# Supplemental Material for TKDE-2011-02-0065

Yi Zhang, Kristian Lum, and Jun Yang

## 1 Coping with Topology Changes

We distinguish two types of topology changes. Changes in the underlying network topology do not affect our suppression and redundancy schemes, because we operate at the level of suppression edges, which we do not assume to be direct communication edges.

Changes in the suppression topology can be handled in a straightforward manner. For each suppression edge $i \to j$ that we want to change or create, the base station first learns a new suppression model from the past values of the readings involved (there is no need for any pilot runs because we have already been collecting data). Then, we send the new model to both $i$ and $j$, which start tracking the suppression state and the transmission history map anew for the suppression edge.

Note that changing the topology of cascaded suppression likely involves multiple suppression edges, because we must ensure the property that for each sensor $k$, there is a path consisting of suppression edges involving $k$'s reading to the base station, and the error bounds along the path sum up to no more than $\epsilon$. Hence, frequent changes to the suppression topology may be costly. In practice, however, for many environmental sensing applications, the sensors are carefully designed and deployed to mitigate the need for frequent topology changes. In particular, cluster heads and those with power-hungry sensors carry more battery capacity, in anticipation of imbalanced energy use. This design effectively avoids topology changes due to load imbalance.

## 2 The Decoding Algorithm

Algorithm 1 is our decoding algorithm. During execution, for each column $i$ in the trellis diagram, the algorithm keeps track of a candidate set containing all possible state transitions from column $i-1$ to column $i$. The algorithm takes two kinds of actions: forward and backward. Upon receiving a message at time $t$, it first takes the forward action, creating a candidate set for $t+1$ that includes every edge originating from any target state in the last candidate set and having an output label matching the received message; if the message at $t$ is lost, all possible outgoing edges are included. Whenever there is any target state in the last candidate set that is a "dead end," i.e., no edge in the current candidate set is incident to it, we take a backward action to eliminate that state, which can go recursively into the past. In essence, the backward action uses the current received message to help eliminate some possibilities in the past. After the forward and backward actions, if all edges in a candidate set share the same input label, then we have successfully decoded the input for that timestep.

**Algorithm 1:** Decoding algorithm.

**Input**: current time $t$, received coded message $m_{\text{out}}$, and $C_i$'s ($i \le t$), the candidate (edge) sets
**Output**: updated $C_i$'s, new $C_{i+1}$, and decoded messages, if any

```
C_{t+1} ← ∅;                                              // forward step
foreach state x such that (·, x) ∈ C_t do
    foreach edge (x, y) in the trellis diagram do
        if m_out is lost or (x, y) is labeled with output m_out then
            S_{t+1} ← S_{t+1} ∪ {(x, y)};
i ← t;                                                    // backward step
while {y | (·, y) ∈ C_i} \ {x | (x, ·) ∈ C_{i+1}} ≠ ∅ do
    remove all (·, y) from C_i where y ∉ {x | (x, ·) ∈ C_{i+1}};
    if all (x, y) ∈ C_i are labeled with the same input w then
        output w as the input at time i;
    i ← i − 1;
```

## 3 Why Our Application of Convolutional Coding is Unique

Our application of convolutional coding departs from its standard usage (such as Viterbi [1] and Fano [2] decoding) in several important ways. First, our decoding algorithm recovers inputs that we are absolutely certain of. If there exist multiple possibilities for a particular input, we would not make any assertion on this input. This feature is important in our setting because we do not want to make guesses with potentially big impact on data interpretation but without solid grounding. For example, one standard decoding approach is based on maximum likelihood (e.g., Viterbi), but it has to assume a channel model. Unfortunately, transient message failures in sensor networks remain poorly understood and difficult to model because of the complex interplay of


- *Y. Zhang and J. Yang are with the Department of Computer Science, Duke University, Durham, NC 27708. Email: {yizhang, junyang}@cs.duke.edu*
- *K. Lum is with the Institute of Mathematics at UFRJ, Brazil. Email: kristian@dme.ufrj.br*

hardware and software issues and (often unpredictable) environmental factors. Thus, we have taken a different, more conservative approach that makes no assumption about the failure model.

Second, on the algorithmic level, traditional decoding approaches typically try to find a single path in the trellis diagram, either the most likely one, or the one with a limited number of decoding errors (such an error occurs when an output does not match any edge going out of the current state). Such approaches are consistent with the assumptions (e.g., reasonable channel models) and requirements (e.g., handling bit-level errors, or having to recover the entire input sequence) in typical uses of convolutional coding. In contrast, with none of these assumptions and requirements, our decoding algorithm focuses on identifying unambiguous inputs without committing to a single path.

## REFERENCES

[1] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[2] R. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transactions on Information Theory*, vol. 9, no. 2, pp. 64–74, 1963.