Algorithms for Infinite Huffman-Codes

Mordecai Golin   Kin-Keung Ma

golin@cs.ust.hk   kkma@cs.ust.hk

Hong Kong University of Science and Technology

Introduction

Consider distribution

$$\mathcal{P} = \{p_i\}_{i=0}^{n-1}, \quad p_0 \geq p_1 \geq p_2 \geq \cdots \geq p_{n-1}.$$

The *Huffman Encoding problem* is to associate a prefix-free set of n binary words $\{w_i\}_{i=0}^{n} \subset \{0,1\}^*$ with $\mathcal{P}$ such that expected word length

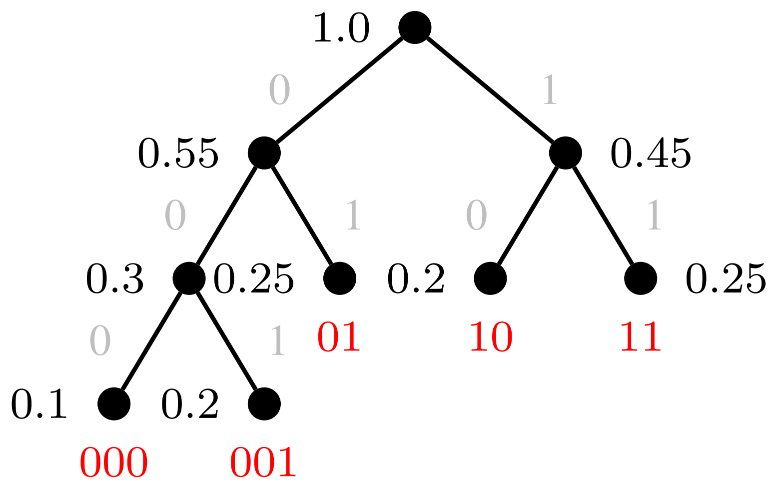
$$Cost(\mathcal{P}, w) = \sum_{i=0}^{n} p_i \cdot \text{length}(w_i)$$

is minimized, (length(w_i) is number of bits in w_i).

Introduction

Huffman coding

- Prefix code usually represented by tree.
- Code cost equals weighted external path length.
- Example: for $\mathcal{P} = \{0.1, 0.2, 0.2, 0.25, 0.25\}$,



- Expected codeword length

$$=0.1\times 3+0.2\times 3+0.2\times 2+0.25\times 2+0.25\times 2=2.3$$

Introduction

- Problem: construct optimum Huffman codes for the infinite geometric source for $0 < p < 1$:

$$\mathcal{P}_p = \left\{(1-p)p^i\right\}_{i=0}^{\infty}$$

Introduction

- Problem: construct optimum Huffman codes for the infinite geometric source for $0 < p < 1$:

$$\mathcal{P}_p = \left\{(1-p)p^i\right\}_{i=0}^{\infty}$$

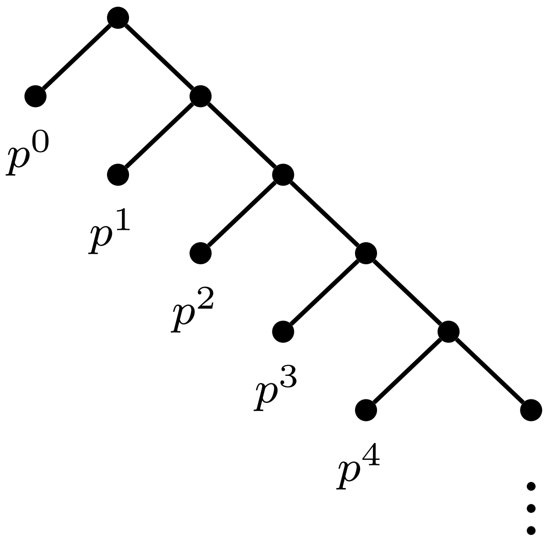
- Some trees and their average codeword lengths:

Introduction

- Problem: construct optimum Huffman codes for the infinite geometric source for $0 < p < 1$:

$$\mathcal{P}_p = \left\{(1-p)p^i\right\}_{i=0}^{\infty}$$

- Some trees and their average codeword lengths:



Average codeword length

$$= (1-p)\sum_{i=0}^{\infty} p^i(i+1)$$

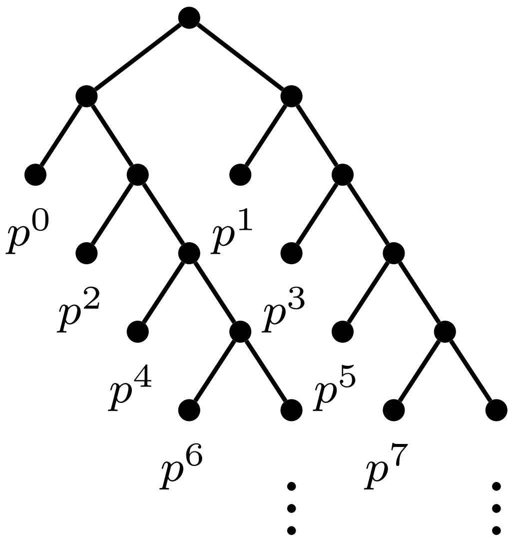
$$= \frac{1}{1-p}$$

Introduction

- Problem: construct optimum Huffman codes for the infinite geometric source for $0 < p < 1$:

$$\mathcal{P}_p = \left\{(1-p)p^i\right\}_{i=0}^{\infty}$$

- Some trees and their average codeword lengths:



Average codeword length

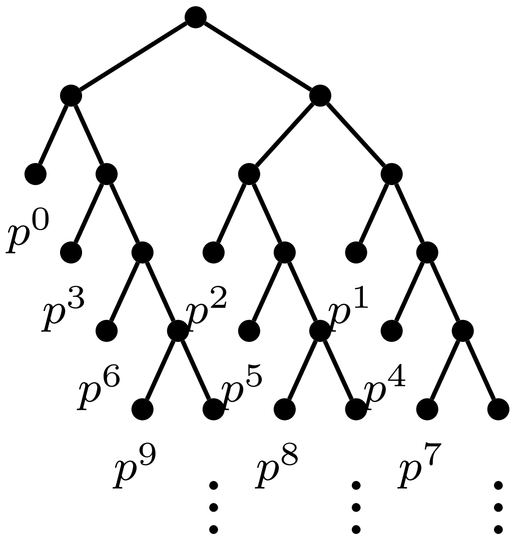
$$\begin{aligned} &= (1-p)\sum_{i=0}^{\infty}(p^{2i}+p^{2i+1})(i+2) \\ &= \frac{1}{1-p^2} - (1-p^2) \end{aligned}$$

Introduction

- Problem: construct optimum Huffman codes for the infinite geometric source for $0 < p < 1$:

$$\mathcal{P}_p = \left\{(1-p)p^i\right\}_{i=0}^{\infty}$$

- Some trees and their average codeword lengths:



Average codeword length

$$= (1-p)\left[2 + \sum_{i=0}^{\infty}(p^{3i+1} + p^{3i+2} + p^{3i+3})(i+3)\right]$$

$$= (1-p)\left[2 + (p + p^2 + p^3)\frac{3-2p^3}{(1-p^3)^2}\right]$$

Introduction

- Problem: construct optimum Huffman codes for the infinite geometric source for $0 < p < 1$:

$$\mathcal{P}_p = \{(1-p)p^i\}_{i=0}^{\infty}$$

- First arose in run-length encoding:

$$X_i = \underbrace{BB\ldots BB}_{i \text{ times}} A$$

(later used in dichotomous search and image compression.)

- Ordinary Huffman algorithm does not work
  (cannot find smallest pair of probabilities to merge)
- Not a-priori true that optimum code exists
  (infinite number of trees)

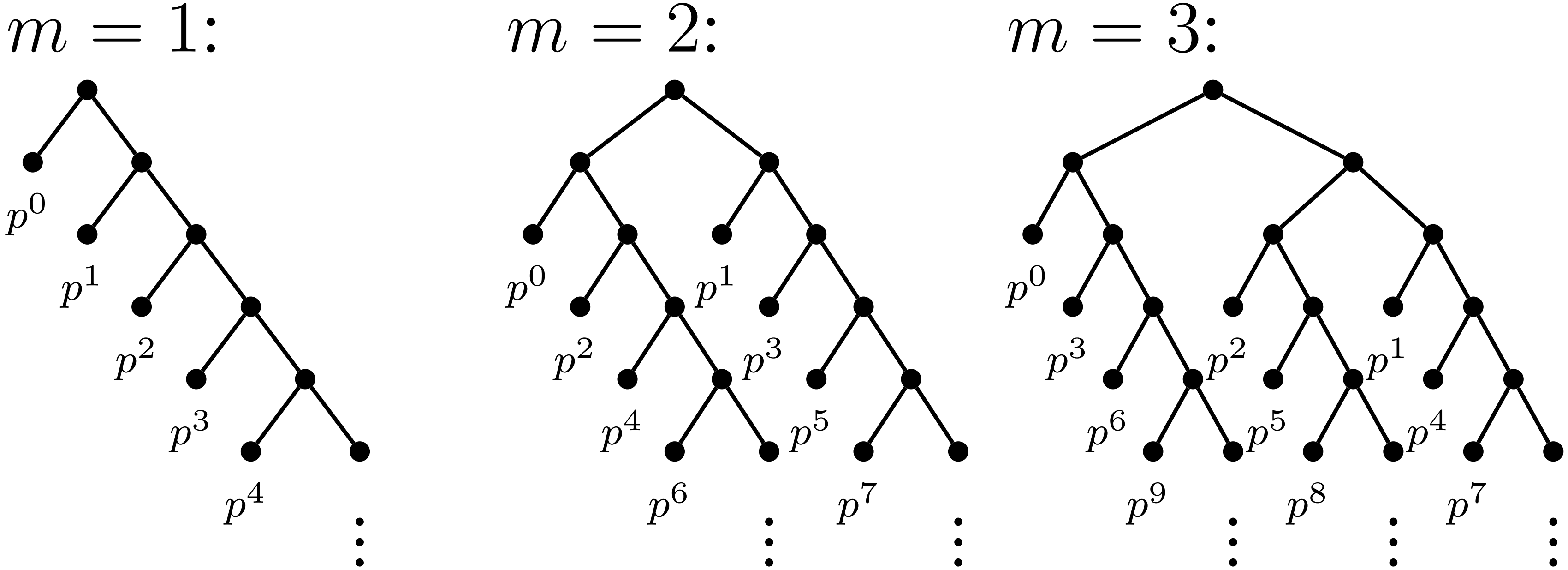
Previous Work

Problem: construct Huffman codes for infinite sources:

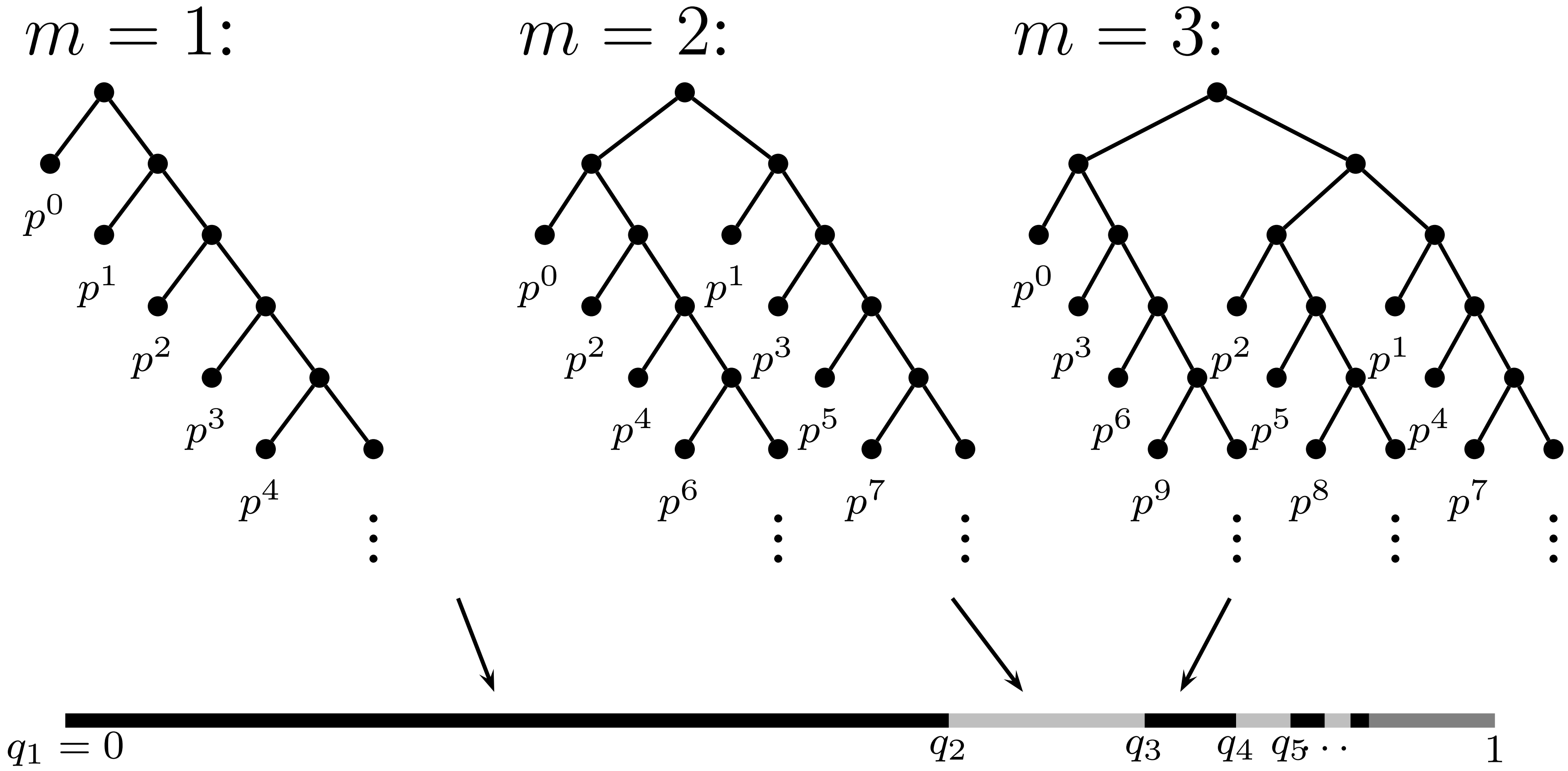
$$\mathcal{P}_p = \{(1-p)p^i\}_{i=0}^{\infty} \quad \text{and} \quad \mathcal{P} = p_0 \geq p_1 \geq p_2 \geq p_3 \geq \cdots$$

- **Golomb (1966):** Introduced problem and some solutions
- **Gallager, Van Voorhis (1975):**
  Gave binary solution for every $\mathcal{P}_p$
- **Abrahams (1994):** Generalized G & V.V. to ternary trees.
- **Linder, Tarokh, Zeger (1997):**
  Optimal tree exists iff Entropy $= -\sum_i p_i \log(p_i) < \infty$
- **Merhav, Seroussi, Weinberger (2000):**
  Optimal solutions for Two-Sided Geometric Distributions
- **Golin (2001):** G & V.V. solution unique

Previous Work: Gallager & Van Voorhis (1975)



Previous Work: Gallager & Van Voorhis (1975)



Partition theorem: Let $q_m \in (0,1)$ be unique real root of $1 - p^m - p^{m-1} = 0$.

Then $\{q_m\}_{m=1}^{\infty} \uparrow 1$ and all $p \in [q_{m-1}, q_m]$ share mth optimal tree for $\mathcal{P}_p$.

Gallager & Van Voorhis Schema

Previous results derived by essentially

- Guessing

  Huffman tree for finite *reduced source* $\mathcal{P}_p^n$ with n outcomes, $n = 1, 2, 3, \ldots$.

  The finite sources "converge" to the infinite source.

- Validating

  guessed tree for reduced source by demonstrating Huffman *sibling property* of guessed trees.

- Proving

  that finite trees "structurally" converge to given infinite tree is therefore optimal for infinite source.

Our results

- When possible to guess and verify optimal codes for finite reduced sources, G. & V.V. schema finds optimal codes for entire class of distributions, e.g., *all geometric, double-sided geometric.*
- But, G. & V.V. schema is not an algorithm
- It doesn't work when it is

  hard to guess code,

  e.g., more complicated variations of geo distributions or

  impossible to verify,

  e.g., most variations of Huffman encoding.
- This talk introduces algorithmic technique that works for most generalizations of the geometric distribution and constrained variations of Huffman encoding.

Our results

The algorithm will be able to construct optimal infinite codes for

- Generalizations of Geometric Distributions
  e.g., $\mathcal{P} = 1, 1, 1, p, p, p, p^2, p^2, p^2, p^3, p^3, p^3, \ldots$.
- Variants of Huffman Coding
  e.g., all words must end with a particular pattern such as 101 or 0001
  or coding letters have unequal-cost.

We will illustrate technique by developing algorithm which, given input p, will construct optimal code for the $(1, 2)$-lopsided tree for $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$.

(α, β)-Lopsided Trees

Label left edges with $\cdot$ and right edges with $-$.

α and β are the respective lengths of $\cdot$ and $-$.

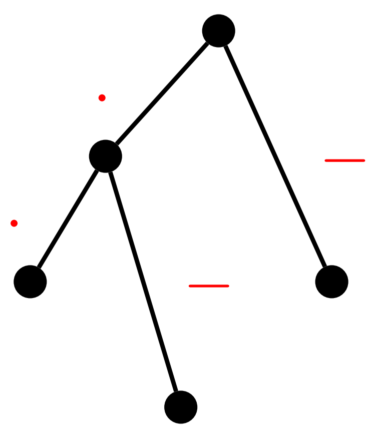
In a $(1, 2)$-Lopsided Trees $(\alpha, \beta) = (1, 2)$

(sometimes known as the *telegraph channel*).

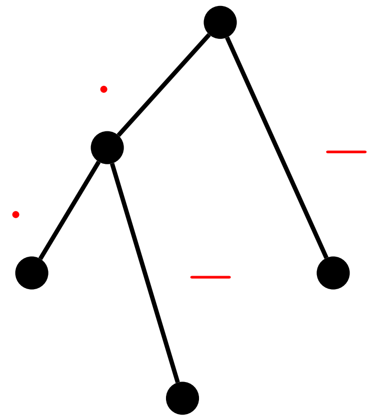
Cost of a leaf is the length of path from root to leaf.

Given $p_0 \geq p_1 \geq p_2 \geq \cdots \geq p_{n-1}$ cost of tree is $\sum_{i=0}^{n} p_i \cdot \text{depth}(v_i)$.

Example: tree below corresponds to $\{\cdot\cdot, -, \cdot-\}$ with cost $2p_0 + 2p_1 + 3p_2$.



(α, β)-Lopsided Trees



Currently unknown whether finding optimal codes for (α, β)-Lopsided Trees is NP-hard or not. But, for fixed integral α, β, there are polynomial-time dynamic programming algorithms for solving the problem.

We will now see an algorithm which, given p, constructs the optimal infinite $(1, 2)$-lopsided tree for $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$.

Algorithm Schema

- (I) In **finite** case (Golin, Rote 1998) showed how to find $(1, 2)$-lopsided trees by constructing a min-cost path in weighted directed graph.

Algorithm Schema

- (I) In **finite** case (Golin, Rote 1998) showed how to find $(1, 2)$-lopsided trees by constructing a min-cost path in weighted directed graph.

  For infinite distribution $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$ this would generalize to finding min-cost infinite length paths. How?

Algorithm Schema

- (I) In **finite** case (Golin, Rote 1998) showed how to find $(1, 2)$-lopsided trees by constructing a min-cost path in weighted directed graph.

  For infinite distribution $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$ this would generalize to finding min-cost infinite length paths. How?

- (II) Will show that optimum infinite prefix-code trees for geometric source must have a cyclic structure.

Algorithm Schema

- (I) In **finite** case (Golin, Rote 1998) showed how to find $(1, 2)$-lopsided trees by constructing a min-cost path in weighted directed graph.

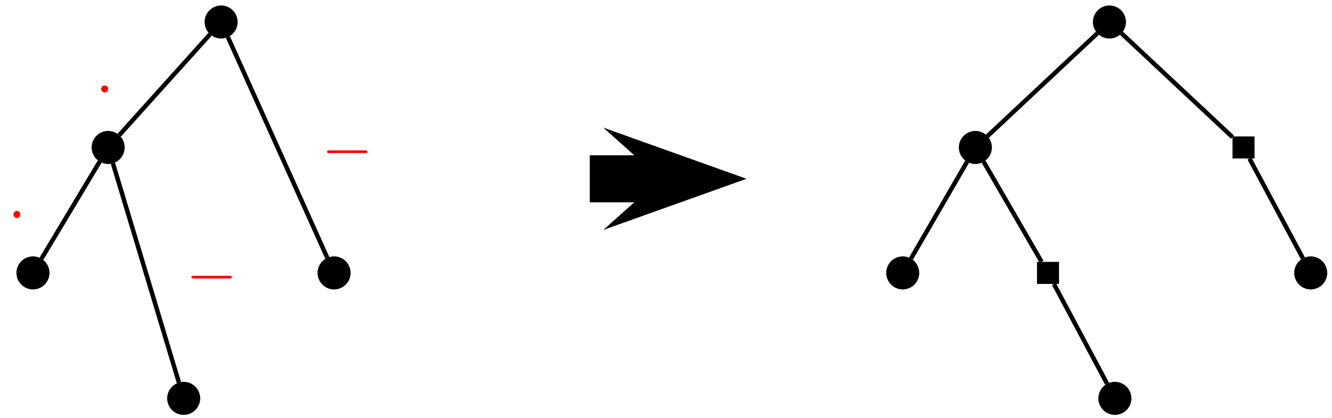
  For infinite distribution $\mathcal{P}_p = \{p^i\}_{i=0}^{\infty}$ this would generalize to finding min-cost infinite length paths. How?

- (II) Will show that optimum infinite prefix-code trees for geometric source must have a cyclic structure.

  Cyclic structure will correspond to "cyclic behavior" of path in graph. So, min cost infinite path can be found by just looking at finite prefixes of the paths. Max length of finite prefix will depend upon p.

Encoding the tree

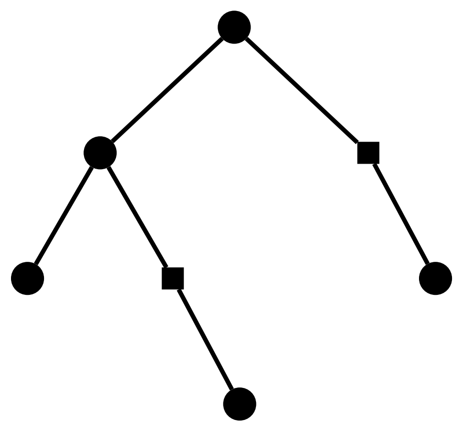
- Split long edges by *cross nodes* (■)



- Define
  - I_l: number of *internal nodes* on level l
  - E_l: the number of *external nodes* on level l
  - C_l: the number of *cross nodes* on level l
  - A_l: the number of *external nodes* on or above level l

Encoding the tree

- Example:



I_l	C_l	E_l	A_l
1	0	0	0
1	1	0	0
0	1	2	2
0	0	1	3

- $(I_l, C_l, E_l)_{l=0}^{\infty}$ completely defines infinite tree.
- $A_l = \sum_{i \le l} E_i$
- $\text{Cost} = \frac{1}{1-p} \sum_{0 \le l} p^{A_l}$

Remainder of Talk

- Properties of infinite trees for geometric sources
  1. Infinite trees have bounded width
  2. Infinite trees must cycle
  3. Infinite trees are “computable”
  4. Generalized partition Theorem
- Algorithm
  1. Derivation of an efficient algorithm
  2. Implementation results
- Open Questions

Bounded-Width Lemma

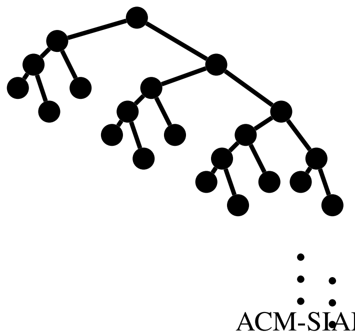
Set

$$B(p) = \min\{k : p^k < \frac{1-p}{2}\}.$$

If T is an optimal tree for $\mathcal{P}_p$, then, at every level l,

$$I_l, C_l, E_l \leq 3B(p)$$

Proof-Idea: If $I(l) > B(p)$, replacing internal node on level l by leaf produces cheaper tree. Bounding internal nodes also bounds leaves and cross nodes.



Cycle theorem

Theorem: Let T be an optimal tree for $\mathcal{P}_p$ and, for some i, t, a repeat occurs.

$$(I_i, C_i, E_i) = (I_{i+t}, C_{j+t}, E_{j+t}).$$

Set $f(l) = i + [(l - i) \bmod t]$.

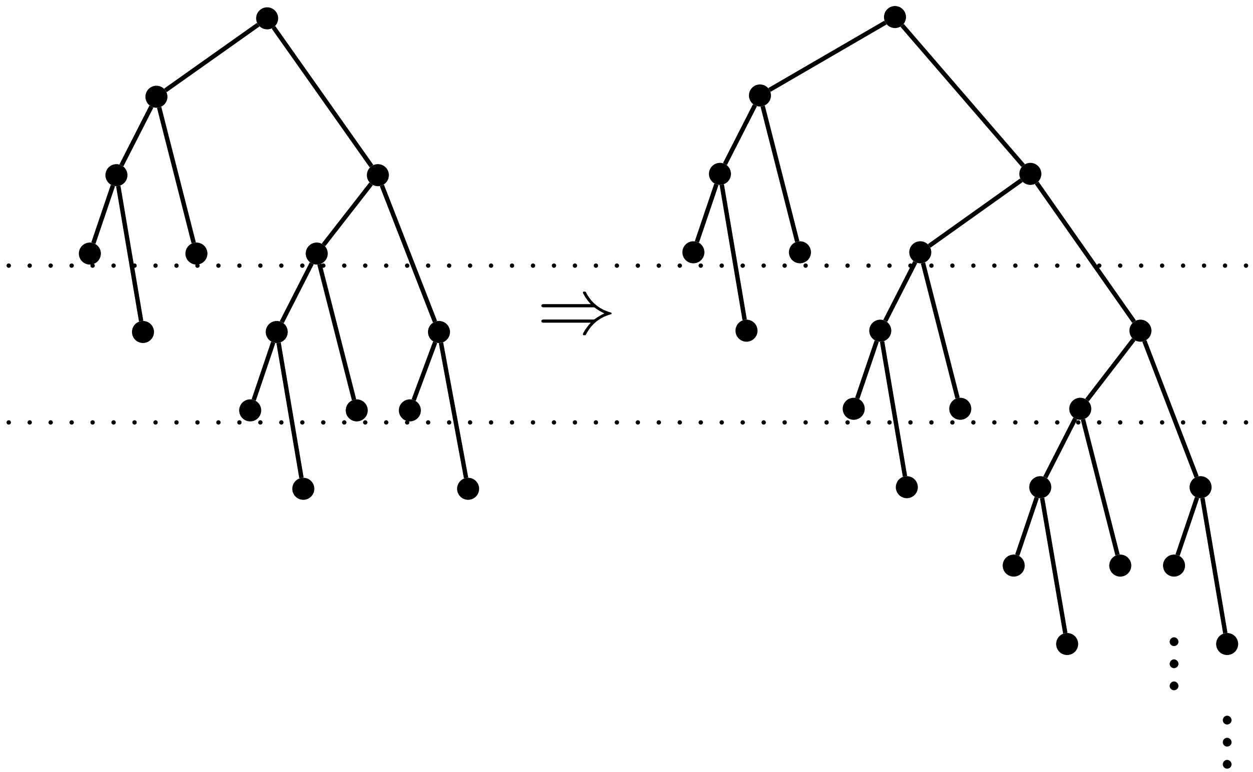
Then "cyclic" tree $(I'_l, C'_l, E'_l)_{l=0}^{\infty}$ is optimal for $\mathcal{P}_p$.

$$(I'_l, C'_l, E'_l) = \begin{cases} (I_l, C_l, E_l) & \text{if } i \leq l \\ \left(I_{f(l)}, C_{f(l)}, E_{f(l)}\right) & \text{if } i \geq l \end{cases}$$

Once a repetition occurs, it cycles forever.

Cycle theorem

If tree on the left is top of some optimal tree then *cyclic* tree on the right is an optimal tree



I	C	E
1	0	0
1	1	0
2	1	0
1	2	2
2	1	1
1	2	2
2	1	1
1	2	2
2	1	1

Optimal tree is Computable

- No. of (I_l, C_l, E_l) triples is $\Theta\left(B^3(p)\right)$.
- Once $(I_i, C_i, E_i) = (I_{i+t}, C_{j+t}, E_{j+t})$, is seen, cyclic tree is determined.
- No. of possible optimal trees is $O\left(\left(B^3(p)\right)!\right)$
- Cost of a cyclic tree can be written as $\frac{P(p)}{Q(p)}$.

  $P(.), Q(.)$ polynomials of degree bounded by function of $B(p)$.

Then optimal tree can be found by enumerating all possible trees and returning one with minimal cost.

Ugly but theoretically computable. Rest of talk is how to do this efficiently

Generalized Partition Theorem

Fix p': $\forall p \le p'$, $B(p) \le B(p')$ so

- No. of cyclic trees that can be optimal for *any* $p \le p'$ is a function of $B(p')$.
- Cost of a cyclic tree can be written as $\frac{P(p)}{Q(p)}$.

  $P(.)$, $Q(.)$ polynomials (degree bounded by function of $B(p')$).
- Complexity of lower envelope of these functions is therefore a function of p'

This generalizes old result of G & V.V to:

Partition theorem: $\exists \{q_m\}_{m=1}^{\infty} \uparrow 1$ such that all $p \in [q_{m-1}, q_m)$ share the same optimal tree for $\mathcal{P}_p$.

Algorithm

- Implicitly build an infinite weighted DAG with a unique source.
  Generalization of finite DAG from G. & Rote 1998
- $1 \leftrightarrow 1$ correspondence between infinite source rooted paths in graph and infinite codes for $\mathcal{P}_p$.
  Cost of path = cost of tree
- Need to find min cost infinite path in DAG
- Finite path in graph corresponds to top of tree
  Only need to look at finite paths until they cycle and then create corresponding cyclic tree

Algorithm: new encoding of the tree

$(m; e, c)$: *level* of a tree has

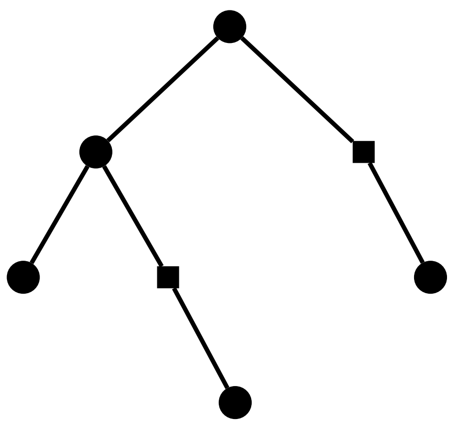
- m leaves *above it*
- c cross nodes *on it*
- e actual nodes *on it*

Algorithm: new encoding of the tree

$(m; e, c)$: *level* of a tree has

- m leaves *above it*
- c cross nodes *on it*
- e actual nodes *on it*

Example:



m_l	e_l	c_l
0	1	0
0	1	1
0	2	1
2	1	0

Algorithm: new encoding of the tree

$(m; e, c)$: *level* of a tree has

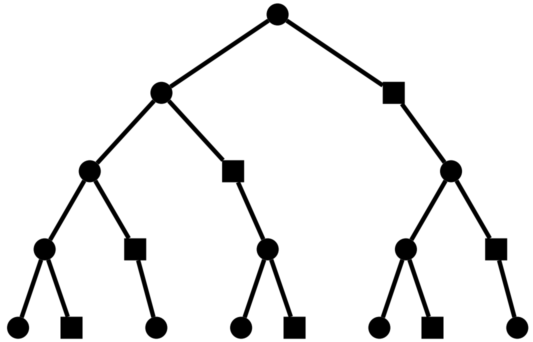
- m leaves *above it*
- c cross nodes *on it*
- e actual nodes *on it*

$(m_l, e_l, c_l)_{l=0}^{\infty}$ fully defines the tree.

When tree with bottom layer (m_b, e_b, c_b) expands (grows) one more layer, only possible change is that q of the nodes at level b can become internal and $e_b - q$ become leaves.

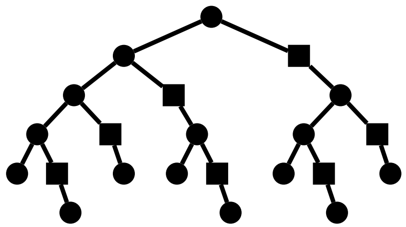
So $(m_{b+1}, e_{b+1}, c_{b+1})$ only depends upon q.

Expansion



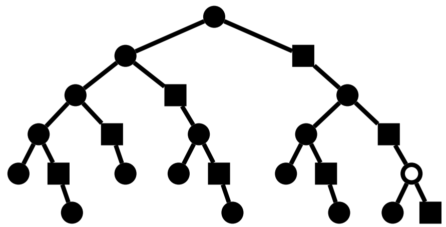
$(m; e, c) = (0; 5, 3)$

$q = 0:$



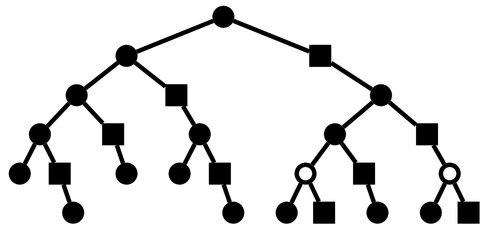
$(5; 3, 0)$

$q = 1:$



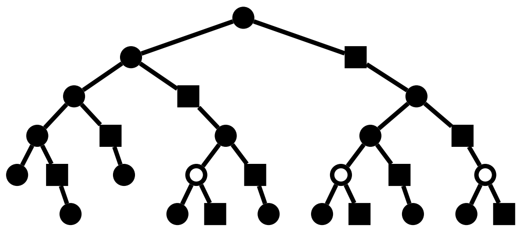
$(4; 4, 1)$

$q = 2:$



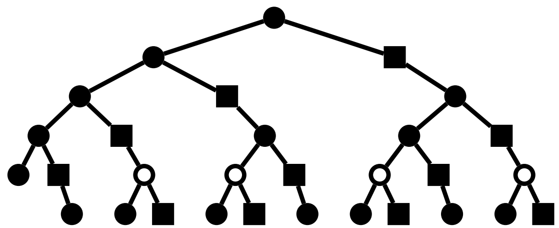
$(3; 5, 2)$

$q = 3:$



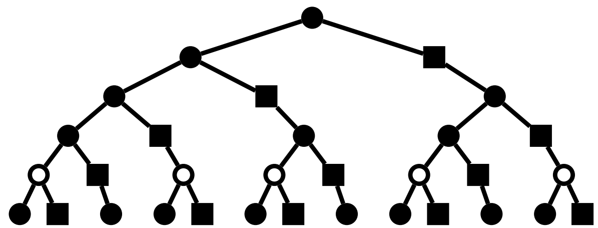
$(2; 6, 3)$

$q = 4:$



$(1; 7, 4)$

$q = 5:$



$(0; 8, 5)$

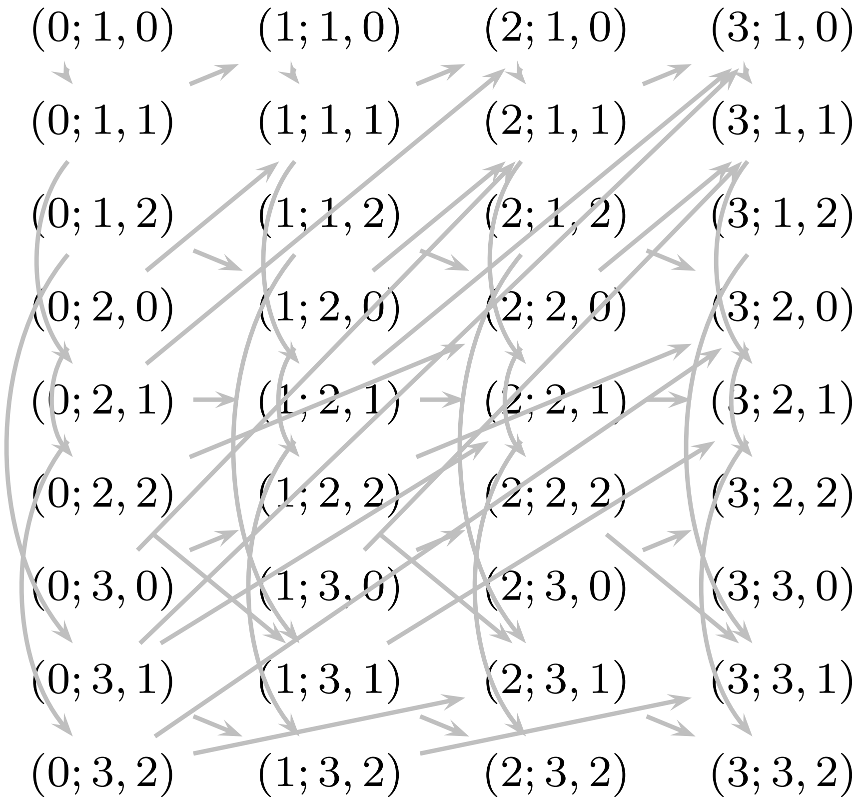
Algorithm

- Implicitly create weighted graph
  Min-cost tree is min-cost infinite path rooted at source in the weighted graph
- Weighted directed graph: $G = (V, E)$ where
  - $V = \{(m; e, c)\}$ where $e \leq 6B(p), c \leq 3B(p)$
  - $E = \{((m; e, c), (m + e - q; c + q, q)) : 0 \leq q \leq e\}$
    - $(m; e, c)$ has $e + 1$ outgoing edges (*expansions*)
    - Weight of edges leaving $(m; e, c) = \frac{p^{m+1}}{1-p}$
- If tree is optimal then, once it cycles, we know the entire infinite tree.
- Tree cycles if a (e, c) pair repeats.
  Must happen in at most $18B^2(p)$ steps.

Graph, Paths and Trees

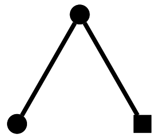
$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$

Graph, Paths and Trees



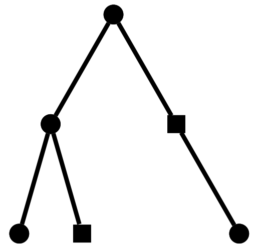
Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$



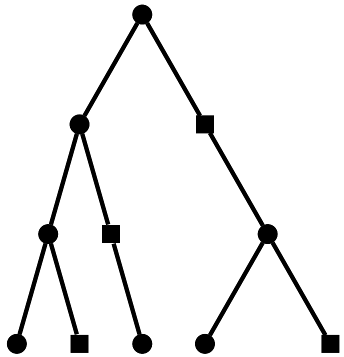
Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$



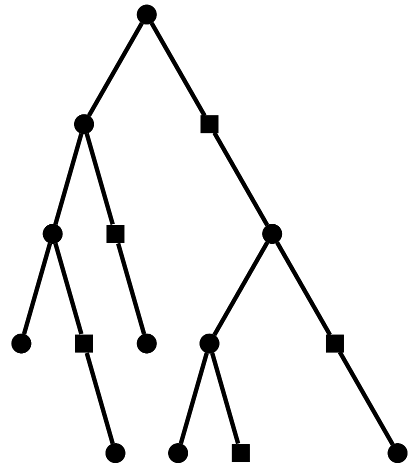
Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$



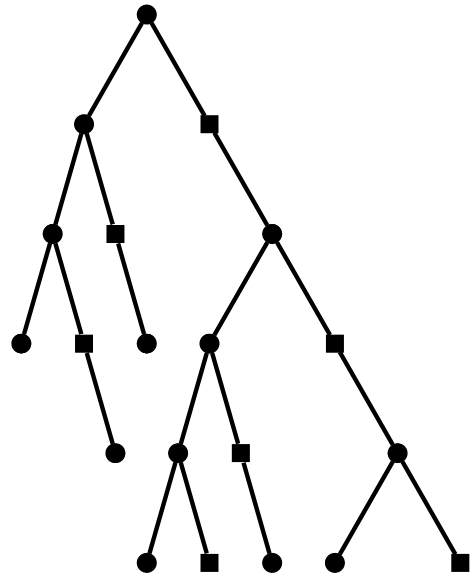
Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$



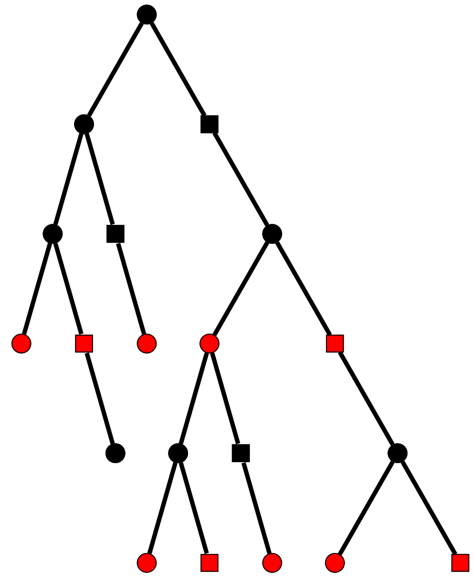
Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$



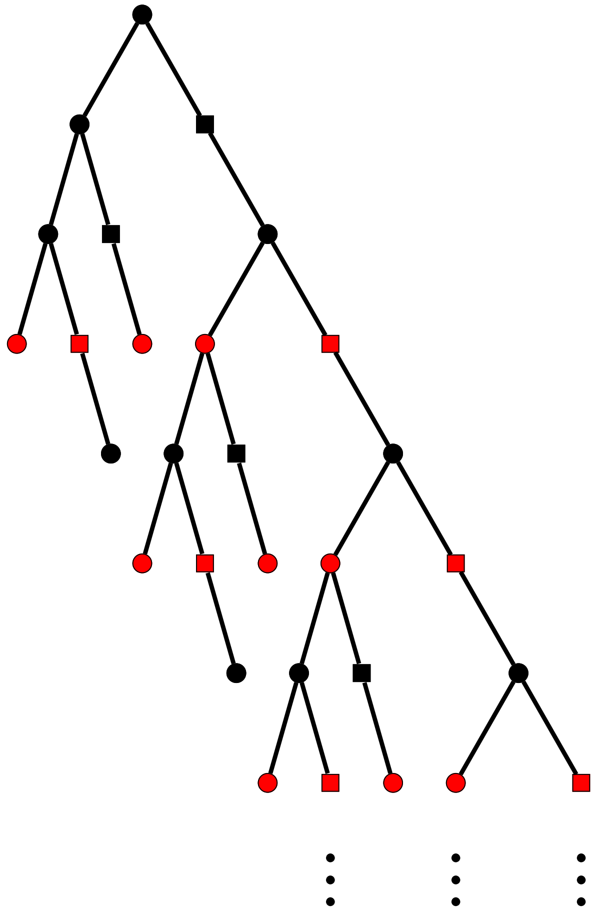
Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$



Graph, Paths and Trees

$(0;1,0)$	$(1;1,0)$	$(2;1,0)$	$(3;1,0)$
$(0;1,1)$	$(1;1,1)$	$(2;1,1)$	$(3;1,1)$
$(0;1,2)$	$(1;1,2)$	$(2;1,2)$	$(3;1,2)$
$(0;2,0)$	$(1;2,0)$	$(2;2,0)$	$(3;2,0)$
$(0;2,1)$	$(1;2,1)$	$(2;2,1)$	$(3;2,1)$
$(0;2,2)$	$(1;2,2)$	$(2;2,2)$	$(3;2,2)$
$(0;3,0)$	$(1;3,0)$	$(2;3,0)$	$(3;3,0)$
$(0;3,1)$	$(1;3,1)$	$(2;3,1)$	$(3;3,1)$
$(0;3,2)$	$(1;3,2)$	$(2;3,2)$	$(3;3,2)$
			$(?;3,1)$
			$(?;3,2)$
			$\vdots$



Algorithm

- Start running Dijkstra's min-cost path spanning tree algorithm from source $(0; 1, 0)$.
- When node is taken off fringe, check if path from root to node creates an (e, c) cycle.
  - if no, continue normally with algorithm
  - if yes: Create infinite cyclic tree T corresponding to path
    Compare $Cost(T)$ to previous minimum cost tree found and
    if is smaller save T as current candidate minimum cost tree

    Throw node away (do not update cost to neighbors).

Algorithm is finite since spanning tree has no path with repeated (e, c) pair; algorithm runs in $O\left(B^4(p)\log B(b)\right)$ time

Can prove that infinite cyclic tree found is min-cost tree for $\mathcal{P}_p$.

Results

Algorithm run on p from 0.001 to 0.960 with 0.001 increments.

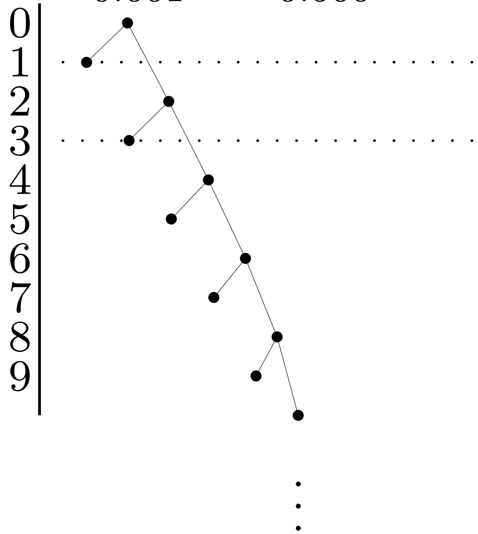
Signature expansions are for first 6 trees found.

All p in given intervals share same tree.
Underlined signatures are where level repetitions occur.

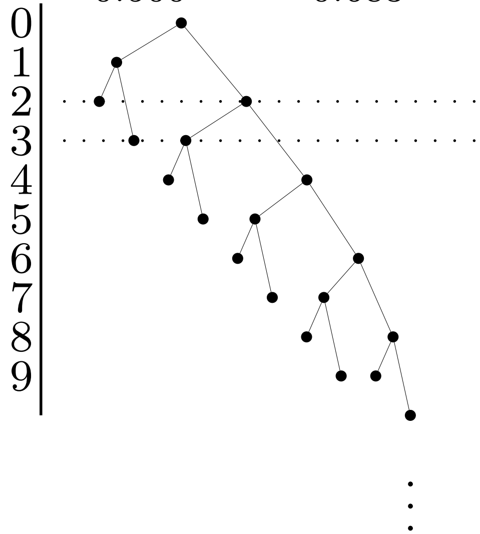
Interval	Expansion
$\mathcal{P}_{0.001} - \mathcal{P}_{0.500}$	$\underline{(0;1,1)} \rightarrow (1;1,0) \rightarrow \underline{(1;1,1)}$
$\mathcal{P}_{0.500} - \mathcal{P}_{\sim 0.683}$	$(0;1,1) \rightarrow \underline{(0;2,1)} \rightarrow \underline{(1;2,1)}$
$\mathcal{P}_{\sim 0.683} - \mathcal{P}_{\sim 0.755}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow \underline{(0;3,2)} \rightarrow (2;3,1) \rightarrow \underline{(3;3,2)}$
$\mathcal{P}_{\sim 0.755} - \mathcal{P}_{\sim 0.809}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow (0;3,2) \rightarrow \underline{(1;4,2)} \rightarrow \underline{(3;4,2)}$
$\mathcal{P}_{\sim 0.809} - \mathcal{P}_{\sim 0.838}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow (0;3,2) \rightarrow \underline{(0;5,3)} \rightarrow (3;5,2) \rightarrow \underline{(5;5,3)}$
$\mathcal{P}_{\sim 0.838} - \mathcal{P}_{\sim 0.864}$	$(0;1,1) \rightarrow (0;2,1) \rightarrow (0;3,2) \rightarrow (0;5,3) \rightarrow \underline{(2;6,3)} \rightarrow \underline{(5;6,3)}$

Result

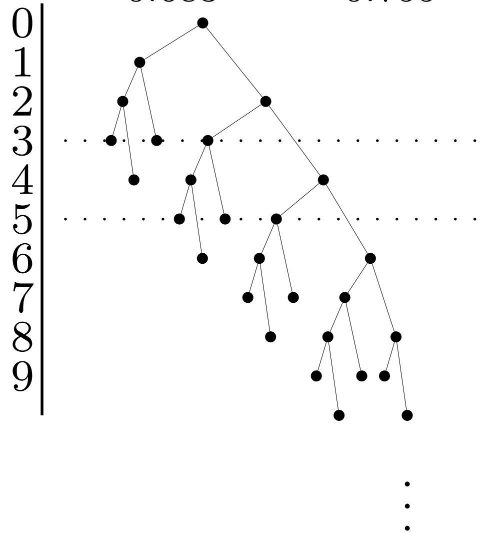
$\mathcal{P}_{0.001} - \mathcal{P}_{0.500}$:



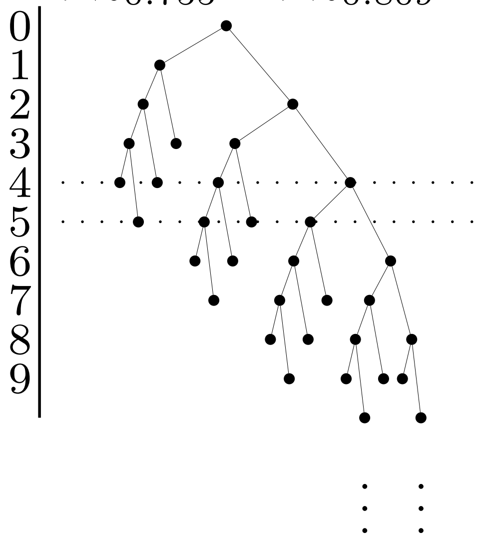
$\mathcal{P}_{0.500} - \mathcal{P}_{\sim 0.683}$:



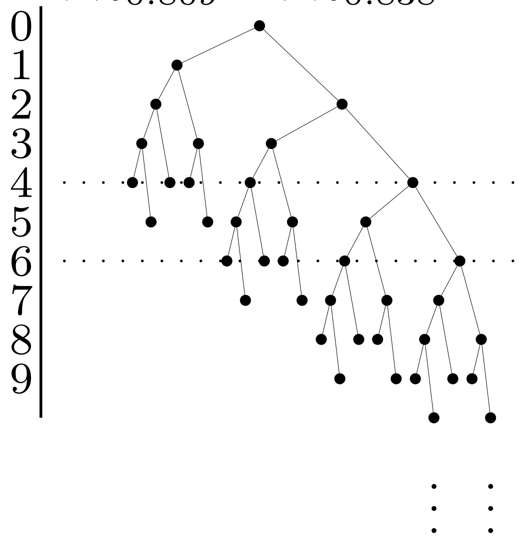
$\mathcal{P}_{\sim 0.683} - \mathcal{P}_{\sim 0.755}$:



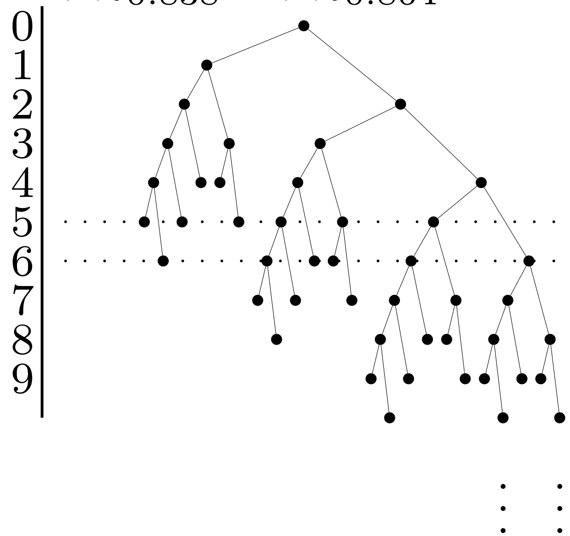
$\mathcal{P}_{\sim 0.755} - \mathcal{P}_{\sim 0.809}$:



$\mathcal{P}_{\sim 0.809} - \mathcal{P}_{\sim 0.838}$:



$\mathcal{P}_{\sim 0.838} - \mathcal{P}_{\sim 0.864}$:



Conclusions

In this talk we developed an algorithm that, given p, found optimum infinite cost $(1,2)$-lopsided tree for $\mathcal{P}_p = \left\{(1-p)p^i\right\}$.

This algorithm can easily be modified to work for:

- general (α, β) lopsided trees,
- cases in which prefix-free codewords are restricted to contain specified patterns
- cases in which the distribution is a generalization of the geometric distribution,
- and combinations of the above

Open Questions

- Suppose that $0 < p < p' < 1$ and p and $\mathcal{P}_p$, $\mathcal{P}_{p'}$ share the same optimal tree T.
  Is it true that $\forall p'' \in (p, p')$, T is also optimal for p'?
- Is it possible to go a *non-algorithmic* route and generalize the Gallager, Van Voorhis schema to solve our problems?
- All optimal infinite trees found to date have been for distributions that are (almost) geometric. Their derivations implicitly use the boundedness lemma.
  Is it possible to find an optimal code for a distribution that does not satisfy the boundedness lemma?

  Example: The Zeta distributions $\mathcal{P}_s = \{n^{-s}\}_{n=1}^{\infty}$.
  Optimal trees for these distributions have exponentially increasing width.

  No optimal tree for *any* $\mathcal{P}_s$ has ever been found.