# Reinforcement Learning for Improving the Performance of Recovery Algorithm in Noisy Group Testing

Seyedeh Nahid Esmati, Arghamitra Talukder, and Ya Mei

Department of Electrical and Computer Engineering

Texas A&M University

Professor: Dr. Dileep Kalathil

May 5, 2021

# Tasks Done by Each Team Member

## Seyedeh Nahid Esmati:

- Implementing advantage actor-critic policy gradient with four different notions of states, and four different notions of rewards as stated in the slides.
- For the baseline work on GT and LDPC: Implementing the BP recovery algorithm (without RL) with flooding and maximum-residual node scheduling policies for both GT and LDPC, in order to reproduce the benchmarks.
- For the baseline paper on LDPC: Implementing Q-learning with direct update rule (no function approximation)+clustering+quantization.

## Arghamitra Talkuder:

- Implementing DQN with multiple agents (NNs) to learn multiple action-value functions in parallel (one agent for each BP iteration) for the state and reward definitions stated in the slides.
- Monte-Carlo simulation to reproduce the benchmark results for the BP algorithm with maximal-residual node scheduling policy for GT.

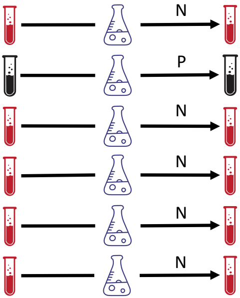
## Ya Mei:

- Implementing DQN with a single agent (NN) to learn a universal action-value function (one agent for all BP iterations) with two different notions of states, and two different notions of rewards as stated in the slides.
- Monte-Carlo simulation to reproduce the benchmark results for the BP algorithm with flooding for GT.

# Group Testing

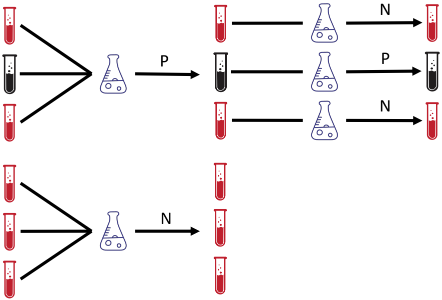
Design testing algorithms by using (binary) tests to identify all defective items among a much larger group of items.

Individual Testing




$\#$ tests (per item) $= 1$

Dorfman's Testing (1943)



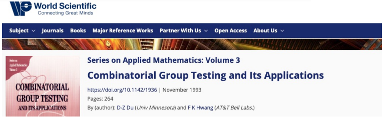

$\#$ tests (per item) $= \frac{5}{6}$

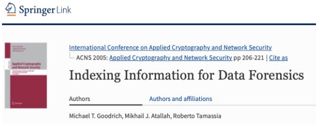
How to minimize $\#$ tests for an unknown config. of def. items?

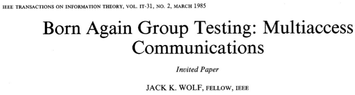
# Applications of Group Testing


World Scientific
Connecting Great Minds
Subject Journals Books Major Reference Works Partner With Us Open Access About Us
Series on Applied Mathematics: Volume 18
Pooling Designs and Nonadaptive Group Testing
Important Tools for DNA Sequencing
https://doi.org/10.1142/6122 | June 2006
Pages: 248
By (author): Ding-Zhu Du (*University of Texas at Dallas, USA & Xi'an Jiaotong University, China*) and Frank K Hwang (*National Chiao Tung University, Taiwan, ROC*)


Springer Link
Genetic Mapping and DNA Sequencing pp 133-154 | Cite as
A Comparative Survey of Non-Adaptive Pooling Designs
Authors Authors and affiliations
D. J. Balding, W. J. Bruno, D.C Torney, E. Knill


World Scientific
Connecting Great Minds
Subject Journals Books Major Reference Works Partner With Us Open Access About Us
Series on Applied Mathematics: Volume 3
Combinatorial Group Testing and Its Applications
https://doi.org/10.1142/1936 | November 1993
Pages: 264
By (author): D-Z Du (*Univ Minnesota*) and F K Hwang (*AT&T Bell Labs.*)


Springer Link
International Conference on Applied Cryptography and Network Security
ACNS 2005: Applied Cryptography and Network Security pp 206-221 | Cite as
Indexing Information for Data Forensics
Authors Authors and affiliations
Michael T. Goodrich, Mikhail J. Atallah, Roberto Tamassia

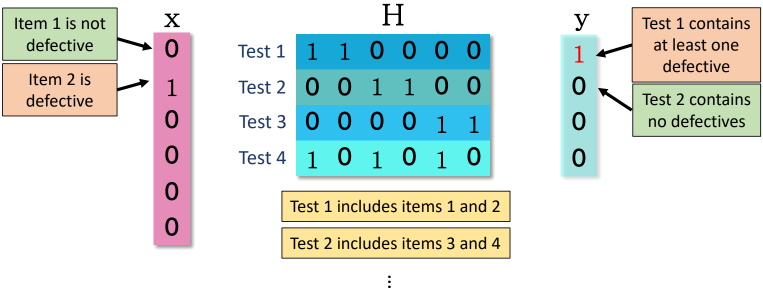

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-31, NO. 2, MARCH 1985
Born Again Group Testing: Multiaccess Communications
*Invited Paper*
JACK K. WOLF, FELLOW, IEEE


- Public health
- Cybersecurity
- Bloom filters
- Data science
- Information theory


nature
Explore content Journal information Publish with us Subscribe
nature > news > article
NEWS · 10 JULY 2020
The mathematical strategy that could transform coronavirus testing
Four charts show how pooling samples from many people can save time or resources.

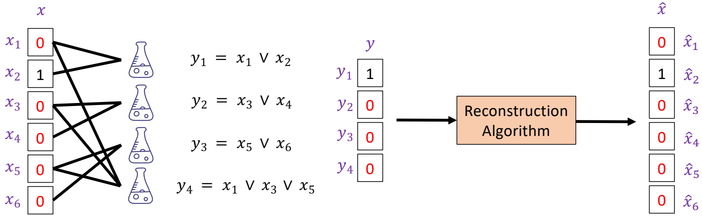
# Problem Setup

- Population of $n$ items (labeled $1, \ldots, n$)
- $k \ll n$ are defective (i.e., probability of defective $= q = \frac{k}{n}$)
- $\mathrm{x} = [x_1, \ldots, x_n]$: binary vector representing status of the items
- $H \in \{0, 1\}^{m \times n}$: binary matrix representing the testing matrix

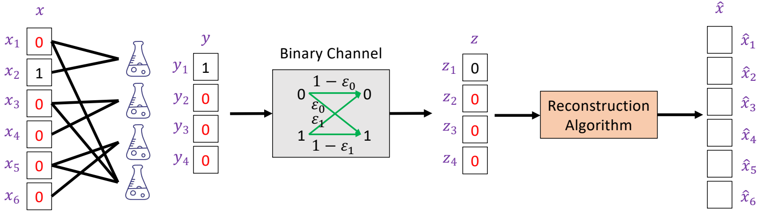


# Noiseless Setting vs. Noisy Setting

## Noiseless:




## Noisy:




**BSC** ($\epsilon_0 = \epsilon_1 = \delta$); Z-channel ($\epsilon_0 = 0$); reverse Z-channel ($\epsilon_1 = 0$).

# Exact Recovery vs. Partial Recovery

**Exact Recovery:** (no false negatives/positives)

- **Requirement:** $\hat{x} = x$;
- **Success probability:** $\Pr(\hat{x} = x)$.

Partial Recovery: ($\leq d_{\max}$ false neg.'s and $\leq d_{\max}$ false pos.'s)

- Requirement: $d(x, \hat{x}) \leq d_{\max}$, where

  $$d(x, \hat{x}) = \max\{|\mathrm{supp}(x) \setminus \mathrm{supp}(\hat{x})|, |\mathrm{supp}(\hat{x}) \setminus \mathrm{supp}(x)|\}$$

  and

  $$\mathrm{supp}(x) = \text{index set of all nonzero components of } x.$$

- Success probability: $\Pr(d(x, \hat{x}) \leq d_{\max})$.

# Recovery Algorithm: Belief Propagation (BP)

Check-to-variable messages: $m^{(l)}_{c\to v}(u_v)$

If $z_c = 0$:

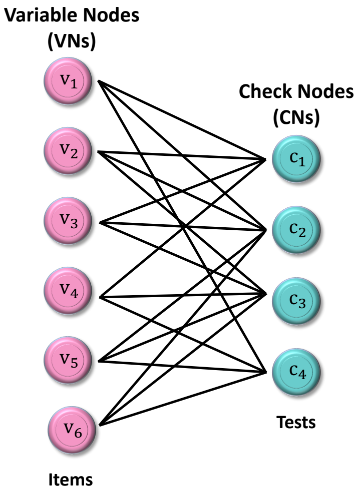
$$\propto \begin{cases} \delta & u_v = 1 \\ \delta + (1-2\delta) \prod_{v' \in \mathcal{N}(c)\setminus\{v\}} m^{(l-1)}_{v'\to c}(0) & u_v = 0 \end{cases}$$

If $z_c = 1$:

$$\propto \begin{cases} 1-\delta & u_v = 1 \\ 1-\delta-(1-2\delta) \prod_{v' \in \mathcal{N}(c)\setminus\{v\}} m^{(l-1)}_{v'\to c}(0) & u_v = 0 \end{cases}$$

Variable-to-check messages: $m^{(l)}_{v\to c}(u_v)$

$$\propto \begin{cases} q \prod_{c' \in \mathcal{N}(v)\setminus\{c\}} m^{(l)}_{c'\to v}(u_v) & u_v = 1 \\ (1-q) \prod_{c' \in \mathcal{N}(v)\setminus\{c\}} m^{(l)}_{c'\to v}(u_v) & u_v = 0 \end{cases}$$



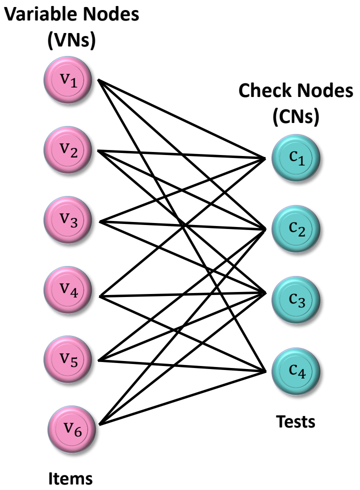
# Recovery Algorithm: Belief Propagation (BP)

Log Likelihood Ratio (LLR): $L_v^{(l)}$

$$= \begin{cases} \ln \dfrac{q}{1-q} & l = 0 \\ \ln \dfrac{q}{1-q} + \sum\limits_{c \in \mathcal{N}(v)} \ln \dfrac{m_{c \to v}^{(l)}(1)}{m_{c \to v}^{(l)}(0)} & l \geq 1 \end{cases}$$

Decoding Rule:

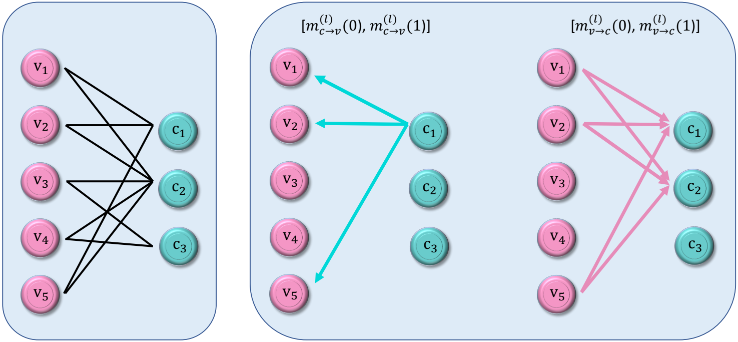
- If $L_v^{(l)} > 0$: $\hat{x}_v^{(l)} = 1$
- If $L_v^{(l)} = 0$: $\hat{x}_v^{(l)} = 1$ or $0$ w.p. $\frac{1}{2}$
- If $L_v^{(l)} < 0$: $\hat{x}_v^{(l)} = 0$



# Flooding vs. Node Scheduling

Flooding: In each iteration, all CNs send messages to their neighboring VNs, and all VNs send messages to their neighboring CNs.

Node Scheduling: In each iteration, a single CN sends messages to its neighboring VNs, and these VNs send messages to their neighboring CNs.




**Goal:** Given testing matrix $H$, # defectives $k$, and crossover prob. $\delta$, find an optimal node scheduling policy that maximizes the success probability.

# Apply RL to Learn Optimal Node Scheduling Policy

- Scheduling a single check node at each iteration of the BP algo. can be viewed as a Multi-Armed Bandits (MAB) process.
- State:
  (1) The sum of the LLRs of the VNs connected to each CN,
  $$\left[\sum_{v\in\mathcal{N}(1)} L_v^{(l)}, \sum_{v\in\mathcal{N}(2)} L_v^{(l)}, \ldots, \sum_{v\in\mathcal{N}(m)} L_v^{(l)}\right],$$
  where
  $$L_v^{(l)} = \begin{cases} \ln \frac{q}{1-q} & l = 0 \\ \ln \frac{q}{1-q} + \sum_{c\in\mathcal{N}(v)} \ln \frac{m_{c\to v}^{(l)}(1)}{m_{c\to v}^{(l)}(0)} & l \geq 1 \end{cases}.$$
  (2) The LLRs of the VNs connected to each CN,
  $$\left[\{L_v^{(l)}\}_{v\in\mathcal{N}(1)}, \{L_v^{(l)}\}_{v\in\mathcal{N}(2)}, \ldots, \{L_v^{(l)}\}_{v\in\mathcal{N}(m)}\right].$$
  (3) A quantized version of (1) or (2);
  (4) (1), (2), or (3) + the observed noisy test results $\hat{y}_1, \ldots, \hat{y}_m$.

# Apply RL to Learn Optimal Node Scheduling Policy

- Reward:

(1) Max. change in LLR of a variable node incident to the currently-scheduled check node $c^*$,

$$r_{c^*} = \max_{v \in \mathcal{N}(c^*)} |L^{\mathrm{new}}_{c^* \to v} - L^{\mathrm{old}}_{c^* \to v}|,$$

where

$$L^{\mathrm{new}}_{c \to v} = \ln \frac{m^{\mathrm{new}}_{c \to v}(1)}{m^{\mathrm{new}}_{c \to v}(0)}, \quad L^{\mathrm{old}}_{c \to v} = \ln \frac{m^{\mathrm{old}}_{c \to v}(1)}{m^{\mathrm{old}}_{c \to v}(0)}.$$

(2) Ratio of the max. change in LLR of a variable node incident to the currently-scheduled check node $c^*$ to the max. change in LLR of any variable node when scheduling any check node $c$,

$$R = \frac{r_{c^*}}{\max_{c \in [m]} r_c}.$$

(3) Minus the Hamming distance between the original signal $\mathrm{x} = [x_1, x_2, \ldots, x_n]$ and the currently-estimated signal $\hat{\mathrm{x}}^{(l)}$,

$$\hat{\mathrm{x}}^{(l)} = \left[ \mathbb{1}_{(L_1^{(l)} > 0)}, \mathbb{1}_{(L_2^{(l)} > 0)}, \ldots, \mathbb{1}_{(L_n^{(l)} > 0)} \right].$$

(4) A linear/non-linear function of (1), (2), or (3) to further encourage higher rewards and/or discourage lower rewards.

# Case Study

Problem parameters:

- # items $n = 100$
- # tests $m = 20$
- # defective items $k = 2$
- Testing matrix $H$: A randomly generated $20 \times 100$ Bernoulli matrix with probability $1 = \ln(2)/2 \approx 0.35$
- Crossover probability of noisy channel $\epsilon = 0.05$
- # BP iterations: flooding $= 10$, node scheduling $= 100$

Benchmarks:

| | Flooding | Maximum-Residual Node Scheduling |
|---|---|---|
| Success Probability (%) (Unknown $k$) | 32.99 | 51.22 |
| Success Probability (%) (Known $k$) | 50.00 | 61.34 |

# Policy Gradient (Advantage Actor-Critic)

State: Sum of the LLRs of the VNs connected to each CN,

$$\left[\sum_{v \in \mathcal{N}(1)} L_v^{(l)}, \sum_{v \in \mathcal{N}(2)} L_v^{(l)}, \ldots, \sum_{v \in \mathcal{N}(m)} L_v^{(l)}\right]$$

where

$$L_v^{(l)} = \begin{cases} \ln \frac{q}{1-q} & l = 0 \\ \ln \frac{q}{1-q} + \sum_{c \in \mathcal{N}(v)} \ln \frac{m_{c \to v}^{(l)}(1)}{m_{c \to v}^{(l)}(0)} & l \geq 1 \end{cases}.$$

Reward: A non-linear function of the ratio $R$,

$$10 \tanh(R - 0.5)$$

where

$$R = \frac{\max_{v \in \mathcal{N}(c^*)} \left| L_{c^* \to v}^{\mathrm{new}} - L_{c^* \to v}^{\mathrm{old}} \right|}{\max_{c \in [m]} \max_{v \in \mathcal{N}(c)} \left| L_{c \to v}^{\mathrm{new}} - L_{c \to v}^{\mathrm{old}} \right|},$$

$$L_{c \to v}^{\mathrm{new}} = \ln \frac{m_{c \to v}^{\mathrm{new}}(1)}{m_{c \to v}^{\mathrm{new}}(0)}, \quad L_{c \to v}^{\mathrm{old}} = \ln \frac{m_{c \to v}^{\mathrm{old}}(1)}{m_{c \to v}^{\mathrm{old}}(0)}.$$

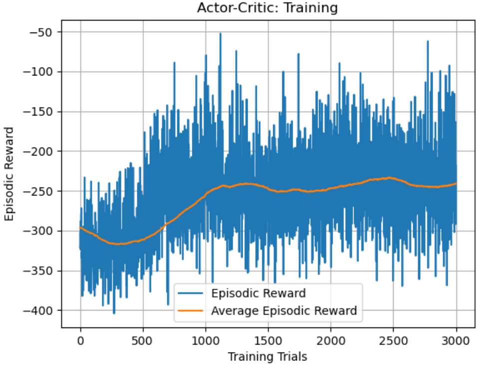
# Results: Policy Gradient (Advantage Actor-Critic)

Learning parameters:

- Learning rate $\alpha = 0.001$
- Discount factor $\gamma = 0.99$

Actor and critic NNs:

- # hidden layers: 2
- # nodes per layer: 60




| | Flooding | Max-Res NS | **RL-based Node Scheduling** | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 500 eps | 1000 eps | 1500 eps | 2000 eps | 2500 eps | 3000 eps |
| Success Probability (%) (Unknown $k$) | 32.99 | 51.22 | **35.4** | **45.5** | **47.1** | **48.3** | **48.9** | **49.2** |
| Success Probability (%) (Known $k$) | 50.00 | 61.34 | **47.2** | **56.4** | **57.6** | **58.1** | **58.5** | **59.0** |

# DQN with Multiple NNs

## (One NN for Each BP Iterations)

**Following the sequential steps as code: Declaring agent**.

Goal: To train $l_{max}$ NNs, each as a nonlinear approx. of the action-value function $Q^{(l)}(\text{state}, \text{action})$ for the BP iteration $1 \leq l \leq l_{\max}$. Individual agent for each iteration

For state (1) and reward (1):

- # hidden layers: 2
- # nodes per layer: $\{32, 64\}$
- Batch size: 64
- Replay buffer size: $1e5$
- Learning rate $\alpha$: $1e-4$
- Discount factor $\gamma$: 0.99
- Decay rate ($\epsilon$-greedy): 0.999
- Iteration per episode: 101

# DQN with Multiple NNs
## (One NN for Each BP Iterations)

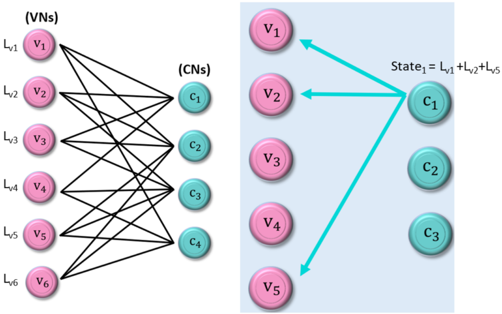
**First observation**

State: The sum of the LLRs of the VNs connected to each CN,

$$\left[\sum_{v\in\mathcal{N}(1)} L_v^{(l)}, \sum_{v\in\mathcal{N}(2)} L_v^{(l)}, \ldots, \sum_{v\in\mathcal{N}(m)} L_v^{(l)}\right],$$

where

$$L_v^{(l)} = \begin{cases} \ln \frac{q}{1-q} & l = 0 \\ \ln \frac{q}{1-q} + \sum_{c\in\mathcal{N}(v)} \ln \frac{m_{c\to v}^{(l)}(1)}{m_{c\to v}^{(l)}(0)} & l \geq 1 \end{cases}.$$

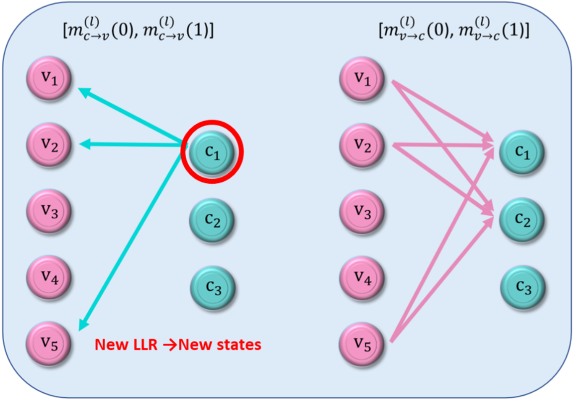


# DQN with Multiple NNs
## (One NN for Each BP Iterations)

**First iteration: Choose action**

$\epsilon$-greedy: For first iteration selecting a random node. Action space is same as state space

**First iteration: New observation based on chosen action**



# DQN with Multiple NNs
## (One NN for Each BP Iterations)

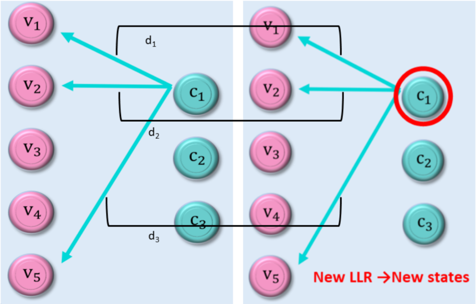
**First iteration: Reward**

Reward: The max. change in LLR of a variable node incident to the currently-scheduled check node $c^*$,

$$\max_{v \in \mathcal{N}(c^*)} \left| L_{c^* \to v}^{\mathrm{new}} - L_{c^* \to v}^{\mathrm{old}} \right|,$$

where

$$L_{c \to v}^{\mathrm{new}} = \ln \frac{m_{c \to v}^{\mathrm{new}}(1)}{m_{c \to v}^{\mathrm{new}}(0)}, \quad L_{c \to v}^{\mathrm{old}} = \ln \frac{m_{c \to v}^{\mathrm{old}}(1)}{m_{c \to v}^{\mathrm{old}}(0)}.$$



# DQN with Multiple NNs
## (One NN for Each BP Iterations)

**First iteration: After reward**

- Store in replay buffer
- Learn: loss, backpropagation, step

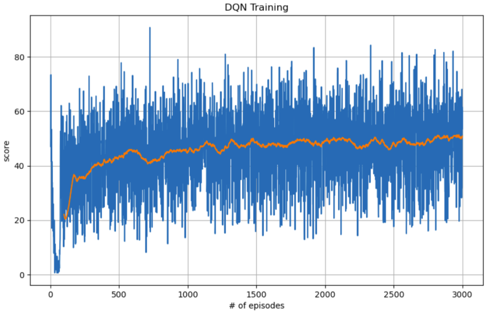



Table: Success probability for different episodes

| Success Probability | 500 | 1500 | 2500 | 3000 |
|---|---|---|---|---|
| % for unknown k | 39.2 | 41.3 | 46.9 | 48.7 |
| % for known k | 50.2 | 53.5 | 58.3 | 60.3 |

- We see a gradual increase in success probability as the number of episodes increases.
- Reward remains almost same, irrespective of episodes (around 55); and a gradual learning can be noticed from episode 0-500.

# DQN with a Single NN
## (One NN for All BP Iterations)

**State:**

(1) The LLRs of the VNs connected to each CN,

$$\left[\{L_v^{(l)}\}_{v\in\mathcal{N}(1)}, \{L_v^{(l)}\}_{v\in\mathcal{N}(2)}, \ldots, \{L_v^{(l)}\}_{v\in\mathcal{N}(m)}\right].$$

(2) The sum of the LLRs of the VNs connected to each CN,

$$\left[\sum_{v\in\mathcal{N}(1)} L_v^{(l)}, \sum_{v\in\mathcal{N}(2)} L_v^{(l)}, \ldots, \sum_{v\in\mathcal{N}(m)} L_v^{(l)}\right].$$

**Reward:**

(1) Minus the Hamming distance between the original signal $\mathrm{x} = [x_1, x_2, \ldots, x_n]$ and the currently-estimated signal $\hat{\mathrm{x}}^{(l)}$,

$$\hat{\mathrm{x}}^{(l)} = \left[\mathbb{1}_{(L_1^{(l)}>0)}, \mathbb{1}_{(L_2^{(l)}>0)}, \ldots, \mathbb{1}_{(L_n^{(l)}>0)}\right].$$

(2) The max. change in LLR of a variable node incident to the currently-scheduled check node $c^*$,

$$\max_{v\in\mathcal{N}(c^*)} |L_{c^*\to v}^{\mathrm{new}} - L_{c^*\to v}^{\mathrm{old}}|.$$

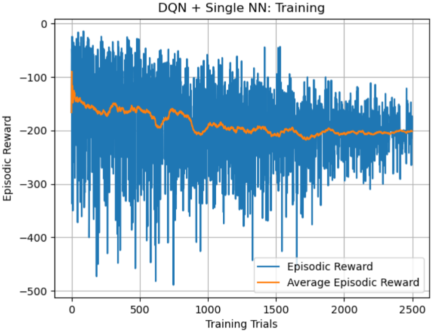
# DQN with a Single NN

## (One NN for All BP Iterations)

Goal: To train a neural network (NN) as a nonlinear approx. of the action-value function $Q(\text{state}, \text{action})$.

For state (1) and reward (1):

- $\#$ hidden layers: $\{1, 2, 3, 4, 5\}$
- $\#$ nodes per layer (3 layers): $\{1000, 500, 100\}$
- Batch size (exp. replay): $\{4, 16, 32\}$
- Update freq. (target net): $\{4, 8, 16\}$
- Learning rate $\alpha$: $\{0.01, 0.005, 0.001, 0.0001\}$
- Discount factor $\gamma$: $\{0.9, 0.95, 0.99\}$
- Decay rate ($\epsilon$-greedy): $\{0.99, 0.999, 0.9999, 0.99999\}$




Training: Not successful!

(Success probability $= 0\%$)

# Summary

**Goal:** To find a node scheduling (NS) policy that maximizes the success probability of the BP recovery algo. for noisy group testing.

Our RL-based NS policies (actor-critic and DQN + multiple NNs):

- outperform the flooding approach in terms of success probability.
- perform closely to the maximum-residual NS policy in terms of success probability, and are less computationally complex.

# Future Work

- Applying variance reducing methods e.g. Averaged-DQN, which is an extension to the DQN based on averaging previously learned Q-values estimates.
- Improving PG by applying a more directed exploration strategy that promotes exploration of under-appreciated rewards.
- Applying other policy gradient algorithms e.g., Natural Policy Gradient (NPG), Proximal Policy Optimization (PPO).
- Implementing DQN + multiple NNs with target network to improve the performance.
- Different variations of the problem:
  - BSC with different cross-over probabilities
  - Z-channel and reverse Z-channel
  - Partial recovery guarantee