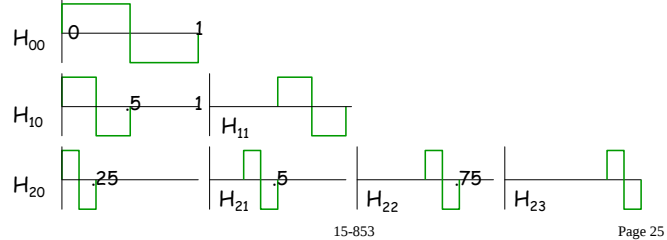
15-853

Page 24

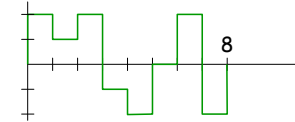
Discrete Haar Wavelet Transform



How do we convert this to the wavelet coefficients?



Discrete Haar Wavelet Transform



How do we convert this to the wavelet coefficients?

```
for (j = n/2; j >= 1; j = j/2) {
  for (i = 1; i < j; i++) {
    b[i] = (a[2i-1] + a[2i])/2;
    b[j+i] = (a[2i-1] - a[2i])/2; }
  a[1..2*j] = b[1..2*j]; }
```

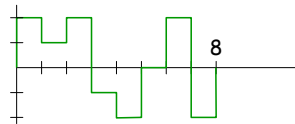
Averages (points to the first line of the inner loop)
Differences (points to the second line of the inner loop)

Linear time!

15-853

Page 26

Haar Wavelet Transform: example



$a = \begin{bmatrix} 2 & 1 & 2 & -1 & -2 & 0 & 2 & -2 \\ 1.5 & .5 & -1 & 0 & .5 & 1.5 & -1 & 2 \\ 1 & -.5 & .5 & -.5 \\ .25 & .75 \\ .25 & .75 & .5 & .5 & .5 & 1.5 & -1 & 2 \end{bmatrix}$

15-853

Page 27

Wavelet decomposition

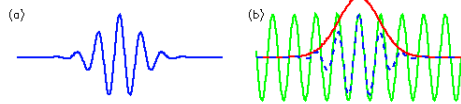


15-853

Page 28

Morlet Wavelet

$$\phi(x) = \text{Gaussian} \otimes \text{Cosine} = e^{-(x^2/2)} \cos(5x)$$

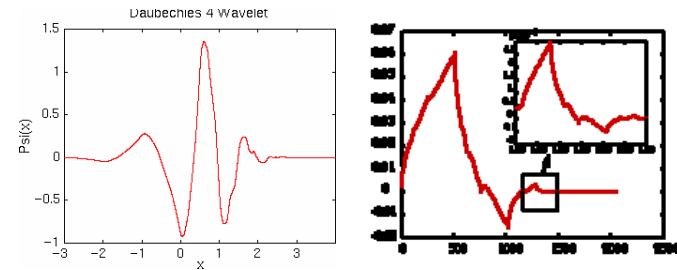


Corresponds to wavepackets in physics.

15-853

Page 29

Daubechies Wavelet



15-853

Page 30

JPEG2000

Overall Goals:

- High compression efficiency with good quality at compression ratios of .25bpp
- Handle large images (up to $2^{32} \times 2^{32}$)
- Progressive image transmission
 - Quality, resolution or region of interest
- Fast access to various points in compressed stream
- Pan and Zoom while only decompressing parts
- Error resilience

15-853

Page 31

JPEG2000: Outline

Main similarities with JPEG

- Separates into Y, I, Q color planes, and can downsample the I and Q planes
- Transform coding

Main differences with JPEG

- Wavelet transform
 - Daubechies 9-tap/7-tap (irreversible)
 - Daubechies 5-tap/3-tap (reversible)
- Many levels of hierarchy (resolution and spatial)
- Only arithmetic coding

15-853

Page 32

JPEG2000: 5-tap/3-tap

$$h[i] = a[2i-1] - (a[2i] + a[2i-2])/2;$$
$$l[i] = a[2i] + (h[i-1] + h[i] + 2)/2;$$

$h[i]$: is the "high pass" filter, ie, the **differences**
it depends on 3 values from a (3-tap)

$l[i]$: is the "low pass" filter, ie, the **averages**
it depends on 5 values from a (5-tap)

Need to deal with boundary effects.

This is reversible: assignment

15-853

Page 33

JPEG 2000: Outline

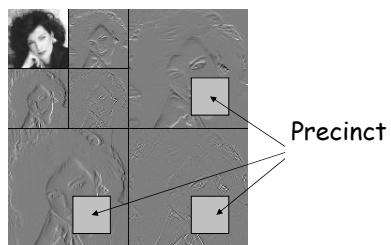
A spatial and resolution hierarchy

- **Tiles:** Makes it easy to decode sections of an image. For our purposes we can imagine the whole image as one tile.
- **Resolution Levels:** These are based on the wavelet transform. High-detail vs. Low detail.
- **Precinct Partitions:** Used within each resolution level to represent a region of space.
- **Code Blocks:** blocks within a precinct
- **Bit Planes:** ordering of significance of the bits

15-853

Page 34

JPEG2000: Precincts



15-853

Page 35

JPEG vs. JPEG2000



JPEG: .125bpp



JPEG2000: .125bpp

15-853

Page 36

Compression Outline

Introduction: Lossy vs. Lossless, Benchmarks, ...

Information Theory: Entropy, etc.

Probability Coding: Huffman + Arithmetic Coding

Applications of Probability Coding: PPM + others

Lempel-Ziv Algorithms: LZ77, gzip, compress, ...

Other Lossless Algorithms: Burrows-Wheeler

Lossy algorithms for images: JPEG, MPEG, ...

➡ **Compressing graphs and meshes:** BBK

15-853

Page 37

Compressing Structured Data

So far we have concentrated on Text and Images, compressing sound is also well understood.

What about various forms of "structured" data?

- Web indexes
- Triangulated meshes used in graphics
- Maps (mapquest on a palm)
- XML
- Databases



15-853

Page 38

Compressing Graphs

Goal: To represent large graphs compactly while supporting queries efficiently

- e.g., adjacency and neighbor queries
- want to do significantly better than adjacency lists (e.g. a factor of 10 less space, about the same time)

Applications:

- Large web graphs
- Large meshes
- Phone call graphs

15-853

Page 39

How to start?

Lower bound for n vertices and m edges?

1. If there are N possible graphs then we will need $\log N$ bits to distinguish them
2. in a directed graph there are n^2 possible edges (allowing self edges)
3. we can choose any m of them so $N = (n^2 \text{ choose } m)$
4. We will need $\log (n^2 \text{ choose } m) = O(m \log (n^2/m))$ bits in general

For sparse graphs ($m = kn$) this is hardly any better than adjacency lists (perhaps factor of 2 or 3).

15-853

Page 40

What now?

Are all graphs equally likely?

Are there properties that are common across "real world" graphs?

Consider

- link graphs of the web pages
- map graphs
- router graphs of the internet
- meshes used in simulations
- circuit graphs

LOCAL CONNECTIONS / SMALL SEPARATORS

15-853

Page 41

Edge Separators

An edge separator for (V, E) is a set of edges $E' \subset E$ whose removal partitions V into two components V_1 and V_2

Goals:

- **balanced** ($|V_1| \approx |V_2|$)
- **small** ($|E'|$ is small)

A class of graphs S satisfies a $f(n)$ -edge separator theorem if

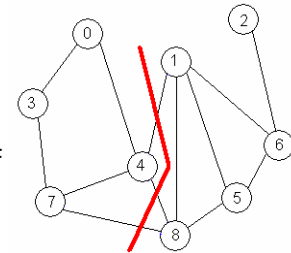
$\exists \alpha < 1, \beta > 0$

$\forall (V, E) \in S, \exists \text{ separator } E',$

$|E'| < \beta f(|V|),$

$|V_i| < \alpha |V|, i = 1, 2$

Can also define vertex separators.



15-853

Page 42

Separable Classes of Graphs

Planar graphs: $O(n^{1/2})$ separators

Well-shaped meshes in R^d : $O(n^{1-1/d})$ [Miller et al.]

Nearest-neighbor graphs

In practice, good separators from circuit graphs, street graphs, web connectivity graphs, router connectivity graphs

Note: All separable classes of graphs have bounded density (m is $O(n)$)

15-853

Page 43

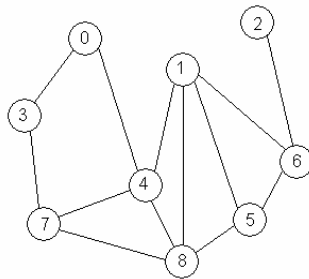
Main Ideas

- Number vertices so adjacent vertices have similar numbers
 - Use separators to do this
- Use difference coding on adjacency lists
- Use efficient data structure for indexing

15-853

Page 44

Compressed Adjacency Tables

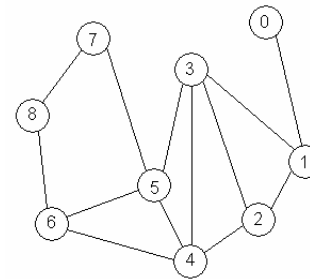


#	D	Neighbors	Differences
0	2	3 4	3 1
1	4	4 5 6 8	3 1 1 2
2	1	6	4
3	2	0 7	-3 7
4	4	0 1 7 8	-4 1 6 1
5	3	1 6 8	-4 5 2
6	3	1 2 5	-5 1 3
7	3	3 4 8	-4 1 4
8	4	1 4 5 7	-7 3 1 2

15-853

Page 45

Compressed Adjacency Tables



#	D	Neighbors	Differences
0	1	1	1
1	3	0 2 3	-1 2 1
2	3	1 3 4	-1 2 1
3	4	1 2 4 5	-1 1 2 1
4	4	2 3 5 6	-2 1 2 1
5	4	3 4 6 7	-2 1 2 1
6	3	4 5 8	-2 1 3
7	2	5 8	-2 3
8	2	6 7	-2 1

15-853

Page 46

Log-sized Codes

Log-sized code: Any prefix code that takes $O(\log(d))$ bits to represent an integer d .
Gamma code, delta code, skewed Bernoulli code

Example: Gamma code

Prefix: unary code for $\lfloor \log d \rfloor$
Suffix: binary code for $d - 2^{\lfloor \log d \rfloor}$
(binary code for d , except leading 1 is implied)

Decimal	Gamma
1	1
2	01 0
3	01 1
4	001 00
5	001 01
6	001 10
7	001 11
8	0001 000

15-853

Page 47

Difference Coding

For each vertex, encode:

- Degree
- Sign of first entry
- Differences in adjacency list

#	D	Differences
0	2	3 1

010 0 011 1
degree sign 3 1

Concatenate vertex encodings to encode the graph

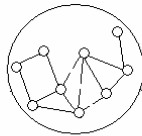
#	D	Differences
4	4	-4 1 6 1

00100 1 00100 1 00110 1
degree sign 4 1 6 1

15-853

Page 48

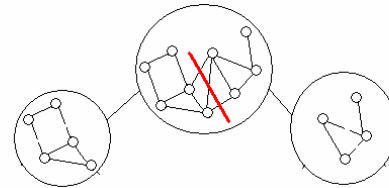
Renumbering with Edge Separators



15-853

Page 49

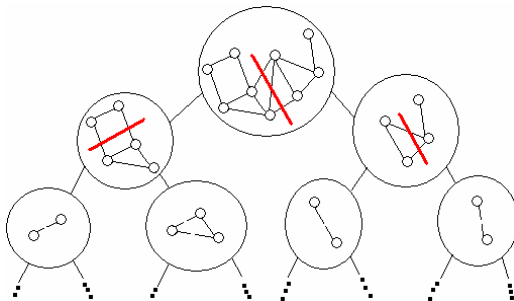
Renumbering with Edge Separators



15-853

Page 50

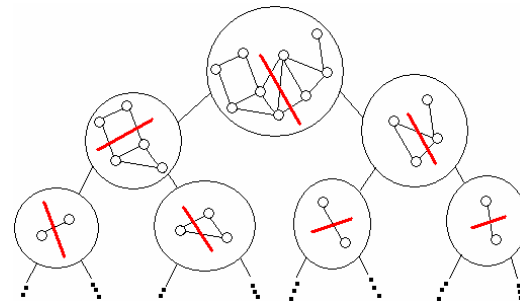
Renumbering with Edge Separators



15-853

Page 51

Renumbering with Edge Separators



15-853

Page 52

Theorem (edge separators)

Any class of graphs that allows $O(n^c)$ edge separators can be compressed to $O(n)$ bits with $O(1)$ access time using:

- Difference coded adjacency lists
- $O(n)$ -bit indexing structure

15-853

Page 53

Performance: Adjacency Table

	dfs		metis-cf		bu-bpq		bu-cf	
	T_d	Space	T/T_d	Space	T/T_d	Space	T/T_d	Space
auto	0.79	9.88	153.11	5.17	7.54	5.90	14.59	5.52
feocean	0.06	13.88	388.83	7.66	17.16	8.45	34.83	7.79
m14b	0.31	10.65	181.41	4.81	8.16	5.45	15.32	5.13
ibm17	0.44	13.01	136.43	6.18	11.0	6.79	20.25	6.64
ibm18	0.48	11.88	129.22	5.72	9.5	6.24	17.29	6.13
CA	0.76	8.41	382.67	4.38	14.61	4.90	35.21	4.29
PA	0.43	8.47	364.06	4.45	13.95	4.98	33.02	4.37
googleI	1.4	7.44	186.91	4.08	12.71	4.18	40.96	4.14
googleO	1.4	11.03	186.91	6.78	12.71	6.21	40.96	6.05
lucent	0.04	7.56	390.75	5.52	19.5	5.54	45.75	5.44
scan	0.12	8.00	280.25	5.94	23.33	5.76	81.75	5.66
Avg		10.02	252.78	5.52	13.85	5.86	34.54	5.56

Time is to create the structure, normalized to time for DFS

15-853

Page 54

Performance: Overall

Graph	Array		List		bu-cf/semi	
	time	space	time	space	time	space
auto	0.24	34.2	0.61	66.2	0.51	7.17
feocean	0.04	37.6	0.08	69.6	0.09	11.75
m14b	0.11	34.1	0.29	66.1	0.24	6.70
ibm17	0.15	33.3	0.40	65.3	0.34	7.72
ibm18	0.14	33.5	0.38	65.5	0.32	7.33
CA	0.34	43.4	0.56	75.4	0.58	11.66
PA	0.19	43.3	0.31	75.3	0.32	11.68
googleI	0.24	37.7	0.49	69.7	0.45	7.86
googleO	0.24	37.7	0.50	69.7	0.51	9.90
lucent	0.02	42.0	0.04	74.0	0.05	11.87
scan	0.04	43.4	0.06	75.4	0.08	12.85

time is for one DFS

Page 55

Conclusions

$O(n)$ -bit representation of separable graphs with $O(1)$ -time queries

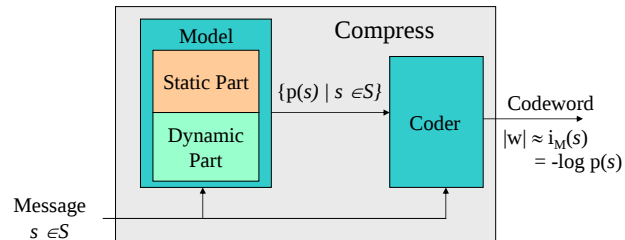
Space efficient and fast in practice for a wide variety of graphs.

15-853

Page 56

Compression Summary

Compression is all about **probabilities**



We want the model to skew the probabilities as much as possible (*i.e.*, decrease the **entropy**)

15-853

Page 57

Compression Summary

How do we figure out the probabilities

- Transformations that skew them
 - Guess value and code difference
 - Move to front for temporal locality
 - Run-length
 - Linear transforms (Cosine, Wavelet)
 - Renumber (graph compression)
- Conditional probabilities
 - Neighboring context

In practice one almost always uses a combination of techniques

15-853

Page 58