# Supplementary Material for: Learning to Decode: Reinforcement Learning for Decoding of Sparse Graph-Based Channel Codes

## S.1 Proof of Theorem 1

The proofs are ordered according to the claims in the theorem as follows.

1. There are $p$ VN neighbors of a CN $c \in C_u$, and none of these VN neighbors can share another CN neighbor due to the absence of 4-cycles. Thus, the number of VNs in $\mathcal{N}_V(c)$ that have no neighbors outside of $C_u$ is at most $(z-1)/2$ (each of these VNs has two other neighbors in the remaining $z-1$ CNs of $C_u$, and all are distinct). Thus, there are at least $p-(z-1)/2$ VNs adjacent to $c$ that have at least one neighbor outside of $C_u$. This is true for all choices of $c$. Since we may be counting each of these VNs up to two times, there are at least $(p-[(z-1)/2])z/2$ VNs in $W$. The result follows.
2. If all the rows corresponding to the CNs of cluster $C_u$ are in the same row group, then no two CNs in $C_u$ will have any VNs in common. Hence, each VN in $\mathcal{N}_V(C_u)$ must also be adjacent to $\bar{C}_u$, implying that $\mathcal{N}_V(C_u)$ is a CCS with $|\mathcal{N}_V(C_u)| = |C_u|p = zp$.
3. Suppose that the CNs in $C_u$ form triples, and let $\phi = \{\{c_1, c_2, c_3\}, \ldots, \{c_{z-2}, c_{z-1}, c_z\}\}$, $|\phi| = z/3$, be a set of all these triples. Suppose that each CN triple in $\phi$ is associated to a non-overlapping 6-cycle. Since, in this case, there will be $|\phi|$ non-overlapping 6-cycles in the Tanner graph induced by $C_u \cup \mathcal{N}_V(C_u)$, and each 6-cycle has 3 VNs, the number of VNs that have degree 2 with respect to $C_u$ will be $|\phi| \times 3 = z$. Note that there are $zp$ edges emanating from $C_u$ since each CN degree is $p$. Hence, in the worst case, there will be $zp - z$ distinct VNs in $\mathcal{N}_V(C_u)$, which is also a CCS since each of these degree 3 VNs are also connected to $\bar{C}_u$.

□

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

## S.2 Algorithm 1

**Algorithm 1:** MAB-NS for LDPC codes

**Input :** $\mathbf{L}$, $\mathbf{H}$
**Output:** reconstructed signal

Initialization:
  $\ell \leftarrow 0$
  $m_{c \to v} \leftarrow 0$ // for all CN to VN messages
  $m_{v \to c} \leftarrow L_v$ // for all VN to CN messages
  $\hat{\mathbf{L}}_\ell \leftarrow \mathbf{L}$
  $\hat{\mathbf{S}}_\ell \leftarrow \mathbf{H}\hat{\mathbf{L}}_\ell$
**foreach** $a \in [[m]]$ **do**
  $s_\ell^{(a)} \leftarrow g_M(\hat{s}_\ell^{(a)})$ // $M$-level quantization
**end**
// decoding starts
**if** *stopping condition not satisfied or* $\ell < \ell_{\max}$ **then**
  $s \leftarrow$ index of $\mathbf{S}_\ell$
  select CN $a$ according to an optimum scheduling policy
  **foreach** *VN* $v \in \mathcal{N}(a)$ **do**
    compute and propagate $m_{a \to v}$
    **foreach** *CN* $c \in \mathcal{N}(v) \setminus a$ **do**
      compute and propagate $m_{v \to c}$
    **end**
    $\hat{L}_\ell^{(v)} \leftarrow \sum_{c \in \mathcal{N}(v)} m_{c \to v} + L_v$ // update LLR of $v$
  **end**
  **foreach** *CN* $j$ *that is a neighbor of* $v \in \mathcal{N}(a)$ **do**
    $\hat{s}_\ell^{(j)} \leftarrow \sum_{v' \in \mathcal{N}(j)} \hat{L}_\ell^{(v')}$
    $s_\ell^{(j)} \leftarrow g_M(\hat{s}_\ell^{(j)})$ // update syndrome $\mathbf{S}_\ell$
  **end**
  $\ell \leftarrow \ell + 1$ // update iteration
**end**

## S.3 Details of Algorithm 2

**Algorithm 2:** Clustered Q-learning

**Input :** $\mathscr{L}$, $\mathbf{H}$
**Output:** Estimated $Q_{\ell_{\max}}(s_u, a_u)$ for all $u$

Initialization: $Q_0(s_u, a_u) \leftarrow 0$ for all $s_u$, $a_u$ and $u$
**for** *each* $\mathbf{L} \in \mathscr{L}$ **do**
    $\ell \leftarrow 0$
    $\hat{\mathbf{L}}_\ell \leftarrow \mathbf{L}$
    $\hat{\mathbf{S}}_\ell \leftarrow \mathbf{H}\hat{\mathbf{L}}_\ell$
    **foreach** $a \in [[m]]$ **do**
        $s_\ell^{(a)} \leftarrow g_M(\hat{s}_\ell^{(a)})$ // $M$-level quantization
    **end**
    **while** $\ell < \ell_{\max}$ **do**
        schedule CN $a_u$ according to an $\epsilon$-greedy approach
        select $u$ as cluster index of CN $a_u$
        $\mathbf{S}_\ell^{(u,z)} \leftarrow s_\ell^{(j_1)}, s_\ell^{(j_2)}, \ldots, s_\ell^{(j_z)}$ // CN indices $j_1, j_2, \ldots, j_z \in \{0, \ldots, m-1\}$ are in ascending order for MQC, and randomly ordered for MQR. For MQO, their ordering depends on the underlying cluster $\tilde{C}_u^*$
        $s_u \leftarrow$ index of $\mathbf{S}_\ell^{(u,z)}$
        **foreach** *VN* $v \in \mathcal{N}(a_u)$ **do**
            compute and propagate $m_{a_u \to v}$
            **foreach** *CN* $c \in \mathcal{N}(v) \setminus a_u$ **do**
                compute and propagate $m_{v \to c}$
            **end**
            $\hat{L}_\ell^{(v)} \leftarrow \sum_{c \in \mathcal{N}(v)} m_{c \to v} + L_v$ // update LLR of $v$
        **end**
        **foreach** *CN* $j$ *that is a neighbor of* $v \in \mathcal{N}(a_u)$ **do**
            $\hat{s}_\ell^{(j)} \leftarrow \sum_{v' \in \mathcal{N}(j)} \hat{L}_\ell^{(v')}$
            $s_\ell^{(j)} \leftarrow g_M(\hat{s}_\ell^{(j)})$ // update syndrome $\mathbf{S}_\ell$
        **end**
        $s'_u \leftarrow$ index of updated $\mathbf{S}_\ell^{(u,z)}$
        $R_\ell(s_u, a_u, s'_u) \leftarrow$ highest residual of CN $a_u$
        compute $Q_{\ell+1}(s_u, a_u)$
        $\ell \leftarrow \ell + 1$ // update iteration
    **end**
**end**