A Fast Maximum-Likelihood Decoder for Convolutional Codes

Jon Feldman — jonfeld@mit.edu
Matteo Frigo — athena@vanu.com
Ibrahim Abou-Faycal — iaboufay@mit.edu

M.I.T.
Vanu, Inc.

Convolutional Codes

- Commonly used in TDMA/GSM cellular phones and other wireless standards.
- Simple linear time encoder.
- Viterbi algorithm is an optimal decoder, but:
  - Software radios cannot use parallelism,
  - Running time $\Theta(2^k n)$

    (k = "constraint length", n = code length).
- When SNR is high, decoding should be easier.
  - Software radios can take advantage of running time that (implicitly) depends on SNR...

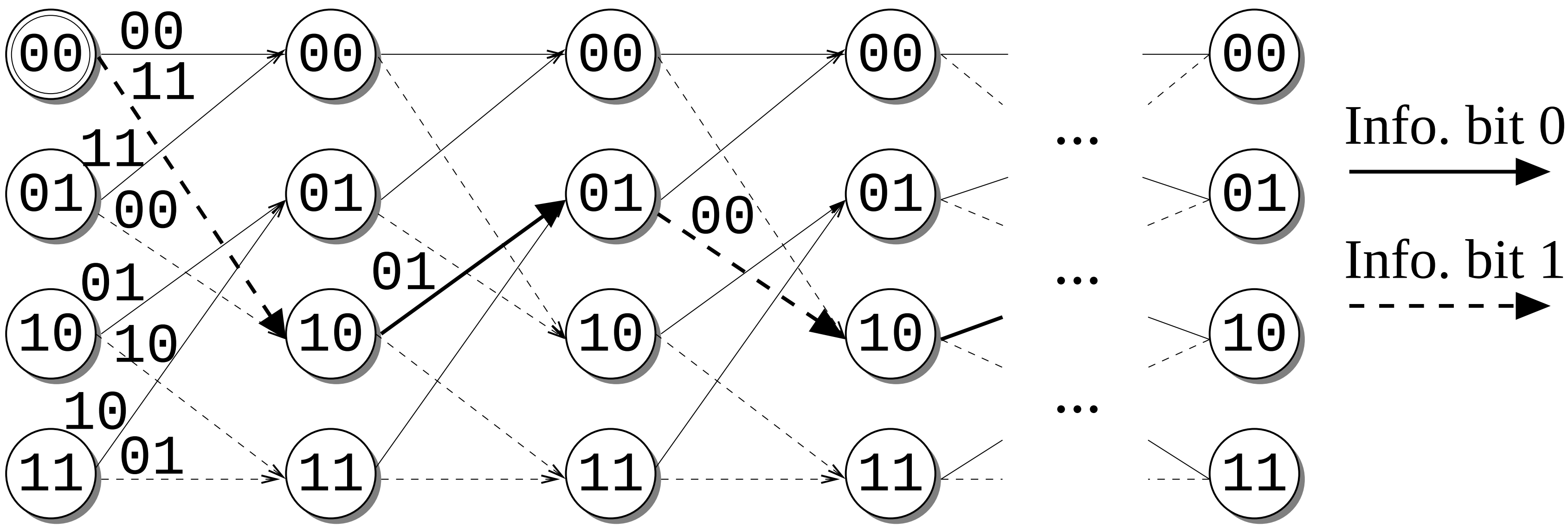
Convolutional Decoding Alternatives

- Sequential Decoding:
  - Suboptimal.
- A^* Decoding [ED '96, HHC '93, HCW '02]:
  - Time/block:
    $\Theta(n \log n) \ldots \Theta(2^k n(k + \log n))$ (high . . . low SNR).
  - Worst-case running time worse than Viterbi.
  - No experiments available to compare running time.
- Our "Lazy Viterbi" algorithm:
  - Time/block:
    $\Theta(n) \ldots \Theta(2^k n)$ (high . . . low SNR).
  - In software: up to 10 times faster than optimized Viterbi.
  - Works on "streaming" data (non-blocked data).

Outline

- Convolutional codes, Viterbi algorithm.
- A^* decoding, advantages and limitations.
- The Lazy Viterbi decoder.

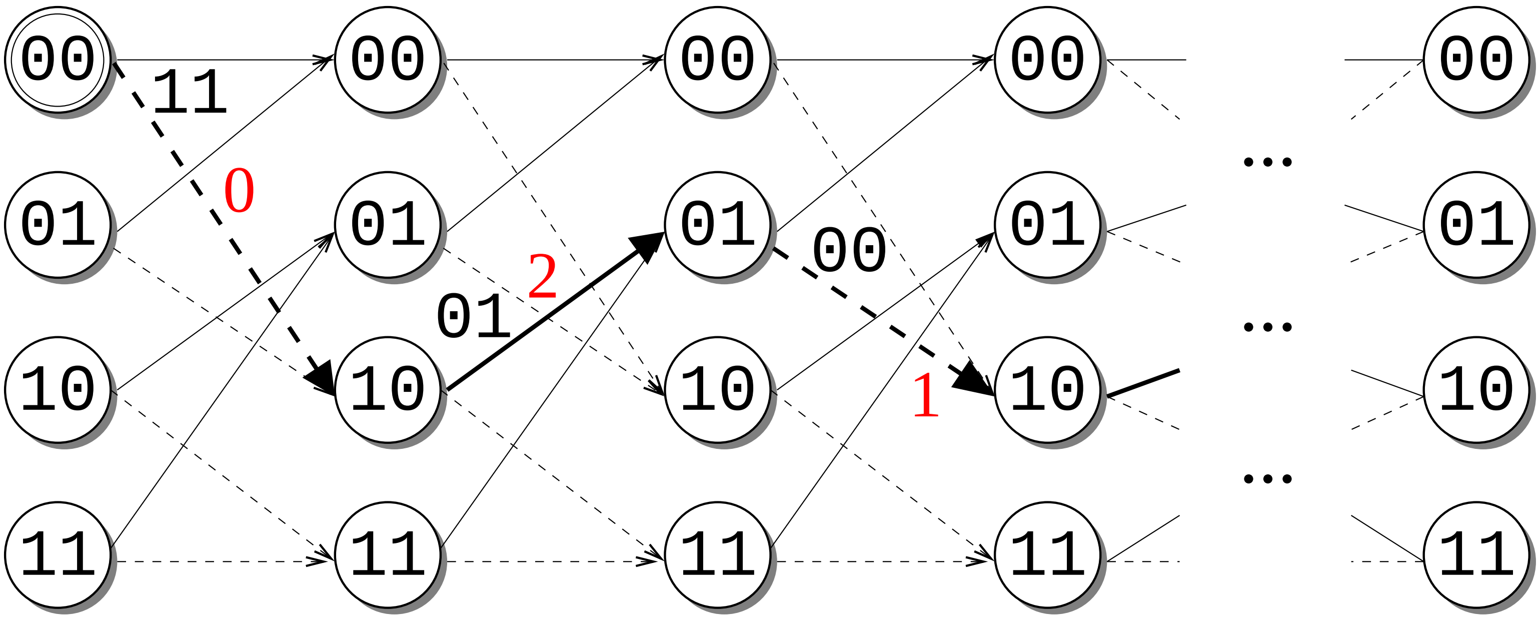
Convolutional codes: the trellis



Generator Polynomials: $1 + d^2$, $1 + d + d^2$

- Encoding: path through the trellis (labels = code bits).
- Info bits: $1, 0, 1, \cdots$
  States: $00 \rightarrow 10 \rightarrow 01 \rightarrow 10 \rightarrow \ldots$
  Code word: $1, 01, 00, \ldots$

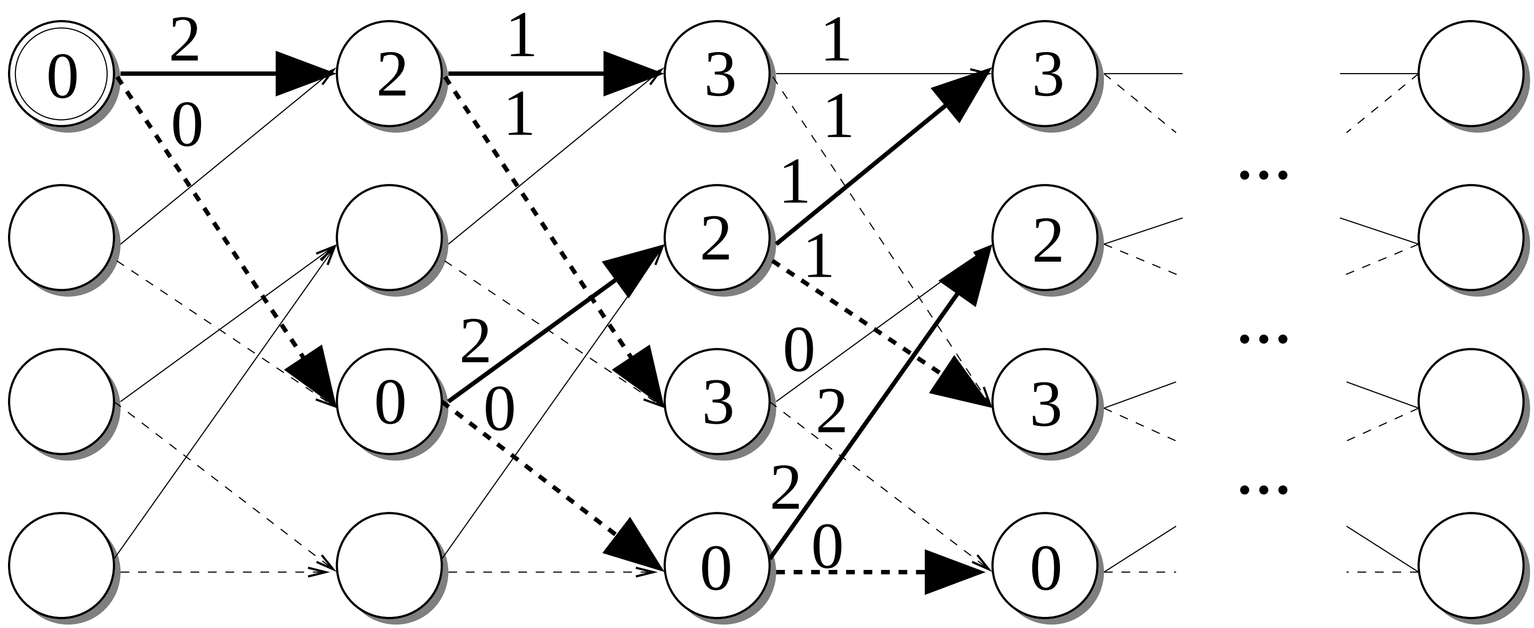
Viterbi Algorithm



Received: 11 10 01 ...

- Branch Metric of edge: δ(code bits, received bits).
- $\delta(11, 11) = 0$, $\delta(01, 10) = 2$, $\delta(00, 01) = 1$.
- Viterbi Algorithm: Find path with lowest total metric.

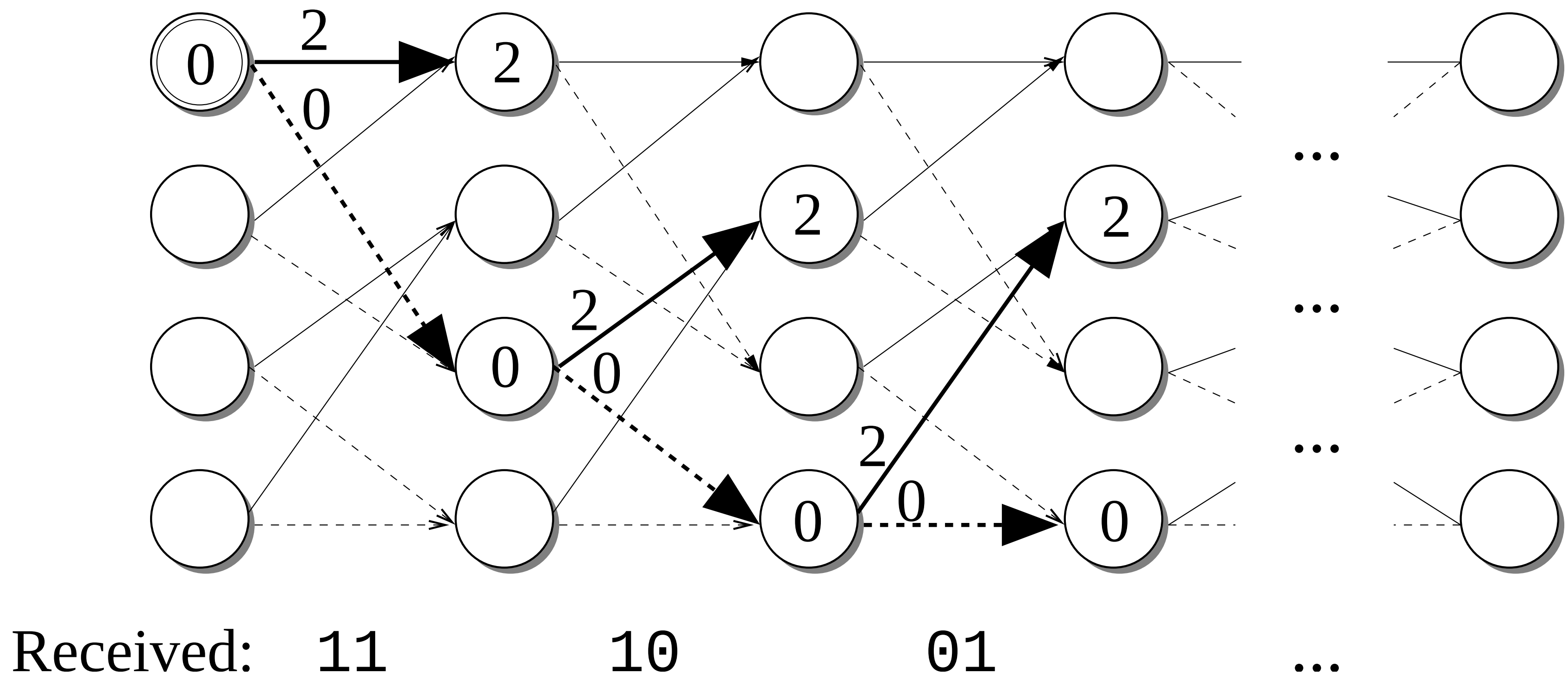
Viterbi Algorithm is BFS



Received: `11` `10` `01` ...

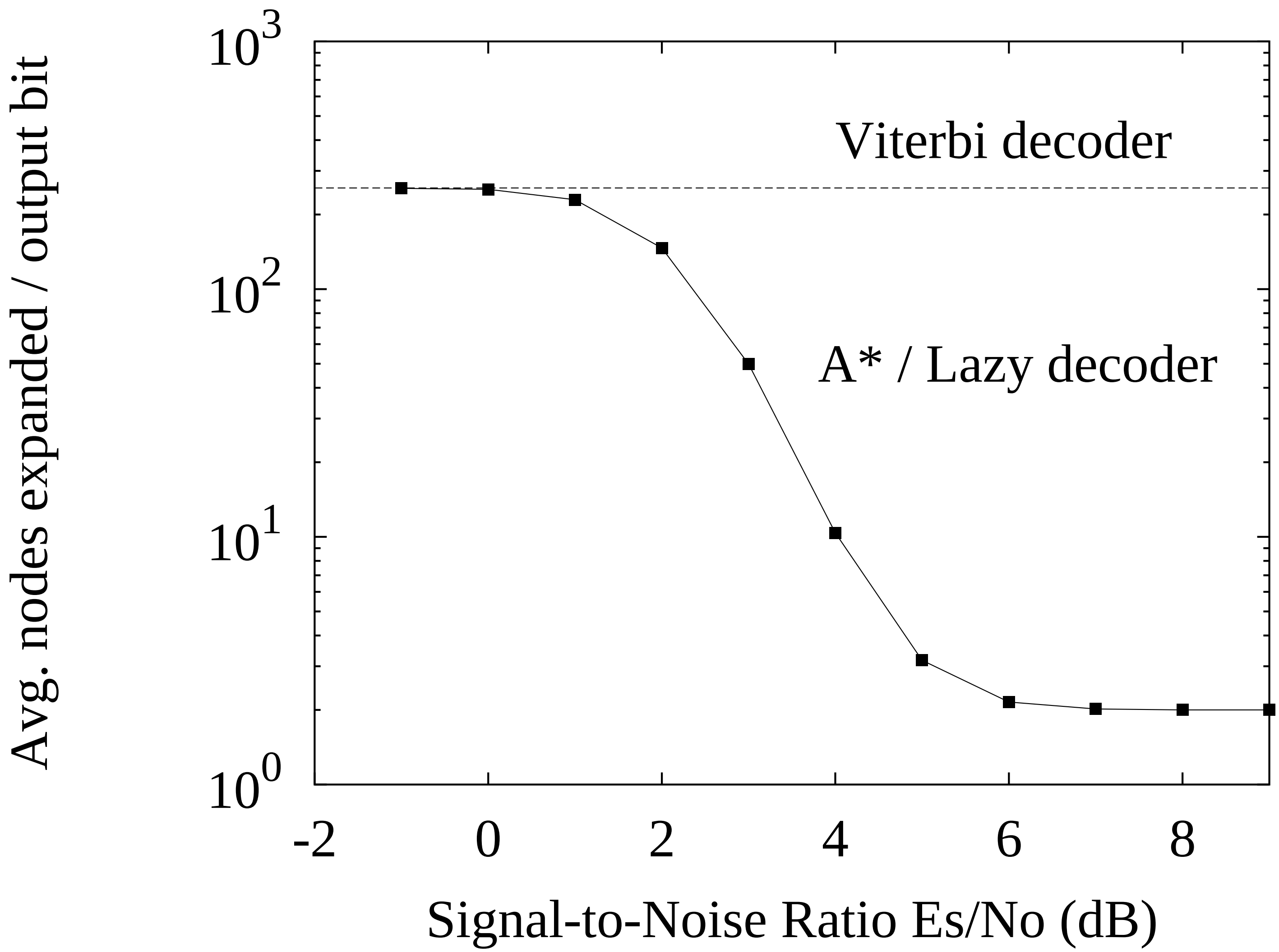
- Computes accumulated error for every node.
- Running time: $\Theta(n \cdot 2^k)$ (# states = 2^k).
- Breadth-First-Search (BFS) is only one possible "shortest-path" algorithm.

Dijkstra's Algorithm



- Dijkstra's Algorithm (Dijkstra, 1959):
  - Repeatedly "expand" the node w/ lowest metric.
- A^*: Dijkstra's algorithm with "heuristic" function.
- Running time $\approx$ (# expansions) $\times$ (time/expansion)

Trellis Expansions as a Function of SNR



AWGN, QPSK, Rate = $1/2$, $k = 9$ (CDMA), gen. polys (753,541) (octal).

Limitations of Previous Work

- Cost of deciding *which* node to expand:
  - Must maintain *priority queue* Q of trellis nodes.
  - Cost = $\Theta(\log |Q|)$ per operation.
  - Not amenable to compile-time optimization.
- High SNR: $|Q| \approx n$, time/block $\Theta(n \log n)$.
- Low SNR: $|Q| \approx 2^k n$, time/block $\Theta(2^k n \cdot \log(2^k n))$.
- Only defined for block codes (rather than streaming data).
- No implementations given, no evidence A^* useful in practice.

The Lazy Viterbi Decoder

- Constant time / expansion via new priority queue
  - ($\Theta(n \log n) \rightarrow \Theta(n)$).
- “Lazier” version of Dijkstra’s algorithm
  - (saves a few cycles per operation).
- Represent trellis as sparse matrix;
  - (saves memory, could improve cache performance).
- Extension to data streams (finite traceback length).
- Experimental study on different processors.

Experimental Comparison (best case)

Decoder	k	Pentium III	PowerPC 7400	StrongARM
Lazy	6	201	200	226
Karn Generic	6	1143	626	892
Viterbi Optimized	6	316	239	310
Lazy	7	205	203	232
Karn Generic	7	2108	1094	1535
Karn Optimized	7	558	486	641
Karn SSE	7	108	N/A	N/A
Lazy	9	235	225	343
Karn Generic	9	8026	3930	5561
Karn SSE	9	310	N/A	N/A

(processor cycles per information bit)

Conclusion

- A new fast convolutional decoder.
- Decodes optimally (maximum-likelihood).
- Generalizations:
  - Soft-decision decoding.
  - Data streams (non-block codes).
  - Equalizers, or any other application of Viterbi algorithm.
- Takes advantage of the flexibility of software radio.