A Brief History of Early Convolutional Coding

For forward error correction (FEC) coding in 1974, decoded information bit error rate (BER) and decoder complexity was everything. It was presumed, almost correctly, that minimizing decoded "codeword" error rate in linear codes equaled minimizing BER (not true, but close enough for the times). And as far as complexity went, the original "forward" Viterbi algorithm (VA) was the only maximum likelihood (ML) decoding rule known in 1973 for linear convolutional codes, and so its then-considerable complexity wasn't a gamebreaker. Simpler approaches, like sequential, feedback or threshold decoding, were not ML. But useful systems exploiting these alternative decoding rules got built anyway.

ML decoding requires comparing all the paths through a convolutional code trellis spanning the length of a "codeword" and selecting the best among them. This is precisely what makes Viterbi decoding optimal over codewords, vice globally optimal. Excepting the sadly unreachable maximum *a priori* (MAP) optimality of the merely-imagined BCJR (Bahl, Cocke, Jelenick and Raviv) decoding, all other realized decoding schemes, by selecting a merely "good" path by some criterion but not this best one, are sub-ML. Due to the essence of ML decoding resting on codewords, it must be noted that convolutional codeword length is a distribution, not a fixed number. The tail of this distribution is not limited. And so all ML decoding is, alas, only approximately ML, but it can be a very good approximation. Practical ML decoding was and is just called maximum likelihood, as if it was exactly so.

Among the several alternatives to ML convolutional decoding, a very simple scheme was feedback decoding. On its own, feedback decoding was not very strong as it depended on a very short "backward look" through the convolutional code trellis or tree (more on that later). Basically, a new code symbol arriving in conflict with previous code symbols could be corrected immediately to resolve the conflict and the correction "fed back" into the incoming codeword. The decoder then proceeds on until the next conflict. This might work for an isolated symbol error or two, but more complex error patterns would quickly induce failure of a realizable "conflict resolution table." So feedback decoding seemed, on first examination, to be useful only for "good" channels and fairly low (inherently strong) code rates but of low code complexity, i.e. low code constraint length (or short convolutional product length), allowing simple resolution tables. I.e., a waste of code rate. But this was misleading in the face of interleaving. Interleaving, a subject in itself, allows the "memory" of the channel to be smoothed out, or even destroyed, by averaging the distance in occurrence between de-interleaved channel errors. This means that the de-interleaved errors are mostly of the simple, single-symbol, isolated kind, perfect for resolution by a feedback decoder. A simple interstitial-encoder-stage interleaver got invented and made feedback decoding useful for over a decade, into the late 1970s at least. But the coding gain of feedback coding is, simply, inferior compared to most anything else (excepting perhaps threshold decoding, which is worse yet, ignored here and essentially unused, even then).

Decoded error rate had been analyzed in the 1960s using exponential bounds to reveal, for a given code rate, the superiority of convolutional over algebraic codes of the same rate, a result that stands. Nevertheless, a rich landscape of algebraic codes, and their sometimes simple decoding rules, was advantageously applied to data structures both before or after channel decoding using convolutional codes. Applying a second code "outside" of an "inner" convolutional code was called "concatenated coding," and an "outer" algebraic code version was common in FEC before 1980. Still later, algebraic codes such as Reed-Muller were used stand-alone to reliably transfer meta-data about subsequent channel coding parameters and other aspects of the physical layer transmission format.

Our subject is convolution coding. But as an important aside, algebraic coding is not at all dead, but now more often an accessory than alternative to convolutional coding. This of course ignores the modern reprise of the Gallagher block codes, now known as low density parity check (LDPC) codes, closer to algebraic codes than to convolutional. The LDPC codes, for the long block lengths that make them valuable, are a viable alternative to either algebraic or convolutional codes in some applications. Just where algebraic and iteratively-decoded LDPC codes will be in another generation of FEC from now is perhaps as unknown as where turbo convolutional coding would be after its first few years following their discovery in 1992. Indeed, mythology has it that Viterbi himself did not initially believe, for a very short while, that turbo-coding could be "real." Turbo codes and the Belief Propagation algorithm were alien to the ironically deterministic world of FEC algorithms in those first few years.

Now back to the formative, if not dominant emergence of Viterbi decoding. Decoding by VA of convolutional, i.e. regular trellis, codes was and remains the only literal ML procedure, since MAP decoding (e.g., BCJR) does not contain an explicit "select" (and discard) step on the way to finding a best encoder state, vice information decision. In applying the VA, there is a successively-progressing computation at every state for each depth of the trellis called add-compare-select, or ACS, an monogram made famous by Linkabit Corporation in the late 1960s and early 70s. It became widely adopted by any practitioner who got into convolutional coding as ACS wasn't patented. That corporate strategy would change by the 1990s as patents began to reign supreme, and pervasive in coding, as in virtually all other technologies.

Oddly, Viterbi's likelihood rule, or VA, was only one "part" of Viterbi decoding. The other part, "tracing back" *efficiently* from a "best" selection by likelihood ratio to the output decisions, was recognized and described by Viterbi. But no easy way of doing this potentially awkward reaching back was found until Viterbi's colleague J. A. Heller invented a proprietary trace back algorithm, called "chaining back," in 1968 or so on the very heels of the algorithm paper. Once encountered, this trace-back scheme would be readily apparent to the cognosenti but improbably was not reverse-engineered for a decade. The two algorithmic parts were now efficiently

combined to make Viterbi decoding the powerhouse of channel decoding, and in other applications as well, for the next two decades.

There was a powerful alternative to Viterbi decoding of convolutional codes, way beyond the early feedback decoding idea. Sequential decoding, in which paths along a code's "tree" (the unraveled version of its trellis) were "traveled" in a dynamic "search" forward and backward for an acceptable codeword path, that path being the first which "solved" the requisite received corrupted codeword correction, was known at least as early as 1962. One could say that sequential decoding vaulted convolutional coding into stardom even before the VA appeared. Tree search sequential decoding was much lower computational complexity than the VA, and so it was before and would remain a popular implementation. This is because only the tiniest fraction of a code tree, and hence trellis as well, required searching, allowing hugely-longer code constraint lengths (K) to be used. In this way, sequential decoding could achieve a coding gain for a given code rate exceeding, theoretically at least, the best short-K Viterbi decoders that could be built in the same timeframe. But this promise was compromised by the requirement to very occasionally search very deep into the tree.

Sequential decoding used the Fano algorithm or some equivalent search scheme (e.g., the Stack algorithm). For a fixed computation clock-to-data rate ratio, deep searches require a commensurate decoding delay. This throughput decoding delay and depth backward into the tree is reflected in a "threshold" probability metric that is computationally asserted against the metric result of a syndrome decoder, as opposed to a trellis path memory. This is the second main distinction between sequential and VA decoding. The details of metrics are beyond our scope here.

Because sequential decoding tree searches occupy time, a buffer memory is mandatory in order to hold incoming channel data long enough that a decision (on a badly corrupted codeword) can be made and the algorithm finally moved forward to the latest incoming channel data. However, if the search is long enough, the available buffer memory is overrun by the relentless incoming channel data, and sequential decoding "fails." In that case, the search must be abandoned and undecoded, corrupted channel bits mapped as best as possible for the "failed" codeword to the decoded output, with a concomittant burst error hit to the BER. This failure-handling mechanism allows the decoding to pick up and proceed onward, past the failed segment, to the rest of the channel data. Buffer memory failure and its burst of output errors is characterized by a Pareto random variable and distribution. It is mainly Pareto-distributed buffer failures, not computational decoding errors, that dominate sequential decoding performance.

In fairness, sequential decoding with very long K could vie with short-K Viterbi decoding in BER for a given SNR up to the 1980s. But practical decoders must limit buffer memory size for reasons of both cost and allowable thoughput delay. And very high data rates tend to demand unreachable computation clock-to-data rate ratio. It comes down to economic and operational limitations in specific applications. Both schemes were used for decades, but Viterbi decoding dominated

numerically in applications preferring it, and winning in a growing way with ever-growing channel data rates. For those who argue that sequential decoding should get a reprise for IP packet communication, it is pointed out that packet decoding is also natural for turbo- and LDPC coding, both "capacity approaching" (for different individual reasons) and stronger than any single-iteration FEC like sequential decoding. Sequential decoding is now essentially historical.