TSKS01 Digital Communication Lecture 7

ML sequence detection, Viterbi algorithm

Emil Björnson

Department of Electrical Engineering (ISY)

Division of Communication Systems

COMMUNICATION SYSTEMS

LINKÖPING UNIVERSITY

Recall: L-tap Channel with Gaussian Noise

- Sampled output

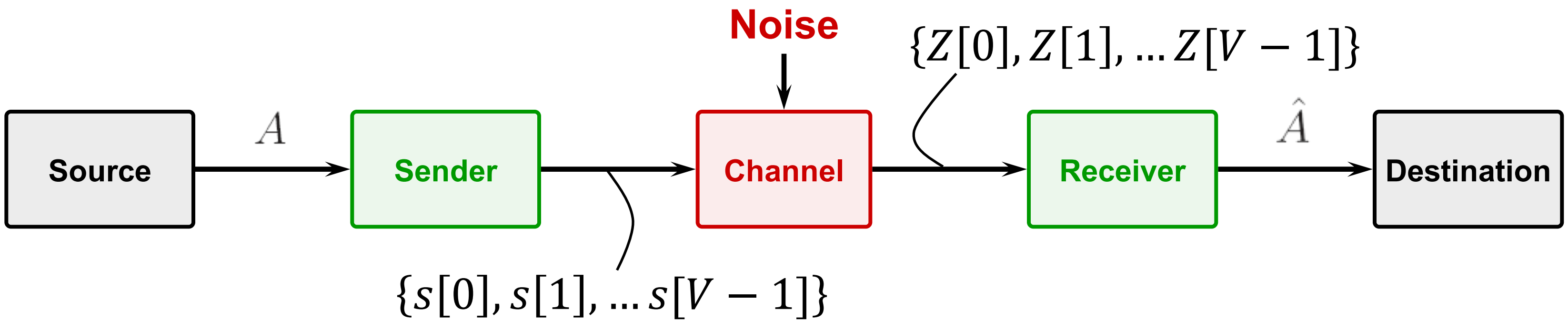
$$Z[k] = \underbrace{\sum_{l=0}^{L-1} s[k-l]g[l]}_{b(s[k],\ldots,s[k-L+1])} + W_\gamma[k]$$

where $W_\gamma[k]$ is

- Gaussian with zero mean and variance $N_0/2$
- Independent for different values of k

Dispersive channel
Memory: $L-1$ previous symbols

Detecting Sequence of V symbols



Message set: $\{a_i\}_{i=0}^{M^V-1}$ Size: M^V

A message consists of V symbols from an M-size constellation ($N = 1$)

Mapping: $a_i \rightarrow \{s_i[0], s_i[1], \dots s_i[V-1]\}$

for $i \in \{0,1,\dots,M^V-1\}$

Assume: $s[-L+1], \dots, s[-1]$ are known (initial memory)

ML Sequence Detection (1/3)

- Recall:

ML decision rule: Set $\hat{a} = a_i$ if $f_{\bar{Z}|A}(\bar{z}|a_n)$ is maximized for $n = i$.

- In this case:

$$\overline{Z} = \begin{pmatrix} Z[0] \\ Z[1] \\ \vdots \\ Z[V-1] \end{pmatrix}$$

- Note

$$Z[k] = \sum_{l=0}^{L-1} s[k-l]g[l] + W_\gamma[k]$$

is Gaussian with variance $N_0/2$ and mean $\sum_{l=0}^{L-1} s[k-l]g[l]$

ML Sequence Detection (2/3)

- Define:

$$b(s_n[k], \dots, s_n[k-L+1]) = \sum_{l=0}^{L-1} s_n[k-l]g[l]$$

  - We obtain:

$$f_{\overline{Z}|A}(\overline{z}|a_n) = \prod_{k=0}^{V-1} f_{Z[k]|A}(z[k]|a_n)$$

$$= \frac{1}{(\pi N_0)^{V/2}} e^{-\frac{1}{N_0}\sum_{k=0}^{V-1}\left(z[k]-b(s_n[k],\dots,s_n[k-L+1])\right)^2}$$

  - Maximized by minimizing

$$\sum_{k=0}^{V-1}\left(z[k]-b(s_n[k],\dots,s_n[k-L+1])\right)^2$$

ML Sequence Detection (3/3)

- Notation:

$$\overline{Z} = \begin{pmatrix} Z[0] \\ Z[1] \\ \vdots \\ Z[V-1] \end{pmatrix} \qquad \bar{b}_n = \begin{pmatrix} b(s_n[0], \ldots, s_n[-L+1]) \\ b(s_n[1], \ldots, s_n[-L+2]) \\ \vdots \\ b(s_n[V-1], \ldots, s_n[V-L]) \end{pmatrix}$$

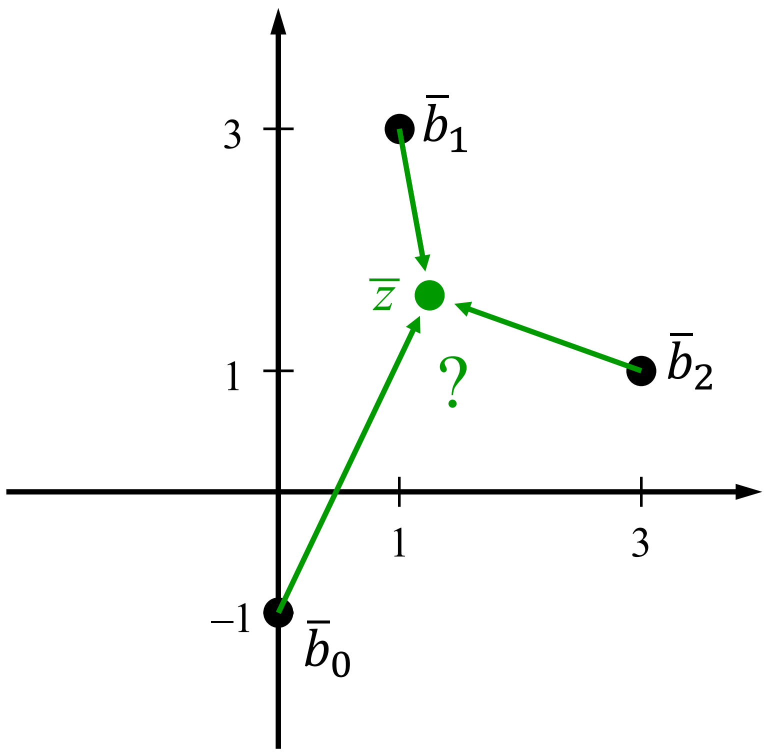
  - We have

$$\sum_{k=0}^{V-1} \big(z[k] - b(s_n[k], \ldots, s_n[k-L+1])\big)^2 = d^2(\bar{z}, \bar{b}_n)$$

ML decision rule: Set $\hat{a} = a_i$ if $d\big(\bar{z}, \bar{b}_n\big)$ is minimized for $n = i$.

  - Euclidean distance between $\bar{z}$ and received signal without noise

Geometric Illustration



Received vector: $\bar{z}$

Goal: Minimize the error probability

$$\Pr\{A \neq \hat{A} | \bar{Z} = \bar{z}\}$$

Sent message

Estimated message

Received vector

Observed realization

Same setup as before

Are we done now?

Complexity Considerations

- ML sequence detection
  - Compute $d(\bar{z}, \bar{b}_n)$ for $n = 0, \dots, M^V - 1$
  - Select the smallest distance

Number of messages grows very fast!

- Ways to reduce complexity
  1. Send one symbol and be silent for $L - 1$ symbols
  2. Design some approximate detection algorithm
  3. Find a way to implement ML detection more efficiently

This is what we will do!

Vector notation

- Observed signal

$$\bar{Z} = \bar{b}_n + \bar{W}_\gamma$$

where

$$\bar{Z} = \begin{pmatrix} Z[0] \\ \vdots \\ Z[V-1] \end{pmatrix} \quad \bar{W}_\gamma = \begin{pmatrix} W_\gamma[0] \\ \vdots \\ W_\gamma[V-1] \end{pmatrix}$$

$$\bar{b}_n = \begin{pmatrix} b(s_n[0], \dots, s_n[-L+1]) \\ \vdots \\ b(s_n[V-1], \dots, s_n[V-L]) \end{pmatrix}$$

ML Sequence Detection

$$\sum_{k=0}^{V-1}\left(z[k]-b(s_n[k],\ldots,s_n[k-L+1])\right)^2 = d^2(\bar{z},\bar{b}_n)$$

ML decision rule: Set $\hat{a}=a_i$ if $d(\bar{z},\bar{b}_n)$ is minimized for $n=i$.

- Euclidean distance between $\bar{z}$ and received signal without noise
  - Same principle as in ML detection for AWGN channels
  - Complexity is different: M messages → M^V messages

How to reduce complexity?

Alternative ML Detection Formulation (1/2)

- Define "memory state":
  - $\mathcal{S}[k] = \{s[k-1], \ldots, s[k-L+1]\}$

- Define metric at time k:

$$\Lambda_k(s[k], \mathcal{S}[k]) = \left(z[k] - b(s_n[k], \ldots, s_n[k-L+1])\right)^2$$

$$= \left(z[k] - \sum_{l=0}^{L-1} s[k-l]g[l]\right)^2$$

  - Depends only on "new" symbol $s[k]$ and state $\mathcal{S}[k]$
  - Is always positive (more precisely: non-negative)

Alternative ML Detection Formulation (2/2)

- We can rewrite

$$\sum_{k=0}^{V-1}\left(z[k]-b(s_n[k],\ldots,s_n[k-L+1])\right)^2 = d^2\left(\bar{z},\bar{b}_n\right)$$

as

$$\sum_{k=0}^{V-1}\Lambda_k(s[k],\mathcal{S}[k]) = d^2\left(\bar{z},\bar{b}_n\right)$$

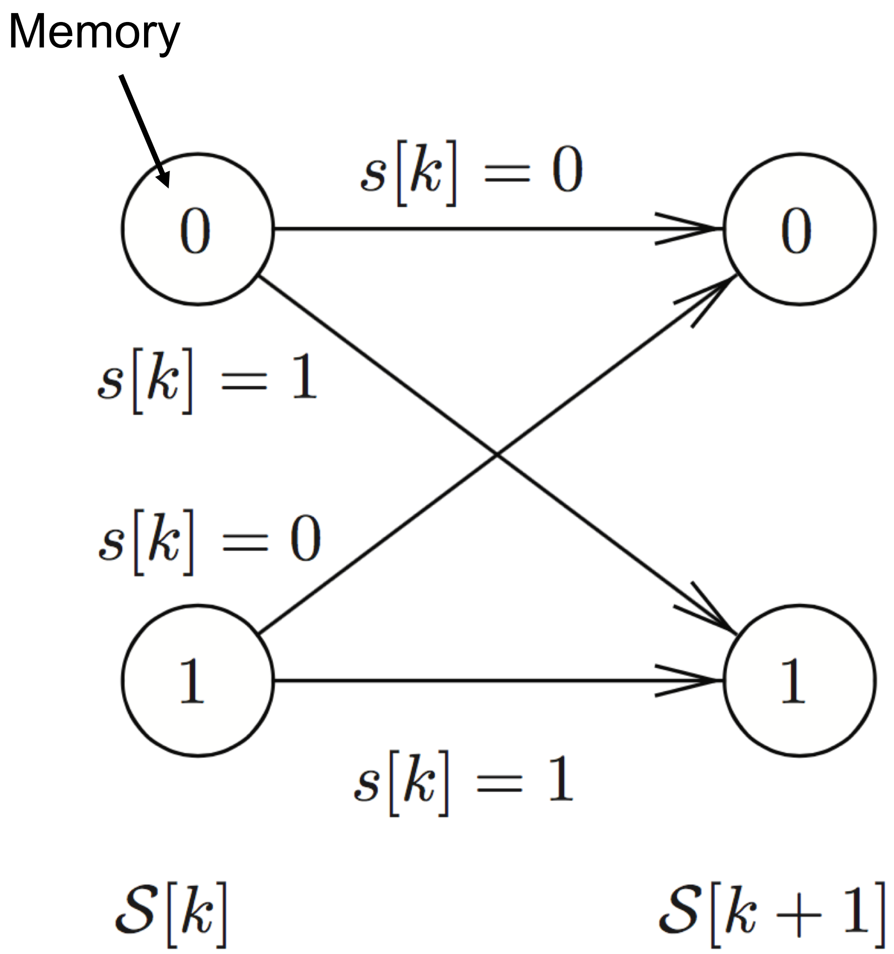
ML decision rule: Set $\hat{a} = a_i$ if $\sum_{k=0}^{V-1}\Lambda_k(s[k],\mathcal{S}[k])$ is minimized for $n = i$.

Example: State Transition ($L = 2$)

On-off keying

$s[k] \in \{0,1\}$

$L = 2$



Trellis

State transition resembles a trellis

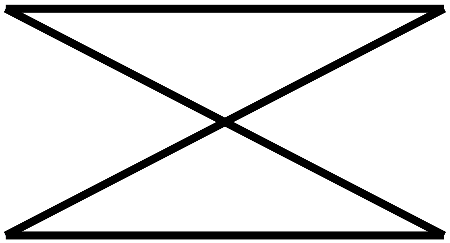
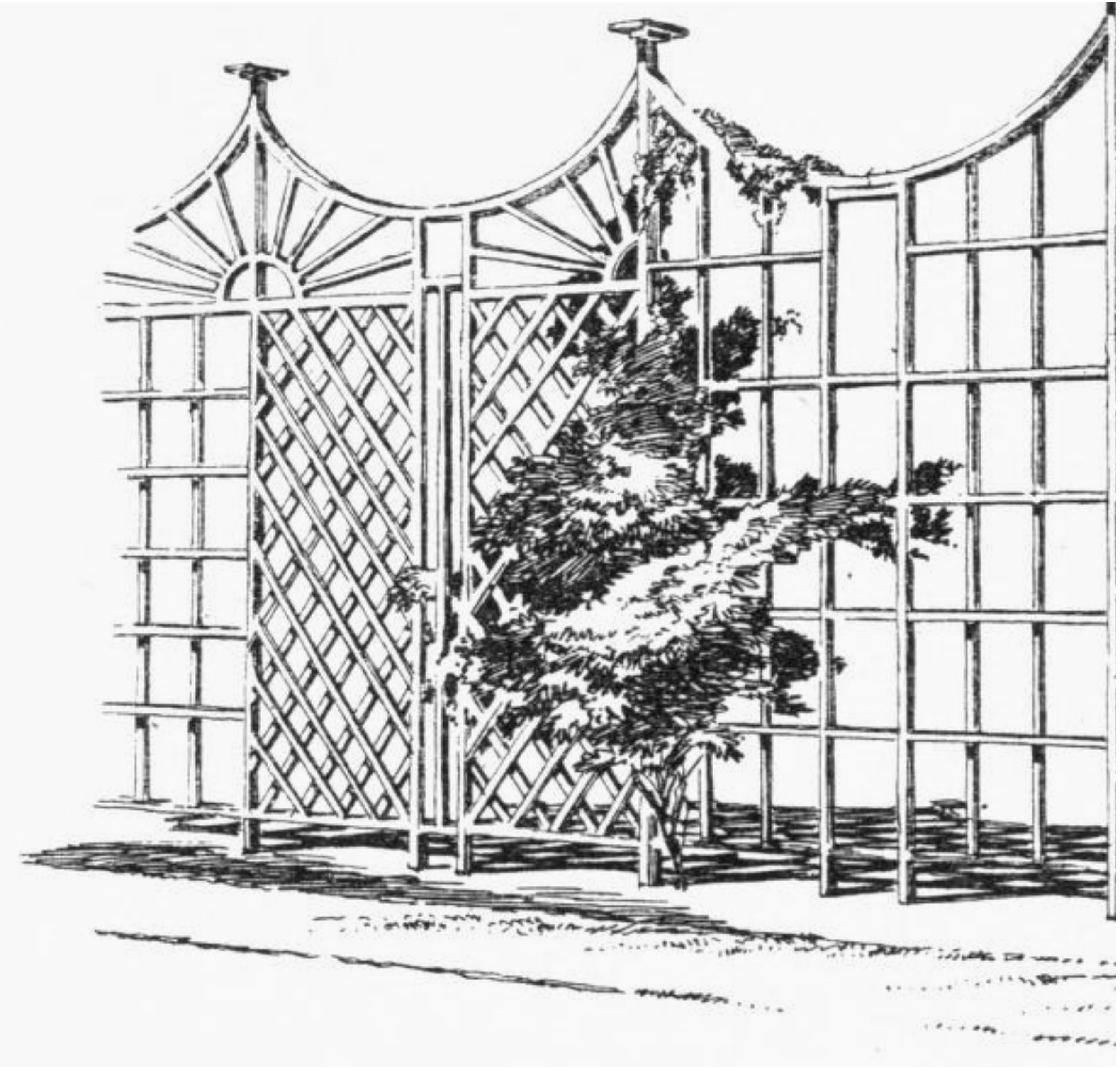
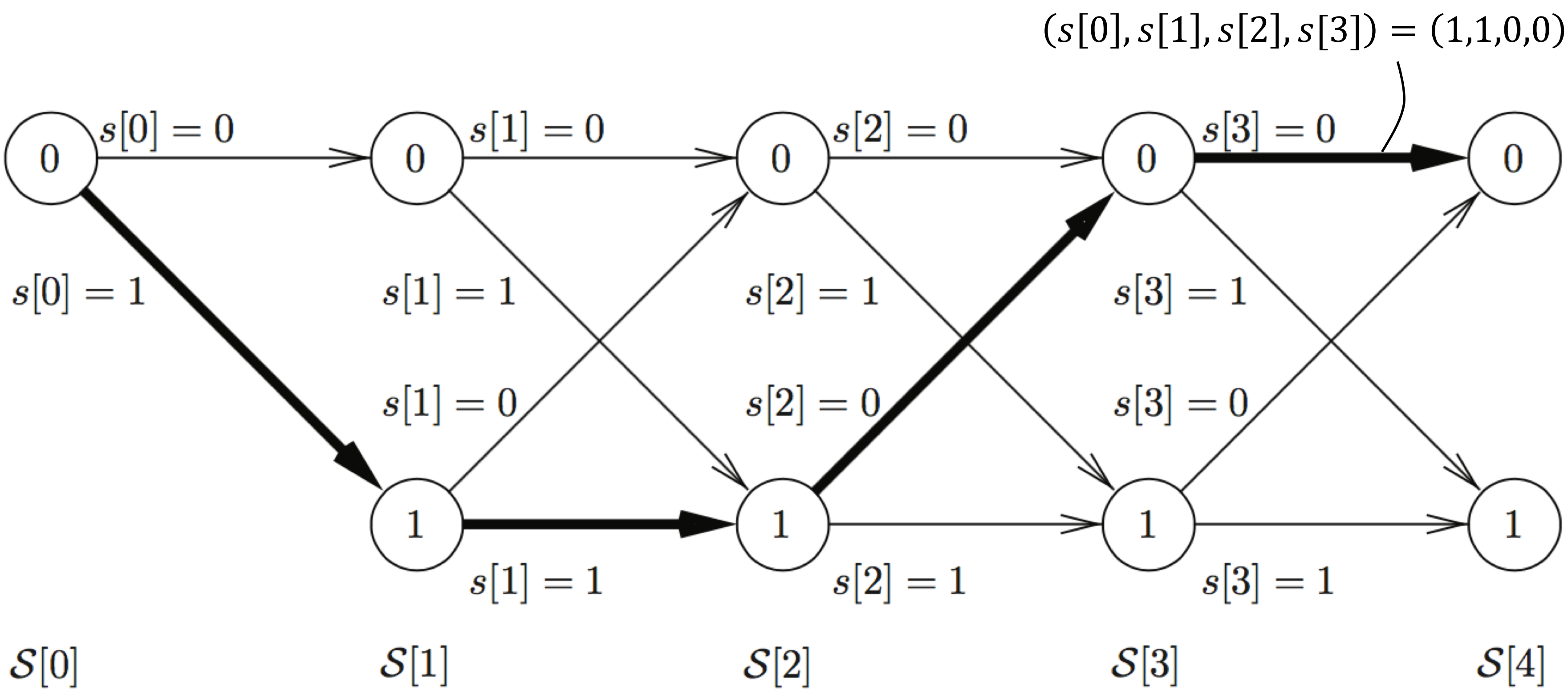


Image source: Wikipedia



Trellis Representation of State Transitions



On-off keying

$s[k] \in \{0,1\}$

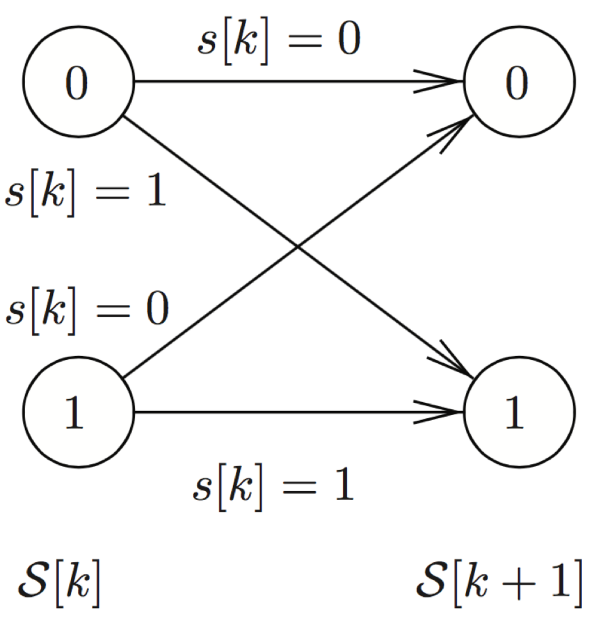
$L = 2, \mathrm{V} = 4$

Each message =
One path through the trellis

How Many Paths?

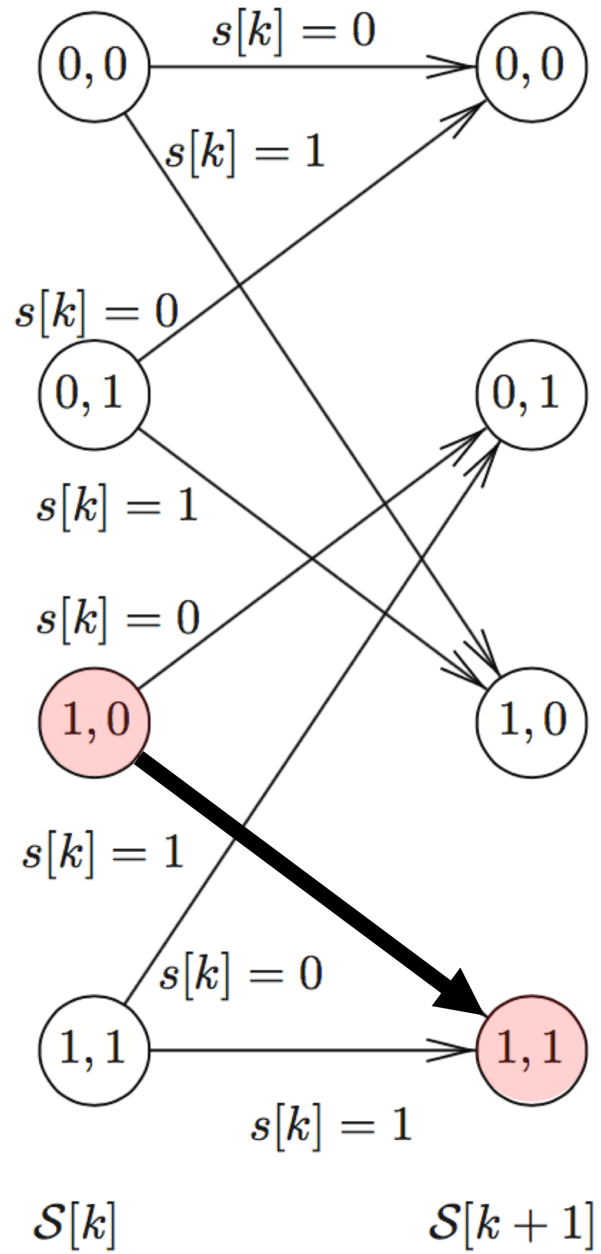
- Number of states
  - At time k: M^{L-1}
  - In total: $\approx VM^{L-1}$
- Number of paths
  - Number of messages: M^V
  - Paths leaving $\mathcal{S}[k]$: M
  - Paths reaching $\mathcal{S}[k]$: M
- Naming
  - $\mathcal{S}[0]$: Initial state
  - $\mathcal{S}[V]$: Ending state

On-off keying
$M = 2, L = 2$



Goal: Disregard all but one path reaching each state

Example: State Transition $(L = 3)$

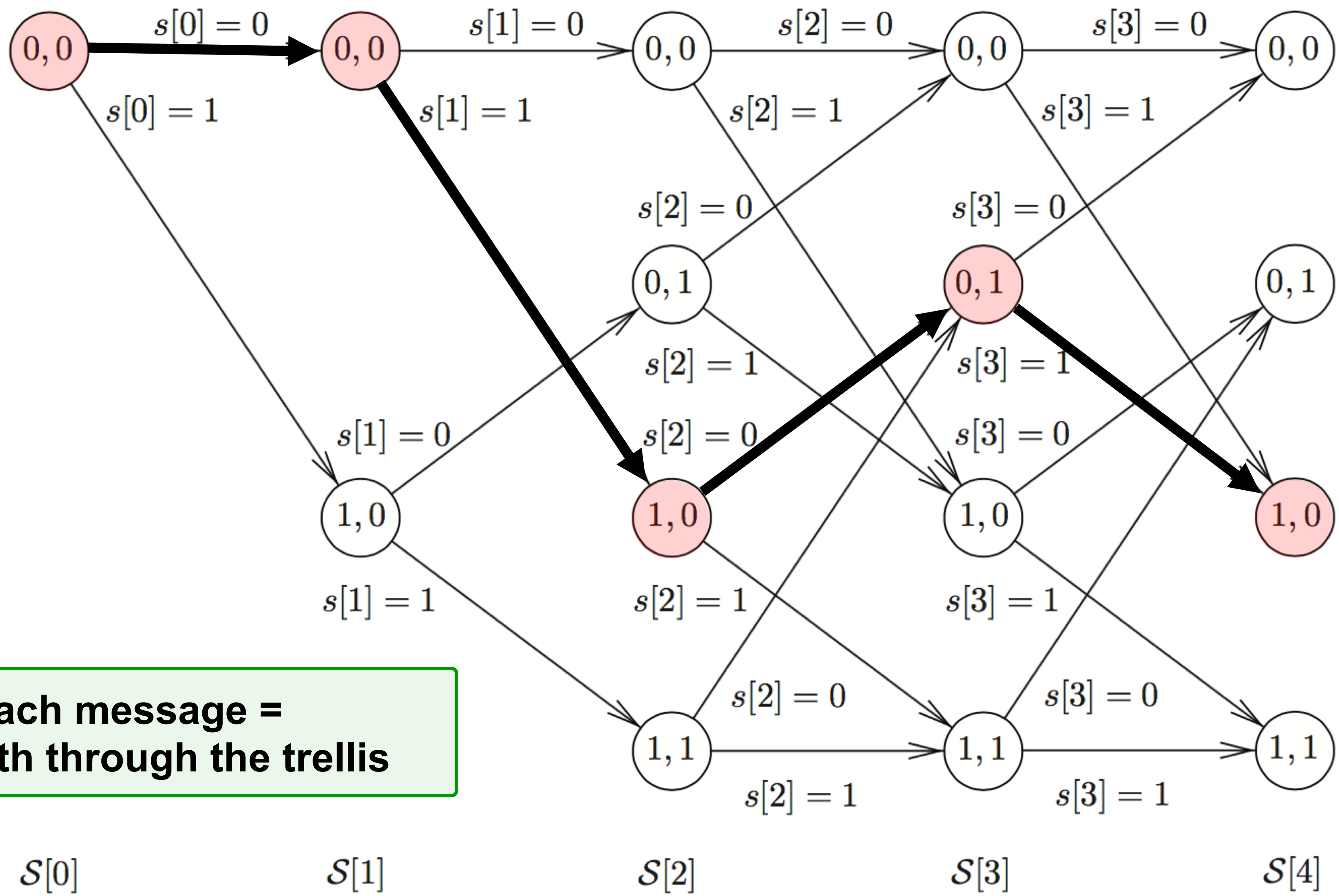


On-off keying

$s[k] \in \{0,1\}$

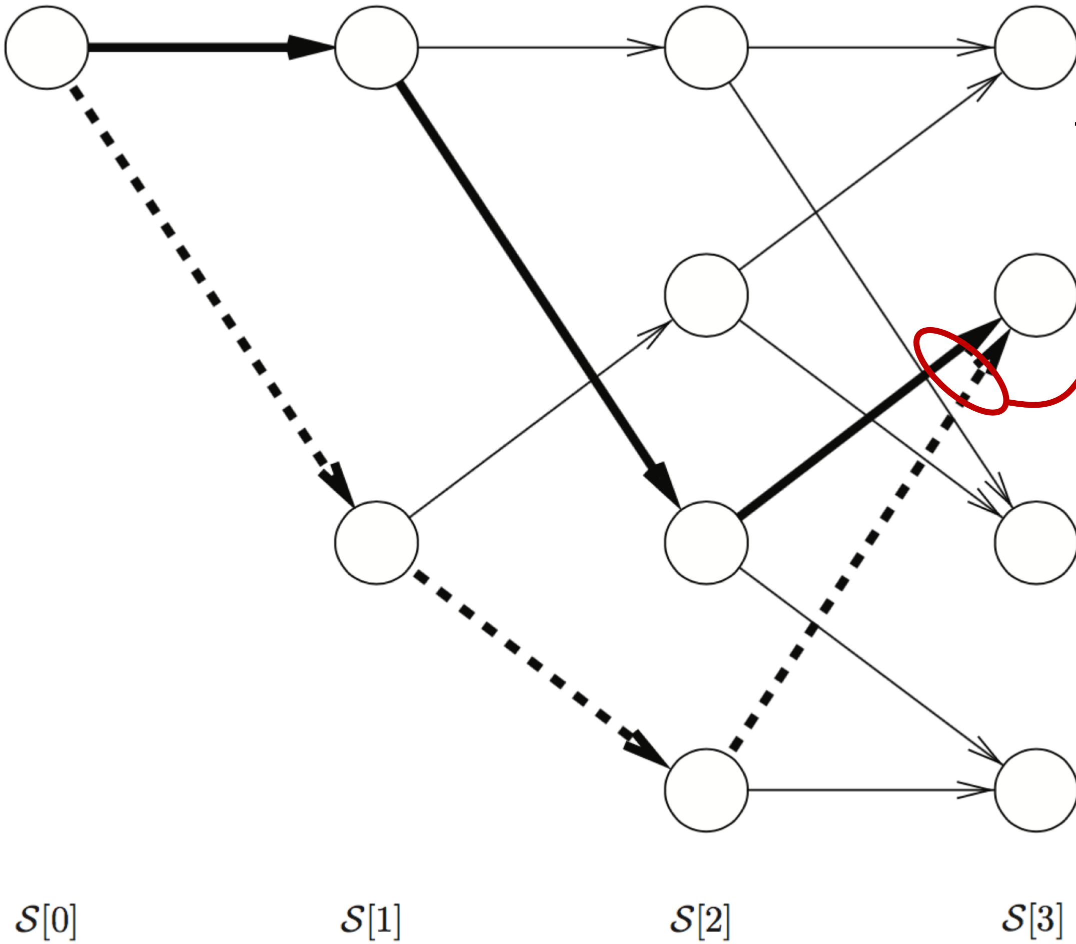
$L = 3$

Trellis Representation of State Transitions



Each message =
One path through the trellis

Comparing Accumulated Metrics



Two different values of

$$\sum_{k=0}^{2} \Lambda_k(s[k], \mathcal{S}[k])$$

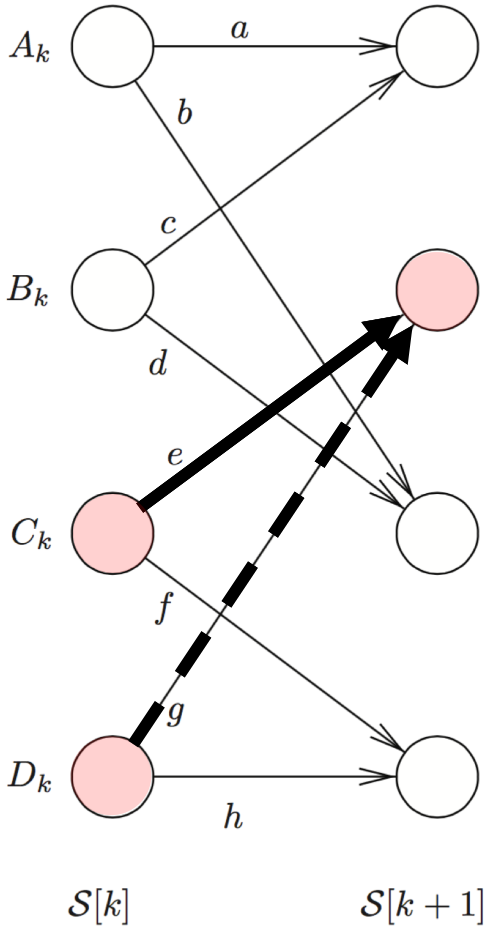
If $(s[3], \ldots s[V-1])$ is the same, remaining metric

$$\sum_{k=3}^{V-1} \Lambda_k(s[k], \mathcal{S}[k])$$

is the same for both paths

Keep only path with smallest accumulated metric

State Transition Computations



$$A_{k+1} = \min\left(A_k + a, B_k + c\right)$$

$$B_{k+1} = \min\left(C_k + e, D_k + g\right)$$

$$C_{k+1} = \min\left(A_k + b, B_k + d\right)$$

$$D_{k+1} = \min\left(C_k + f, D_k + h\right)$$

Each state:
Remember the path that reaches it with smallest metric
Called *survivor*

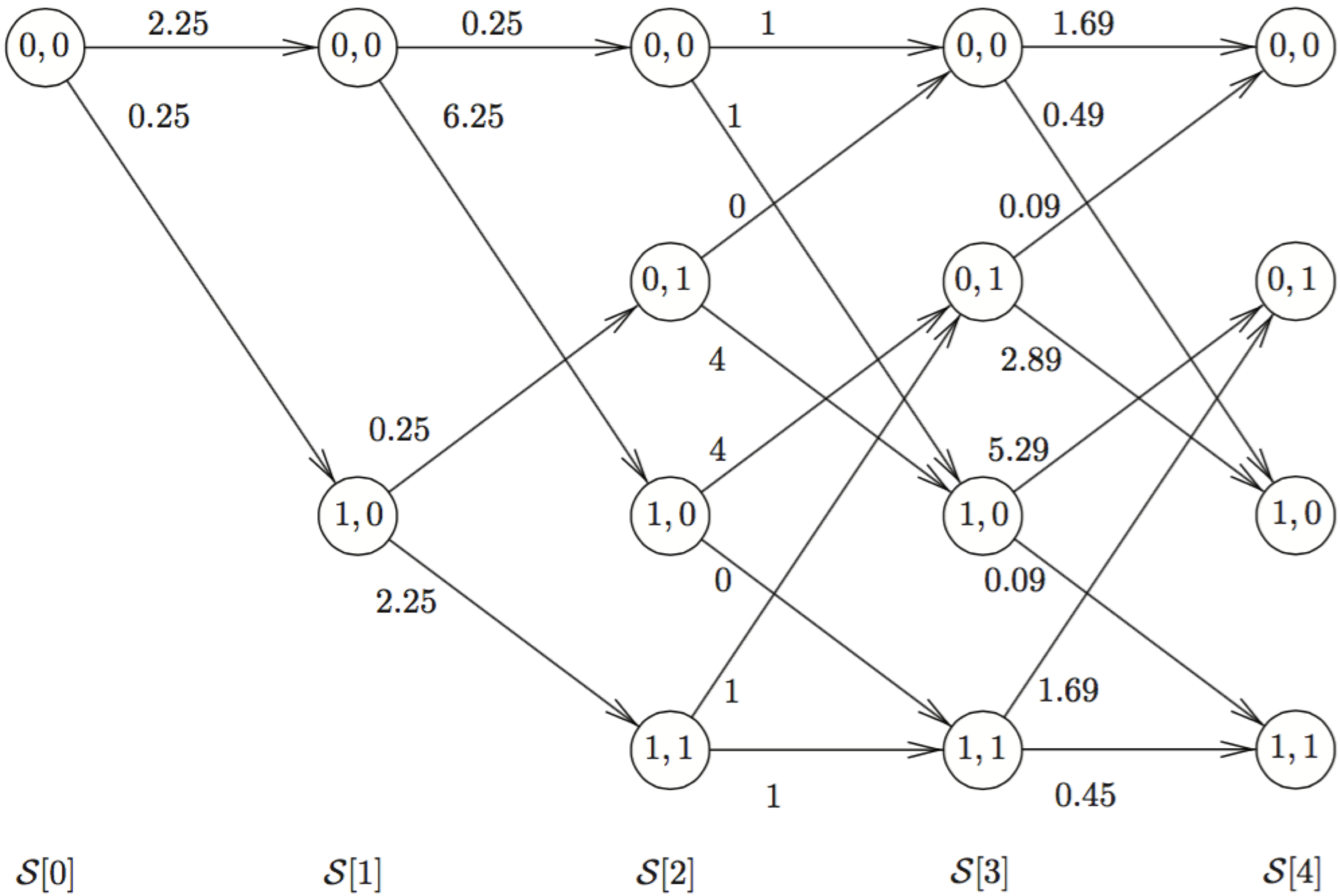
Ending states:
There are M^{L-1} *complete* paths left

Viterbi Algorithm

Perform ML sequence detection in three steps:

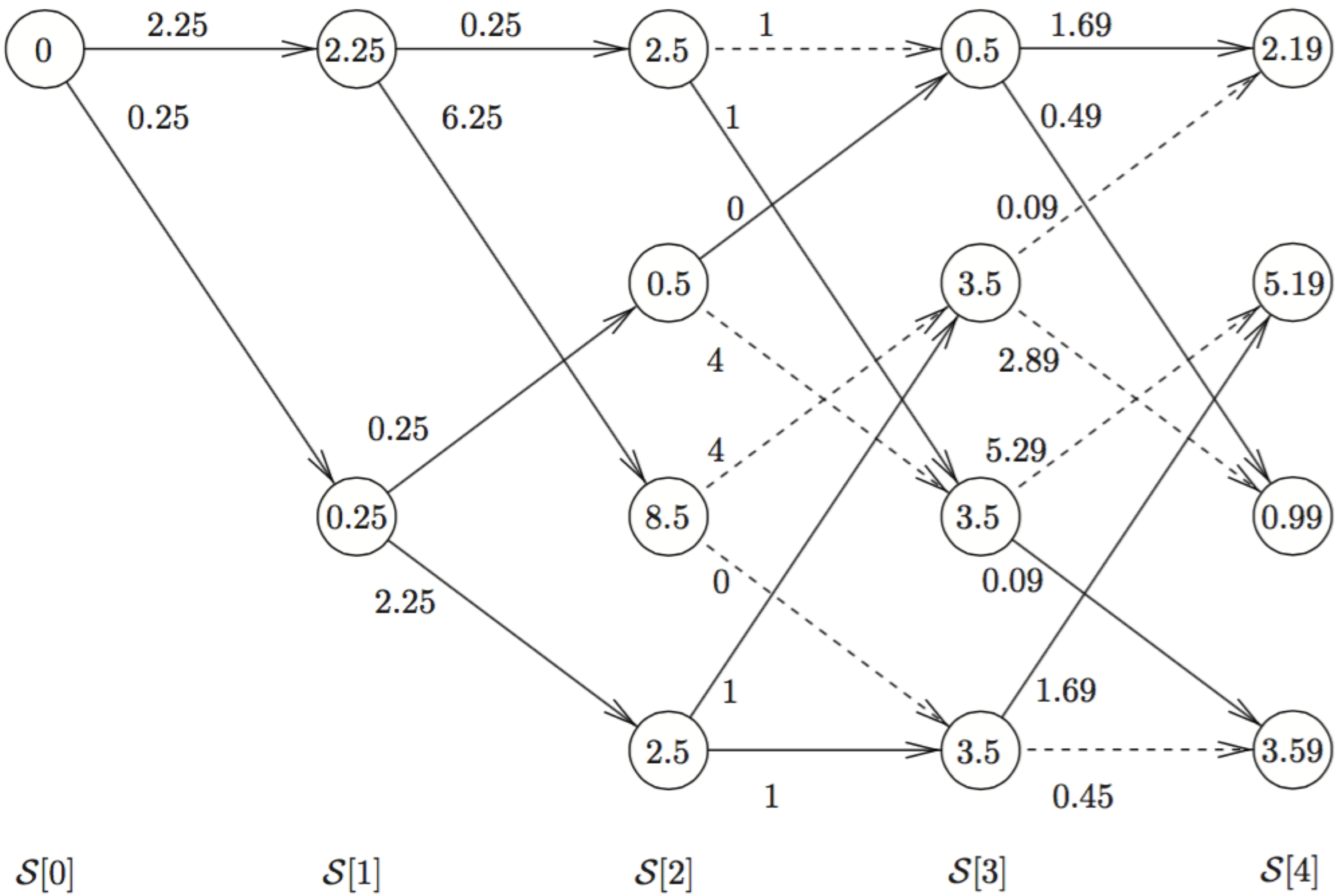
1. Draw trellis and compute transition metric for each path.
2. Accumulate transition metrics and identify the survivor at each state. This is done sequentially, from $k = 0$ to $k = V$.
3. Identify which of ending states has the smallest metric.
   - The path to this ending state is the ML sequence estimate.

Example (1/3)



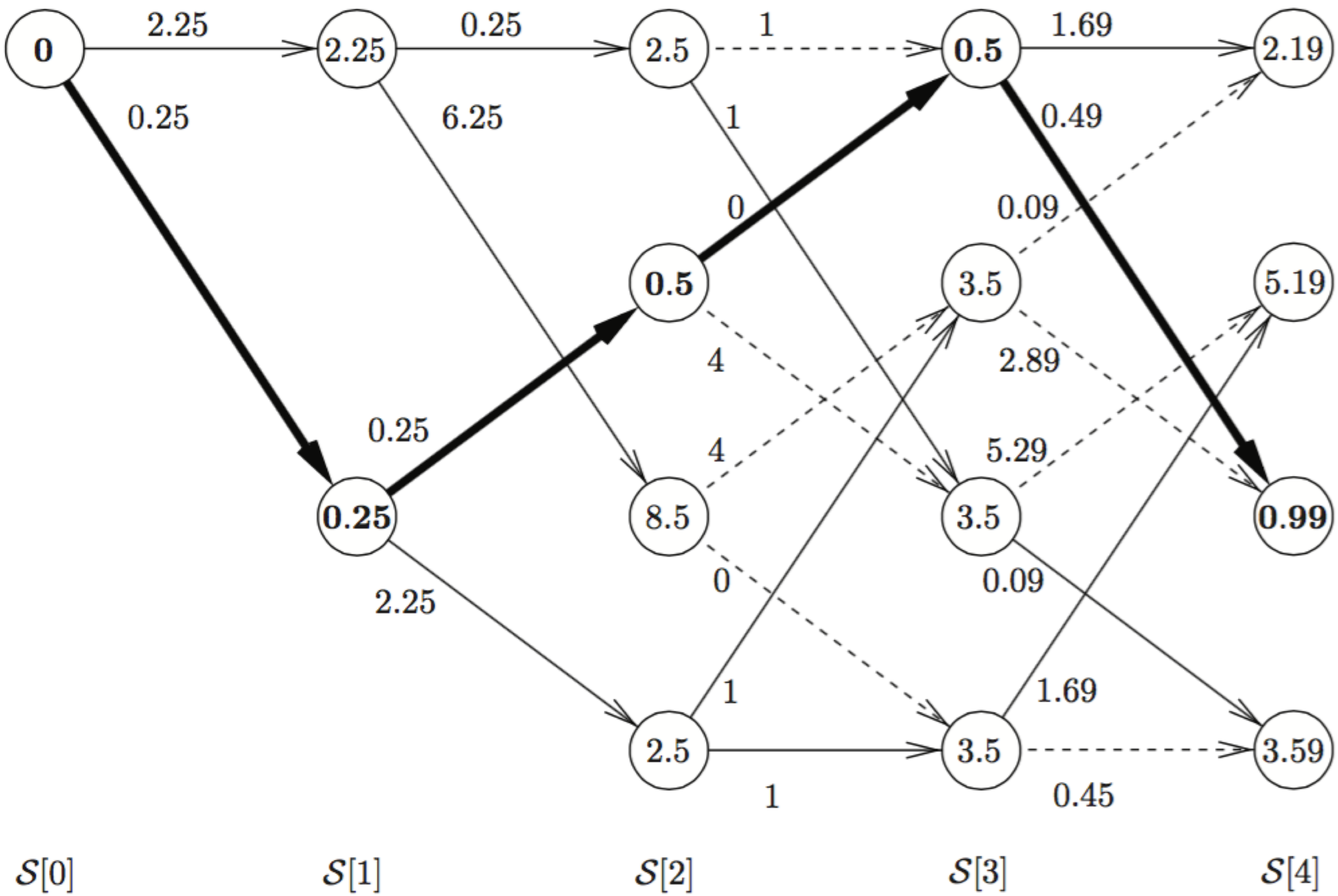
0, 0
0, 1
1, 0
1, 1
2.25
0.25
6.25
1
1.69
0.49
0
0.09
4
2.89
5.29
0.09
2.25
1
1.69
0.45
$\mathcal{S}[0]$
$\mathcal{S}[1]$
$\mathcal{S}[2]$
$\mathcal{S}[3]$
$\mathcal{S}[4]$

Example (2/3)



0
2.25
0.25
2.25
0.25
6.25
0.25
2.25
2.5
1
1
0.5
0
4
8.5
4
0
2.5
1
1
0.5
1.69
0.49
3.5
0.09
2.89
3.5
5.29
0.09
3.5
1.69
0.45
2.19
5.19
0.99
3.59
$\mathcal{S}[0]$
$\mathcal{S}[1]$
$\mathcal{S}[2]$
$\mathcal{S}[3]$
$\mathcal{S}[4]$

Example (3/3)



0
2.25
2.25
0.25
2.5
1
0.5
1.69
2.19
0.25
6.25
1
0.49
0
0.09
0.5
3.5
5.19
4
2.89
0.25
4
5.29
0.25
8.5
3.5
0.99
2.25
0
0.09
1
1.69
2.5
3.5
3.59
1
0.45
S[0]
S[1]
S[2]
S[3]
S[4]

Complexity Comparison

- Main computation
  - Transition metric $\Lambda_k(s[k], \mathcal{S}[k])$
- Brute force ML detection
  - Compute transition metric $V \cdot M^V$ times
- Viterbi algorithm
  - Number of states: $V \cdot M^{L-1}$
  - Paths into each state: M
  - Compute transition metric $V \cdot M^L$ times

Full metrics computed:
Brute-force: M^V
Viterbi: M^L

LINKÖPING UNIVERSITY

www.liu.se