INCdeep: Intelligent Network Coding with Deep Reinforcement Learning .*

Qi Wang[1], Jianmin Liu[1,2], Katia Jaffrès-Runser[3], Yongqing Wang[1], Chentao He[1,2], Cunzhuang Liu[1,2] and Yongjun Xu[1]

(1) Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China ;
(2) University of Chinese Academy of Sciences, Beijing, China ;
{wangqi08, liujianmin18z, wangyongqing, hechentao, liucunzhuang20g, xyj}@ict.ac.cn ;

(3) Université de Toulouse, IRIT / Toulouse INP, Toulouse, France
{katia.jaffres-runser}@irit.fr

Abstract

In this paper, we address the problem of building adaptive network coding coefficients under dynamic network conditions (e.g., varying link quality and changing number of relays). In existing linear network coding solutions including deterministic network coding and random linear network coding, coding coefficients are set by a heuristic or randomly chosen from a Galois field with equal probability, which can not adapt to dynamic network conditions with good decoding performance. We propose INCdeep, an adaptive Intelligent Network Coding with Deep Reinforcement Learning. Specifically, we formulate a coding coefficients selection problem where network variations can be automatically and continuously expressed as the state transitions of a Markov decision process (MDP). The key advantage is that INCdeep is able to learn and dynamically adjust the coding coefficients for the source node and each relay node according to ongoing network conditions, instead of randomly. The results show that INCdeep has generalization ability that adapts well in dynamic scenarios where link quality is changing fast, and it converges fast in the training process. Compared with the benchmark coding algorithms, INCdeep shows superior performance, including higher decoding probability and lower coding overhead through simulations and experiments. The average CPU usage and end-to-end delay when deploying INCdeep in practical are almost the same with benchmark algorithms.

1 Introduction

Network coding [1] has been introduced two decades ago to disseminate data over communication networks to achieve higher transmission rates and improve transmission reliability. Intermediate network nodes recombine received packets into one or several output packets before forwarding them to relay or destination nodes. Final destination decodes original packets from a set of received ones to achieve either higher throughput or improved reliability to fading. Linear network coding [2] is a class of network coding, where a network code is obtained by a linear combination using coefficients chosen from a finite field set. Each of the intermediate nodes (hereinafter called relays) can therefore encode the data by linearly combining the received packets, with coefficients selected from the field of operation. Compared to nonlinear network coding [3] that leverages nonlinear combination functions, linear network coding offers a reduced complexity and simpler models, and thus has been deeply studied and widely used.

In linear network coding, the node selects a vector of coefficients as the weight of the original packets in the encoded packet, and then combines the packets using addition and multiplication over a Galois field of size q. In order to recover the full information, a destination (decoder) has to collect at least as many

*This paper will appear in the proceedings of IEEE INFOCOM 2021, main conference.

linearly independent coefficient vectors as the number of packets that were originally mixed in, and then solve the resulting system of linear equations. It means that a destination may be unable to successfully decode the received data packets due to linearly dependent coefficient vectors. The decoding probability depends on the network code design, especially on the selection of coefficients. Over the past two decades, coding coefficients were determined either by dedicated deterministic algorithms [2] or randomly selected in a finite field in random linear network coding [4]. [2] paved the way towards deterministic network coding algorithms by offering a solid theoretical framework. Deterministic network coding [2] promises that the receiver can decode successfully, but it requires global information such as network topology and link capacities. There are many kinds of topologies in reality, as such it is not practical to design specific coding methods for different kinds of network. Moreover, it is not suitable for dynamic networks, as gathering global information from distributed nodes in real-time is complex to apply at a large scale.

Random linear network coding (RLNC) [4] [5] is one simple realization of network coding, where all participating network nodes keep all packets received so far in their buffer, and forward linear combinations of these packets with random coefficients from some infinite field. Due to its main characteristic of randomly choosing the coefficients of linear combinations, RLNC is rateless and suitable for networks with unknown or changing conditions because it can be easily implemented in a distributed fashion. Moreover, it is asymptotically capacity-achieving for networks with packet loss in a wide range of scenarios [6] [7]. [8] showed that the probability of decoding with success (i.e. the probability that the coefficient matrix has full rank) can be high enough as long as codes are chosen in a sufficiently large finite field. However, infinitely large field size are not practical as shown in [9] [10]: they trigger high computational complexity at encoding and decoding stages. Moreover, the failure of decoding may occur not only due to the fact that relay nodes may extract linearly-dependent coefficients, but also because relay nodes do not receive some of the packets to be encoded. In RLNC, coefficients are randomly extracted from a Galois field $\mathbb{F}_q$, with equal probability. Thus, coefficients can not be adjusted according to network dynamics imposed by time-varying link quality and changing number of relays.

This paper aims to investigate a novel network coding design technique that adjusts to network dynamics in real-time. We propose INCdeep, an adaptive, deep reinforcement learning (DRL) approach for network coding. In recent years, DRL has been prevailing in natural language processing problems, robotics, decision games, etc., and achieves superior results, like the Deep Q Learning (DQN) [11] algorithm and AlphaGo [12]. DRL combines deep learning with reinforcement learning (RL) to implement machine learning from perception to action [13]. We believe that DRL is a good fit for our problem. First, based on a Markov decision process (MDP), RL trains an intelligent agent to learn policies directly through interacting with the environment and automatically maximizing the reward. Our DRL based network coding approach learns to optimize coding coefficients for maximizing the decoding probability objective. It receives the rewards by interacting with the network, and is thus fitted to network dynamics. Second, when making coding coefficients decision, we do not know whether that coding decision was a good one (i.e., whether it leads to a full rank matrix) before the destination decodes successfully. RL naturally captures this characteristic as it does not assume that the impact of a given decision on the performance objective is known immediately.

In this work, we consider a single-source, multiple-relays, and single-destination linear and parallel multihop network employing INCdeep at all transmitting nodes including source and relays. In INCdeep, we first setup an MDP model to capture dynamic network state transitions. Then, as [14] demonstrates the superiority of encoding at both source and relay nodes, we derive two different DQN models to optimize coding coefficients for source and relays nodes, respectively. Relaying nodes simply re-encode packets, without resorting to decoding and follow the same DQN model. In INCdeep, we need a link layer feedback (cf. MORE [15]) that acknowledges whether the next hop relay received an encoded packet, as packets may be lost due to varying link quality. In this way, dynamic network variations can be automatically and continually expressed as MDP state transitions. To increase the probability of successful decoding at the destination node, we devise a DQN to automatically select coding coefficients to adapt to network dynamics. In our INCdeep, the state space for both source and relay DQNs are associated to coded packets in the buffer of the node of interest and the ones in the next hop buffer (which are impacted by link quality). The frequency of state transition is positively correlated with the number of packets to be encoded. INCdeep can be generalized to networks with different link channel models as the coding coefficient evaluation method is independent from the link packet error rate distribution. Moreover, it scales constantly with the number of relays as a unique DQN model is used for the coefficient optimization of all relays. This feature is very important to adjust

to network dynamics. As such, INCdeep DQN models trained under given network settings can be directly leveraged for different network settings. In summary, the DQN based INCdeep can automatically provide adaptive coding coefficients with high probability of successful decoding at the destination node. The main contributions are summarized as follows: Through extensive simulations, we demonstrate that INCdeep not only handles dynamic network variations well, but also achieves superior performance as compared with the classical benchmark coding algorithms (fountain code at source and RLNC at relays), in which the relaying nodes simply recode packets, without resorting to decoding.

- In order to adapt to network dynamics, we propose INCdeep, an adaptive, deep reinforcement learning (DRL) approach to automatically build coding coefficients. Specifically, INCdeep learns to optimize coding coefficients to maximize a decoding probability objective, through the rewards obtained by interacting directly with the network, and thus fitting to network dynamics.
- We setup an MDP model to formulate the coding coefficients selection problem, where network variations can be automatically and continually expressed as MDP state transitions. INCdeep has good generalization ability on networks with different link qualities and with different number of relays, which is an important feature for offering a practical deployment.
- We have performed simulations and testbed experiments to evaluate INCdeep. The extensive results verify the generalization ability of INCdeep that adapts well in dynamic scenarios and also demonstrate that INCdeep converges fast. Compared with benchmark algorithms, INCdeep has higher decoding probability and lower coding overhead through simulations and experiments.

The rest of paper is organized as follows. Section 2 reviews the related research efforts. Section 3 presents the system model. Section 4 introduces INCdeep. Section 5 and section 6 validate INCdeep. Finally, Section 7 concludes the paper.

2 Related work and Motivation

2.1 State-of-the-art Network Coding

Traditional coding techniques consist of two parts: source coding and channel coding. Source coding allows source nodes to encode packets to seamlessly adapt to the offered rate of the transmission and thus improve reliability. Traditional solutions are random linear fountain codes [16] or Luby-transform codes [17]. Channel coding converts noisy channel into a noiseless one by introducing redundant bits in the information sequence.

In 2000, Ahlswede [1] proposed network coding which is regarded as the commencement of network coding theory, whose main benefit is to increase the capacity of a network over the practical limit achieved by regular routing. Additionally, network coding is shown to increase as well the robustness of the transmission. [2] proposed linear network coding and proved it can achieve max-flow bound on the information transmission rate by applying a linear code multicast, a network code obtained by linearly combining information using coefficients chosen from a finite field. [2] paved the way towards deterministic network coding algorithms by offering a solid theoretical framework. Authors of [18] were the first to propose deterministic network coding. The authors in [18] presented an algebraic framework for investigating capacity issues in networks using linear network. [19] considers a multicast configuration with two sources. Deterministic network coding's main characteristic is that coding vector is determined by an algorithm. Deterministic network coding promises that the receiver can decode successfully, but it requires global network information, which makes it incapable of adapting to varying network conditions. Moreover, it has other shortcomings such as high complexity and low robustness.

[18]'s work prepared the fertile ground for the formulation of random linear network coding. Ho et al. proposed random network coding in [20] which makes network coding adaptive to network dynamics. In [8], the same authors model a network as a delay-free acyclic graph with unit capacity directed links and one or more discrete sources and gives a lower bound on the success probability of random network coding for multicast connections. The probability of decoding successfully can be high enough as long as codes are in a sufficiently large finite field. Random linear network coding is suitable for networks with unknown or changing conditions because it can be implemented in a distributed way.

However, the failure of decoding may occur not only due to the fact that relay nodes may extract linearly-dependent coefficients, but also because relay nodes do not receive some of the packets to be encoded due to network dynamics with time-varying link quality. In existing linear network coding including deterministic network coding and random linear network coding, coding coefficients are either determined by an algorithm or randomly selected from Galois field with equal probability. The coding coefficients can not adapt to the dynamic network condition with good decoding performance.

2.2 Adopting a DQN-based DRL Approach

Reinforcement learning (RL) [21] is a learning process in which an agent can periodically make decisions, observe the results, and then automatically adjust its strategy to achieve the optimal policy. Recently, [22]'s work proposed RL-aided dynamic sparse network coding (RL-aided SNC), a reinforcement learning framework for dynamically adjusting coding parameters to improve the performance. They adopt Q-learning to decide the optimal degree (i.e., the number of intermediate packets, which are randomly drawn from a set of pre-coded packets) on the batch transmission size. They neither decide on the optimal coding coefficients, nor consider improving decoding probability. Moreover, reinforcement learning using tabular methods is unsuitable and inapplicable when the state space is large. Recently, deep learning [23] has been introduced as a new breakthrough technique. It can overcome the limitations of reinforcement learning, and thus open a new era for the development of reinforcement learning, namely Deep Reinforcement Learning (DRL). DRL has been adopted in numerous applications of reinforcement learning in practice such as robotics, computer vision, etc [23].

In the areas of communications and networking, DRL has been recently used as an emerging tool to effectively address various problems and challenges [24–26]. We believe that DRL is a good fit for our problem. First, with DRL, INCdeep learns to optimize coding coefficients for maximum decoding probability objective, by the rewards obtained through interaction with network, and thus fit for the network dynamics. When the adapted coding coefficients are found, learning-based solutions often outperform random selected coding coefficients. Second, when making coding coefficients decision, we do not know whether that coding decision was a good one (i.e., whether it leads to a full rank matrix) before the destination decodes successfully. RL naturally captures this characteristic as it does not assume that the impact of a given decision on the performance objective is known immediately.

3 System model

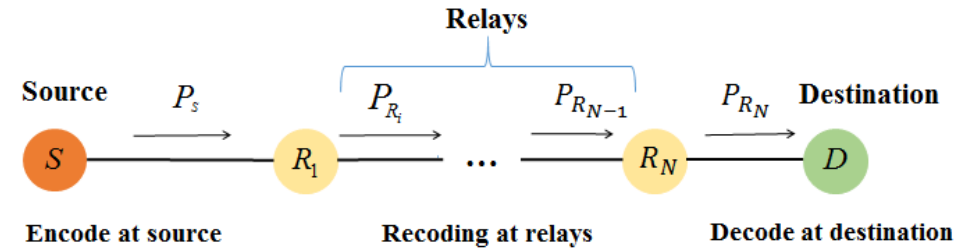
In this work, we investigate two different atomic topologies that are at the core of many deployment scenarios of wireless multi-hop communications. We will concentrate first on a linear deployment scenario as shown in Fig.1(a). In this example, source S aims to transmit a message of K equally sized packets to the destination D using a daisy chain of relays where R_i can only communicate with relay R_{i+1} $(i = 1, ...N-1)$ and D can only receive messages from R_N. The second parallel topology has been investigated as shown in Fig.1(b), where source S delivers a message of K equally sized packets to the destination D with the help of multiple parallel relays. In this work, we enable encoding at the source node S to further mitigate packet losses. Source S continuously sends coded packets until the destination D can successfully recover the original message. All relay nodes simply re-encode packets, without resorting to decoding. A perfect link layer feedback that acknowledges whether the next hop relay received an encoded packet is needed.

3.1 Benchmark coding algorithms

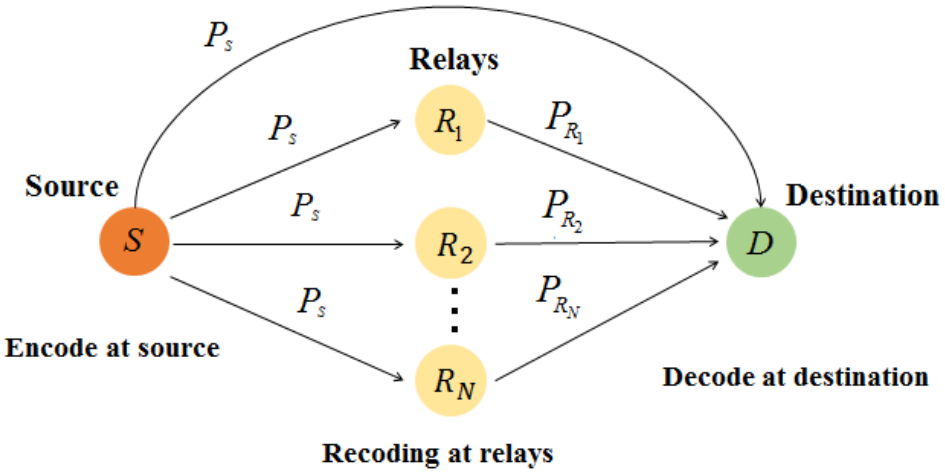
INCdeep is compared to benchmark algorithms including the traditional random linear network coding and state-of-the-art reinforcement learning-aided sparse network coding [22].

3.1.1 Traditional random network coding

Our benchmark implementation considers that the source nodes uses a fountain code and relays combine coded packets using RLNC.



(a) Linear topology



(b) Parallel topology

Figure 1: Investigated multihop network topologies composed of N relay nodes: (a) Linear topology; (b) Parallel topology.

Fountain code at the source. Fountain codes [16] are rateless erasure codes in the sense that a potentially limitless sequence of encoded packets can be generated from the source message. The coding process ends as soon as the destination has received enough packets. We consider random linear fountain codes in this work where a message from source is first partitioned into K fragments (packets) $x_1, x_2, ..., x_K$ with equal length when being sent to destination. For each fragments x_j, a coding coefficient $g_S^j \in \mathbb{F}_q$ is picked independently with uniform probability. The source as the encoder generates the output encoded packet $P_S = \sum_{j=1}^{K} g_S^j x_j$, where all operations are performed in Galois field $\mathbb{F}_q$. The sum can be done by successively exclusive-or-ing (XOR) the packets together. Then the source generates a stream of encoded packet P_S until the destination can successfully decode the original message. Coding coefficients are encoded in packet header.

RLNC at relay nodes. As mentioned above, a source S transmits a random linear encoded flow to a destination D through the multi-hop network as shown in Fig. 1. RLNC [5] is employed at relays, which is traditionally used in multihop networks due to its simplicity and scalability [27]. When a relay node receives an encoded packet P from previous node, P is XORed with randomly selected buffered packets in this relay. Before transmission to the next hop node, the relay generates recoded coefficients in $\mathbb{F}_q$ and applies it to its selected buffered packets. The new recoded coefficients are transmitted in the recoded packet headers. RLNC is in line with sending RL encoded packets at the source. The buffer size M of RLNC is the same as the one considered in INCdeep.

3.1.2 RL-aided SNC

The recently proposed reinforcement learning-aided sparse network coding (RL-aided SNC) [22] is the other benchmark algorithm in this paper. In the RL-aided SNC, the packet transmission operates batch-by-batch. For every batch, the node takes the action, which consists of two parts: degree and batch transmission size. After a batch transmission completes, source node will get the message from destination node that the rank of coefficient matrix, then Q-learning is adopted to get the optimal action.

3.2 Decoding at the destination

INCdeep and benchmark coding algorithms use the same Maximum Likelihood decoding (ML-decoding) method. When the destination receives at least K encoded packets, the decoding process is started. Encoded packets are mapped to a linear system of equations, that can be solved with Gaussian elimination if the coefficient matrix is full rank. The ML-decoding provides small error probability of decoding, however its complexity can grow rapidly with both N_r (number of received packets) and K, in the order of $O(N_r K)$ [28]. K is the dimension K of the RL code.

4 Network coding with deep Reinforcement Learning

This section introduces our proposition of network coding design using a DRL whose objective is to maximize decoding probability. We formalize the network coding problem into a learning task, and propose INCdeep, a deep reinforcement learning (DRL)-based intelligent network coding method.

4.1 Problem definition

Since the coding strategy of source S is different from the coding strategy of relay nodes, we design two DQN models DQN_S and DQN_R. DQN_S is used to optimize the coding strategy of source S, and DQN_R is used to optimize the coding strategy of relay nodes.

Except for the source node, each relay R_i and destination D have a buffer $B_{latest}(R_i)$ (or $B_{latest}(D)$), which is used to store the latest M coded packets that relay R_i (or destination D) has received from its previous node. Meanwhile, except for destination D, source S and each relay R_i have a buffer $B_{all}(S)$ (or $B_{all}(R_i)$), which stores all packets received by its next hop node. We use ACK to confirm the transmitted encoded packet is received by the next hop node. If source S (or relay R_i) receives an ACK for a previously transmitted packet, source S (or relay R_i) stores the transmitted packet into its $B_{all}(S)$ (or $B_{all}(R_i)$), otherwise, the transmitted packet is not stored in $B_{all}(S)$ (or $B_{all}(R_i)$).

Suppose there are N relays in the multi-hop network, we use a set $C_{R_i}(i = 1, 2, ..., N)$ to represent the M encoded packets in buffer $B_{latest}(R_i)$ of R_i and a set C_D to represent the M encoded packets in buffer $B_{latest}(D)$ of D. For R_1, set $C_{R_1} = \{P_S(1), ..., P_S(j), ..., P_S(M)\}$ is defined, where packet $P_S(j)$ is the j-th encoded packet received from source S in buffer $B_{latest}(R_1)$ of R_1. For relay $R_i(i = 2, 3, ..., N)$, set C_{R_i} is defined as $C_{R_i} = \{P_{R_{i-1}}(1), ..., P_{R_{i-1}}(j), ..., P_{R_{i-1}}(M)\}$, where packet $P_{R_{i-1}}(j)$ is the j-th encoded packet received from previous relay R_{i-1} in buffer $B_{lastet}(R_i)$ of R_i.

For destination D, we set $C_D = \{P_{R_N}(1), ..., P_{R_N}(j), ..., P_{R_N}(M)\}$, where packet $P_{R_N}(j)$ is the j-th encoded packet received from relay R_N in buffer $B_{latest}(D)$.

4.2 An adaptive Intelligent Network Coding with Deep Reinforcement Learning

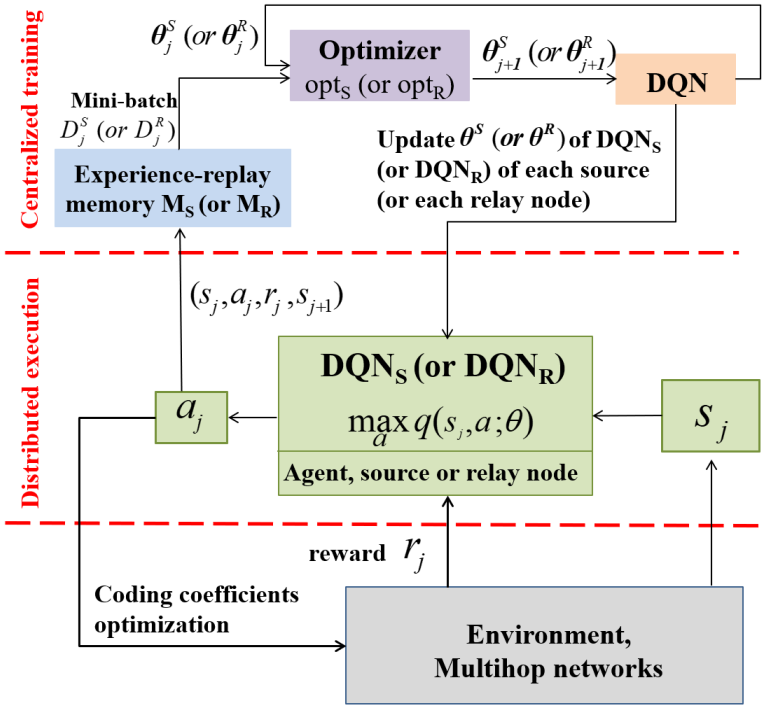


Figure 2: Architecture of INCdeep

To improve decoding probability of the destination, the encoded packet sent by the node (source or relay) improves the decoding probability of the next hop node. Specifically, each node adaptively selects the coding coefficients based on the coding coefficients of the latest M coded packets received by the next hop node instead of randomly selecting a value from Galois field of size with equal probability. The discrete state is associated to the latest M coded packets received by the next hop node. Suppose a message from source is partitioned into K fragments. A coded packet will have K coding coefficients, so the size of state space is about M x K. As M or K becomes larger, the state space increases. However, the general RL approaches are not capable of dealing with large state space due to maintaining a look-up table to store policies. To solve this problem, DRL approaches emerge, e.g., deep Q-learning, which uses a deep neural network (DQN) as an approximate Q-function in lieu of a lookup table. Therefore, we propose the INCdeep, an intelligent network coding method by using DQN to optimize the coding coefficients.

In INCdeep, we estimate the coding coefficients of the packets before sending an encoded packet, and formulate the estimation process with an MDP. The estimation of the coding coefficients of one packet is executed at one discrete decision step. Fig.2 illustrates the architecture of INCdeep. INCdeep is implemented by an alternative approach where coding coefficients optimization of packets is distributively executed at source and each relay node, while training of DQN is centralized to simplify implementation and accelerate

training. Each node (source or relay node) is considered as INCdeep agent. Source and each relay node have the same copy of DQN_S and DQN_R, respectively. The environment is the network including nodes, links in the network topology and encoded packets in the buffer of each relay node. In the execution procedure, at each decision step j, source (or relay node) takes an action a_j under the state s_j to optimize the coding coefficient of the j-th packet of a message (or the j-th encoded packet in the buffer of the relay) by DQN_S (or DQN_R). After a_j is taken, the state transforms from s_j to s_{j+1}, and the node (source or relay) gets a reward r_j from the environment. The node (source or relay) then stores experience (s_j, a_j, r_j, s_{j+1}) to replay memory M_S (or M_R). In the training procedure, the centralized optimizer opt_S (or opt_R) randomly samples mini-batch D^S (or D^R) from the experience-reply memory M_S (or M_R) to update parameters θ^S (or θ^R) of DQN_S (or DQN_R) by minimizing the loss. After the parameter θ^S (or θ^R) is updated, the centralized optimizer opt_S (or opt_R) sends updated parameters θ^S_{j+1} (or θ^R_{j+1}) to source (or each relay node). When receiving updated parameters, source (or each relay node) updates parameters of its DQN_S (or DQN_R).

4.2.1 DQN model for source

When sending an encoded packet P_S, a source S firstly determines a coding coefficient vector $G_S = [g_S^1, ..., g_S^j, ..., g_S^K]$ for K packets $X = [x_1, ..., x_j, ..., x_K]$ of a message, where $P_S = G_S \cdot X$. We consider a decision process for generating coding coefficients of K packets through a Markov decision process. Fig. 3 shows the process of optimizing G_S by our coding model DQN_S. The state, action and reward are depicted as follows:

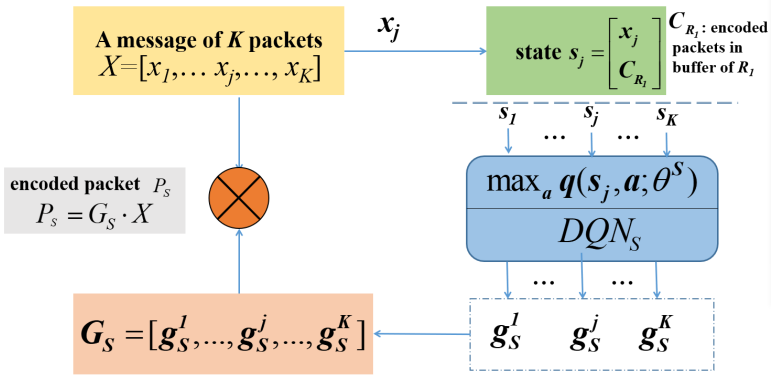


Figure 3: Coding coefficients optimization of source by DQN_S

(1) State, s: At each decision step j ($j = 1, 2, ..., K$), the state of the source S consists of two components, including the information of the j-th packet x_j of a message and M encoded packets in buffer $B_{latest}(R_1)$ of the first relay R_1. Specially, the state of source is represented as $s_j = [x_j, C_{R_1}]$.

(2) Action, a: The action space of source S is given by $A_S = \{0, 1, ..., (q-1)\}$, where q is finite field size $\mathbb{F}_q$. At each decision step j, source S determines coding coefficient g_S^j of the j-th packet x_j according to action $a_j \in A_S$, i.e., $g_S^j = a_j$.

(3) Reward, r: After sending an encoded packet P_S, source evaluates the optimized coding coefficient vector G_S. For the network coding problem considered in this paper, our goal is to maximize decoding probability. According to ML-decoding principle, when the rank of the linear system of equations, which is formed by the packets received at the destination, is equal to the dimension of the RL code K, destination D can recover original information successfully. To achieve our goal, the reward is designed as the increment of the rank offered by the packets that have reached the buffer of the next hop node.

In this paper, we consider the multi-hop network with lossless links. In the case of lossless links, packet error rate $PER = 0$, i.e., the packets can reach the next hop node with a probability of 1, while PER of channel is more than 0 over lossy links.

The encoded packet might be dropped over lossy links, thus source S may not receive an ACK packet for packet P_S. In this case, coding coefficients cannot be evaluated by adding P_S to buffer $B_{all}(S)$, so we introduce a new buffer B_{new} to evaluate the case. For example, suppose source S has sent two packets ($P_S(1)$ and $P_S(2)$) and the two packets are received by the first relay R_1, i.e., $B_{all}(S) = \{P_S(1), P_S(2)\}$. Source now sends the third packet $P_S(3)$, but R_1 does not receive $P_S(3)$. We copy all packets ($P_S(1)$,$P_S(2)$) in buffer $B_{all}(S)$ to a new buffer B_{new}, and add $P_S(3)$ to B_{new} to evaluate the coefficient of $P_S(3)$, instead of $B_{all}(S)$. After adding $P_S(3)$ to B_{new}, if the rank of equations formed by packets ($P_S(1)$,$P_S(2)$,$P_S(3)$) in B_{new} increases, the reward r_j for optimized coding coefficient g_S^j is +1, otherwise, the reward is -1. If source S receives an ACK packet for packet P_S, as with the case of lossless links, the reward is calculated by adding packet P_S to its buffer $B_{all}(S)$.

4.2.2 DQN model for relays

When relay R_i receives an encoded packet from the previous node, it starts to re-encode the packet. For re-encoding, relay R_i generates a coefficient vector G_{R_i} for M packets of its buffer $B_{latest}(R_i)$. The decision process of coding coefficients for M packets is a Markov decision process. At each decision step $j (j = 1, 2, ..., M)$, relay R_i optimizes coefficient $g_{R_i}^j$ relative to the j-th packet in its buffer, and produces a new encoded packet $P_{new}^j(R_i)$. After M steps, relay R_i transmits a new packet $P_{R_i} = P_{new}^{M+1}(R_i)$, as shown in Fig. 4.

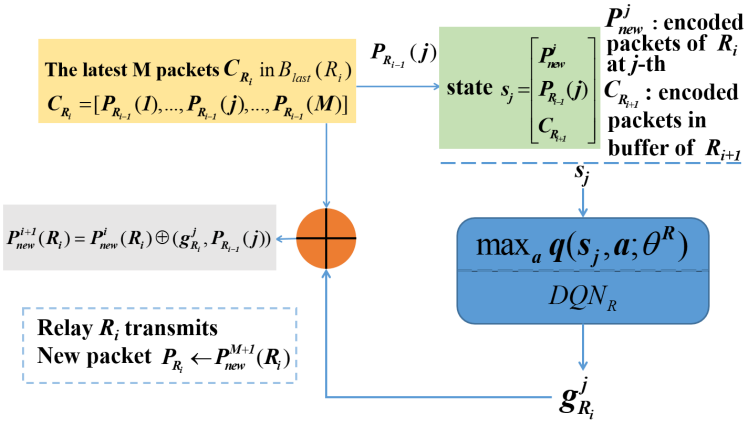


Figure 4: Coding coefficients optimization of relays by DQN_R

(1) State, s: At each decision step j ($j = 1, 2, ..., M$), the state of the first relay R_1 consists of three components, the new encoded packet $P_{new}^j(R_1)$ produced by R_1 at step j, the j-th packet $P_S(j)$ in its buffer $B_{latest}(R_1)$, and M encoded packets C_{R_2} in buffer $B_{latest}(R_2)$ of its next relay R_2. In summary, the state of the first relay R_1 is represented as $s_j = [P_{new}^j(R_1), P_S(j), C_{R_2}]$.

At each decision step j, the state of relay $R_i (i = 2, ..., N-1)$ consists of three components, the new encoded packet $P_{new}^j(R_i)$ produced by R_i at step j, the j-th packet $P_{R_{i-1}}(j)$ (from R_{i-1}) in R_i's buffer $B_{latest}(R_i)$, and M coded packets $C_{R_{i+1}}$ in buffer $B_{latest}(R_{i+1})$ of its next relay R_{i+1}. In summary, the state of relay R_i is represented as $s_j = [P_{new}^j(R_i), P_{R_{i-1}}(j), C_{R_{i+1}}]$.

For the last relay R_N, at each decision step j, its state consists of three components, the new encoded packet $P_{new}^j(R_N)$ produced by R_N at step j, the j-th packet $P_{R_{N-1}}(j)$ in its buffer $B_{latest}(R_N)$, and M encoded packets C_D in buffer $B_{latest}(D)$ of destination D. In summary, the state of the last relay R_N is represented as $s_j = [P_{new}^j(R_N), P_{R_{N-1}}(j), C_D]$.

At the beginning of coding coefficient generation, the new coded packet $P_{new}^1(R_i)$ is the latest encoded packet that relay R_i receives from the previous relay R_{i-1} (or source S). When generating a coding coefficient for the j-th packet $P_{R_{i-1}}(j)$ in its buffer $B_{latest}(R_i)$, the new encoded packet $P_{new}^{j+1}(R_i)$ at

step $j+1$ can be determined by an XOR operation on packet $P^j_{new}(R_i)$ and packet $P_{R_{i-1}}(j)$, i.e., $P^{j+1}_{new}(R_i) = P^j_{new}(R_i) \bigoplus (g^j_{R_i} * P_{R_{i-1}}(j))$.

(2) Action, a: At each step j, relay R_i determines the coding coefficient $g^j_{R_i}$ of the j-th packet in its buffer $B_{latest}(R_i)$ according to an action $a_j \in A_R$, where $A_R = \{0, 1, ..., (q-1)\}$, and $g^j_{R_i} = a_j$.

(3) Reward, r: After transmitting packet P_{R_i}, relay R_i evaluates optimized coding coefficient vector G_{R_i}. Similar to the reward calculation on source, after adding the packet P_{R_i} to the buffer $B_{all}(R_i)$ (or B_{new}) in the case of lossless links (or lossy links), if the rank of linear system of equations formed by these packets in buffer $B_{all}(R_i)$ (or B_{new}) increases, reward r_j for each optimized coding coefficient $g^j_{R_i}$ is +1, otherwise, reward is -1.

The implementation of INCdeep is shown in Algorithm 1. First, source obtains the coefficient vector by Algorithm 2. Then, source generates an encoded packet by multiplying the coefficient vector and the message vector, and sends the encoded packet to its next hop. When a relay R_i receives an encoded packet from its last hop, it sequentially calculates the coefficient of the encoded packet in its buffer by Algorithm 3. Then relay R_i recodes the message by M XOR operations on the encoded packet from its last hop and all encoded packet in its buffer, and transmits the recoded packet. Once the destination can successfully recover the message, source stops sending the encoded packet.

Algorithm 1 INCdeep: intelligent network coding with DRL

while Destination can not recover message X **do**
  Source obtains coefficient vector G_S from Algorithm 2.
  Source sends an encoded packet $P_S = G_S \cdot X$
  for each relay R_i **do**
    if R_i receives an encoded packet P from the last node **then**
      for $j = 1, 2, ..., M$ **do**
        R_i obtains coefficient $g^j_{R_i}$ of j-th packet $P_{R_{i-1}}(j)$ of its buffer from Algorithm 3
        $P \leftarrow P \bigoplus (g^j_{R_i} * P_{R_{i-1}}(j))$.
      end for
      R_i transmits that recoded packet P
    end if
  end for
end while

Algorithm 2 Calculate coefficients for source

Input: Message $X = [x_1, ..., x_j, ..., x_K]$.
Output: Coding coefficient vector G_S.
for j=1,2,...,K **do**
  State of source $s_j = [x_j, C_{R_1}]$
  Calculate Q-values by CNN of DQN_S with s_j as the input
  choose an action a_j by ε-greedy algorithm
  coefficient $g^j_S \leftarrow a_j$
end for
Return $G_S = [g^1_S, ..., g^j_S, ..., g^K_S]$

4.3 Training Methodology

For better modeling the matrix information in state, we adopt convolutional neural network (CNN) to extract key features. In our model, the first layer of CNN is fed by the state vector. Each neuron of the output layer gives an estimate of the Q-function for a given state input and corresponding action output. The total number of output neurons is equal to the size of the action space. Since the state of relays consists of M packets in the buffer of its next node and extra two packets (a current recoded packet and a packet

Algorithm 3 Calculate coefficients for relay

Input: Encoded packet $P^j_{new}(R_i)$.
Output: Coding coefficient $g^j_{R_i}$.
if Relay is the first relay **then**
  State $s_j = [P^j_{new}(R_i), P_S(j), C_{R_2}]$
else if Relay is the last relay **then**
  $s_j = [P^j_{new}(R_i), P_{R_{N-1}}(j), C_D]$
else
  $s_j = [P^j_{new}(R_i), P_{R_{i-1}}(j), C_{R_{i+1}}]$
end if
Calculate Q-values by CNN of DQN_R with state s_j as the input
choose an action a_j by ε-greedy algorithm
coefficient $g^j_{R_i} \leftarrow a_j$
Return $g^j_{R_i}$

considered to be XORed in its buffer), the size M of buffer is set to $K-2$ to build a square matrix of the state as the input of CNN.

5 SIMULATION RESULTS

5.1 Simulation Setup

5.1.1 Network topology

We consider two different atomic topologies of multihop communications as shown in Fig. 1. For any link within a considered topology, an erasure channel model is characterized by its Packet Error Rate (PER).

5.1.2 Simulation platform

We use a Python 3.5-based framework, TensorFlow 1.15, to construct the architecture of INCdeep and its deep neural network. All results are conducted on Ubuntu 16.04 server with 2 RTX2080 Ti GPUs and Intel 2 Xeon E5-2660. We trained our model for 10000 episodes. After training, we tested it for a period of $T = 10$ episodes.

5.1.3 Results assessment

To validate the generalization ability and measure the closeness of decoding probability by simulation and experiment, the root mean squared error (RMSE) is calculated. $RMSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\frac{(d(i)-\tilde{d}(i))^2}{d(i)^2}}$, with n the number results to be compared, $d(i)$ the decoding probability derived by training directly (or simulation), and $\tilde{d}(i)$ the decoding probability derived by generalization ability (or experiment).

The overhead measures the coding efficiency representing the percentage of overhead packets required to decode the original message. It is derived as $overhead = \frac{1}{K}(E) * 100 = \frac{1}{K}(Nr - K) * 100$ where K is the dimension of the network coding, E is the number of excess packets when using network coding and N_r is the number of packets received at destination.

5.2 Performance Evaluation Results

5.2.1 Training efficiency

The convergence of INCdeep is evaluated in different combinations of finite field size q and PER. To achieve better training effect and accelerate the training speed, we adjust parameters including the number of neurons of CNN, $\epsilon = 0.1$ for $\epsilon-$greedy algorithm and learning rate $\lambda = 0.001$. As a result, the reward converges in about 10000 episodes for field size $q = 8$ and $PER = 0.1$, and 7000 episodes for finite field size $q = 8$ and $PER = 0$ as well as finite field size $q = 2$. Consequently, INCdeep can converge fast in the training process.

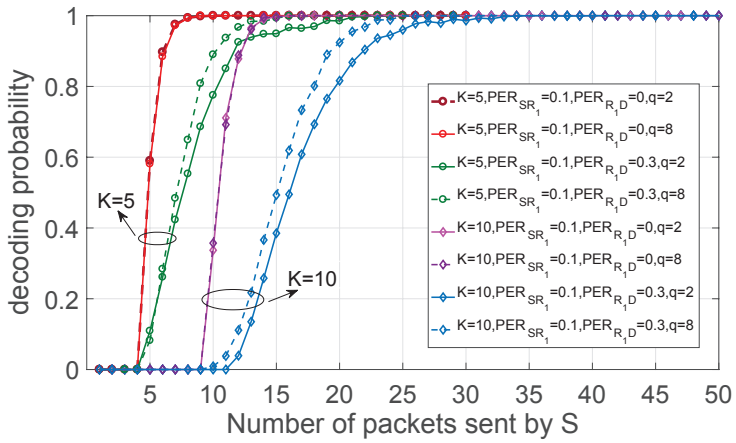
5.2.2 Runtime cost

In the process of network coding, source and relay needs to run the trained model to obtain coding coefficients, leading to extra runtime cost with INCdeep. However the runtime cost is very small, and the runtime is 0.8 ms and 1.6 ms to obtain coding coefficients under $K = 5$ and $K = 10$, respectively.

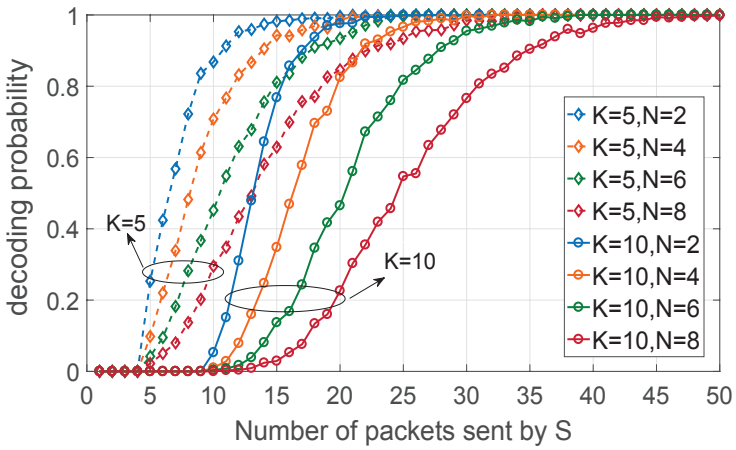
5.2.3 Performance of INCdeep with different PER and different number of relays N

We investigate the performance of INCdeep on the two atomic linear and parallel topologies of Fig. 1. A general multihop network can be decomposed to line up or parallel such paths. We start with a single-relay linear network. Fig. 5(a) illustrates the solution of INCdeep on the decoding probability as a function of the number of transmissions for various combinations of source message size K, finite field size q and PER for each link. It can be observed that decoding probability increases as q increases for the same K and the same PER. For the same finite field size q, as the PER increases, decoding probability decreases much for the same K. We also study the impact of relay number on decoding probability with different K on multihop lossy link as shown in Fig. 5(b). For the same K, decoding probability decreases much as the number of relays increases. Increasing the number of relays with lossy links reduces the number of successful arrivals in D, decreasing the decoding probability.

In the parallel network, we first consider the two-relay parallel network and also study the decoding probability for various combinations of K, finite field size q and PER for each link, as illustrated in Fig. 6(a). Similarly, the decoding probability increases as q increases. For the same finite field size q, as the PER increases, decoding probability decreases much. Fig. 6(b) illustrates the decoding probability for different number of relays with different K. It is seen that as the number of relays increases, the decoding probability increases for the same K, which is contrary to the linear network. The reason is that packets from S to D travel over two hops maximum. Increasing the number of relays increases the number of successful arrivals in D, increasing the decoding probability.

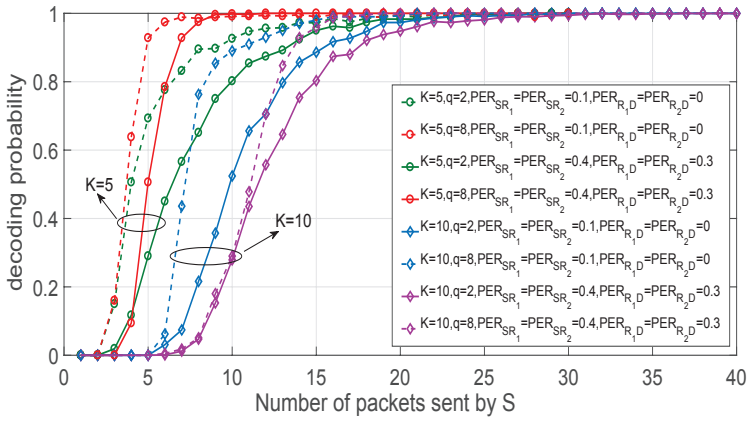


(a) Single-relay linear network, $K \in \{5,10\}$, $q \in \{2,8\}$, PER_{SR_1}=0.1 and $PER_{R_1\mathrm{D}} \in \{0,0.3\}$.

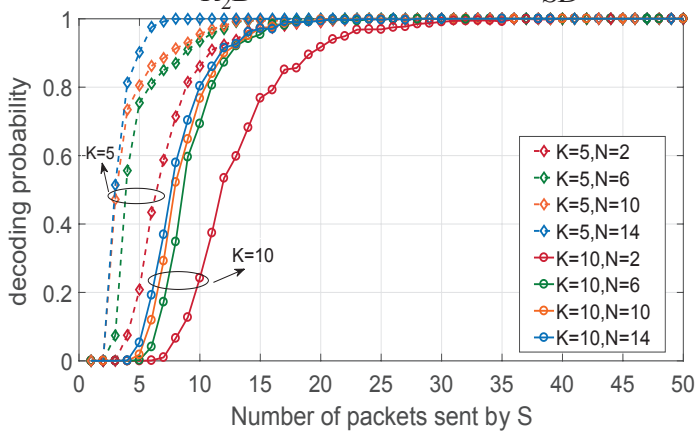


(b) Multiple-relays linear network, $K \in \{5,10\}$, various $N \in \{2,4,6,8\}$, q=2 and $PER_{SR_j=R_jD}$=0.1 (j=1,…,N).

Figure 5: Decoding prob. with different PER and relay number N for linear network.

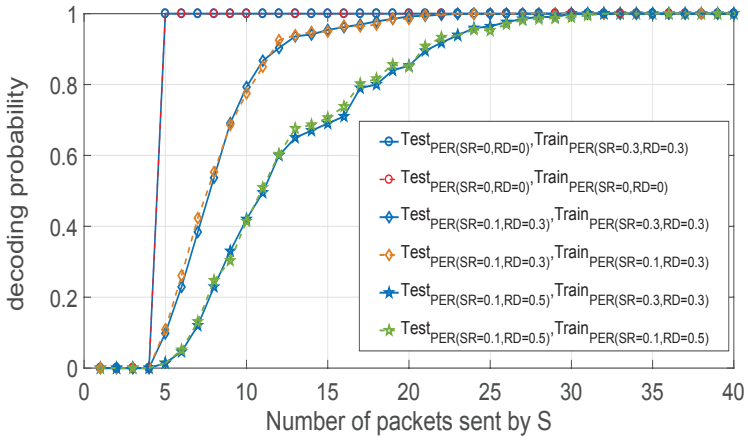


(a) Two-relay parallel network, $K \in \{5,10\}$, $q \in \{2,8\}$, $PER_{SR_1} \in \{0,0.1\}$, $PER_{SR_2} \in \{0.1,0.2\}$, $PER_{R_1\mathrm{D}} \in \{0,0.2\}$, $PER_{R_2\mathrm{D}} \in \{0.1,0.3\}$ and PER_{SD}=0.5.

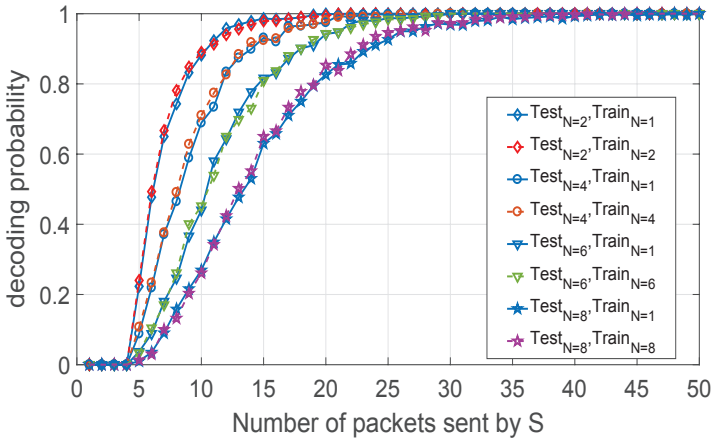


(b) Multiple-relays parallel network, $K \in \{5,10\}$, various $N \in \{2,6,10,14\}$, q=2, PER_{SR_j}=0.1, PER_{R_jD}=0.3 (j=1,…,N), PER_{SD}=0.8.

Figure 6: Decoding prob. with different PER and relay number N for parallel network.



(a) Generalization ability on *PER*.



(b) Generalization ability on relay number N.

Figure 7: Generalization ability on PER and relay number N.

5.2.4 Generalization ability

We verify the generalization of proposed INCdeep on link quality and number of relays in Fig. 7. The generalization ability indicates whether INCdeep can be generalized to the networks with different link qualities and with different number of relays. This feature is very important to adapt to network dynamics. As such, INCdeep DQN models trained under given network settings can be directly leveraged for different network settings.

To verify the generalization ability for different PER, we trained INCdeep (including DQN_S and DQN_R model) with a specific setting $PER_{SR} = 0.3$ and $PER_{RD} = 0.3$, presented as $Train_{PER_{SR}=0.3,PER_{RD}=0.3}$. We tested these trained well models for other different PER_{SR} and PER_{RD} to validate the generalization ability, presented as $Test_{PER_{SR=i,RD=j}}$, $i \in \{0, 0, 1\}$ and $j \in \{0, 0.3, 0.5\}$. As shown in Fig. 7(a), the results obtained by generalization ability match well with the one trained directly. For instance, the decoding probability when $PER_{SR=0,RD=0}$ obtained by trained well model under $Train_{PER_{SR}=0.3,PER_{RD}=0.3}$ matches well with the same $PER_{SR=0,RD=0}$ in training and testing. The RMSE of decoding probability for $(PER_{SR} = 0, PER_{RD} = 0), (PER_{SR} = 0.1, PER_{RD} = 0.3), (PER_{SR} = 0.1, PER_{RD} = 0.5)$ are 0, 0.002 and 0.003 respectively. The really small RMSE indicates that quasi-perfect generalization ability of INCdeep on link quality. This generalization ability is because the coding coefficient evaluation method is independent from the link PER distribution.

Similarly, to verify the generalization ability on number of relays N, we trained INCdeep for a single-relay network, $Train_{N=1}$. We adopt this trained INCdeep $Train_{N=1}$ for other different relay number, presented as $Test_N$, $N \in \{2, 4, 6, 8\}$. In Fig. 7(b), the results obtained by generalization ability match well with the one trained directly. The RMSE of decoding probability for $N = 2, 4, 6, 8$ are 0.0034, 0.0072, 0.011 and 0.015 respectively, which indicates good generalization ability for different N. It is because a unique DQN model is used for the coefficient optimization of all relays.

5.2.5 Comparison between INCdeep and benchmark coding algorithms

Fig. 8 and Fig. 9 illustrate the comparison between the decoding probability obtained by INCdeep, fountain code at source and RLNC at relays, and RL-aided SNC. For linear topology network with PER=0.1 on each link, INCdeep outperforms the benchmark coding algorithms for different relay number N=2,6,10 as in Fig. 8. For parallel topology network with PER_{SR} =0.1, PER_{RD} =0.3 and PER_{SD} =0.8, INCdeep performs better than benchmark coding algorithms as increasing of relay number, seen in Fig. 9. For both of topology, INCdeep always performs better than the RL-aided SNC, since we optimize the decoding probability directly by building optimal network coding coefficients.

Besides, coding overhead that measures the coding efficiency is compared in Table 1. It is observed that INCdeep has the smallest overhead compared with benchmark coding algorithms for the linear topology. In parallel topology, INCdeep has higher overhead than traditional coding when relay number is small, but as the number of relays increases, the overhead of INCdeep outperforms benchmark algorithms.

Table 1: Comparison of the overhead

Topology	N	Coding Method		
		INCdeep	Fountain+RLNC	RL-aided SNC
Linear	2	14.26 %	24.86 %	30.1%
	6	19.6 %	25.1 %	596 %
	10	22.2 %	26.3 %	9855 %
Parallel	2	222.68 %	89.5 %	292.64 %
	6	290.48 %	345.66 %	936.6 %
	10	454.26 %	598.8 %	1605.32 %

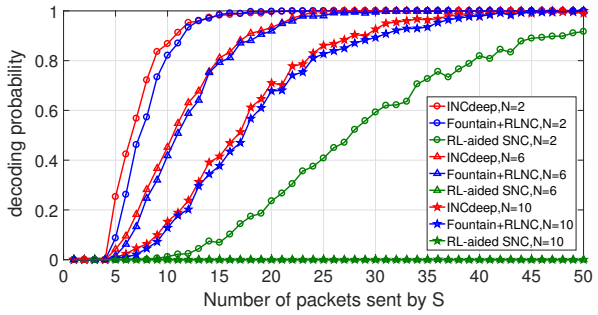


Figure 8: Performance comparison for linear topology, $K = 5$.

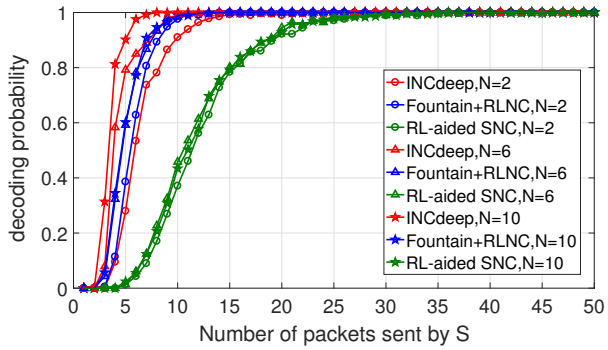


Figure 9: Performance comparison for parallel topology, $K = 5$.

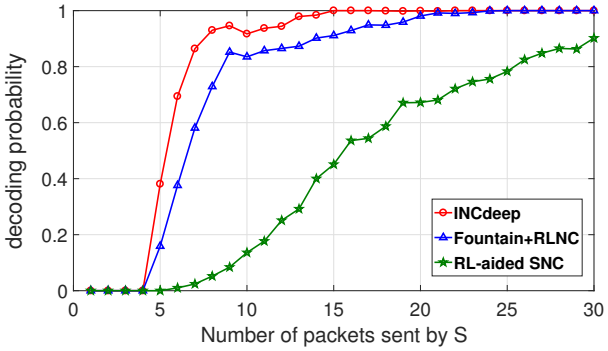


Figure 10: Performance comparison under fast changing environment, $K = 5$.

5.2.6 Fast Changing Environment

Finally we evaluate INCdeep using its generalization ability to compare with benchmark coding algorithms in environments where PER vary much faster. In Fig. 10 we consider PER_{SR} and PER_{RD} be independently randomly drawn from a uniform distributions over [0.0,0,4] for every five packets emitted by source. It is shown that decoding probability of INCdeep outperforms benchmark coding algorithms and further validate that INCdeep adapts well to a dynamic environment.

6 Testbed Experiments

6.1 Experimental Setup

We use Raspberry Pi 3 Model B+ [29] to conduct experiments as in Fig. 11. The Raspberry Pi is a popular single board computer that was initially released in 2012 and has been widely used for numerous applications, including sensor networks [30] [31], internet-of-things [32] [33] and robotics [34] [35]. Raspberry Pi 3B+ has a 1.4 GHz ARM A53 Processor, 1 GB of RAM and integrated wireless and Bluetooth capabilities. We exploit TensorFlow Lite [36] to deploy the trained INCdeep to the Raspberry Pi 3B+.

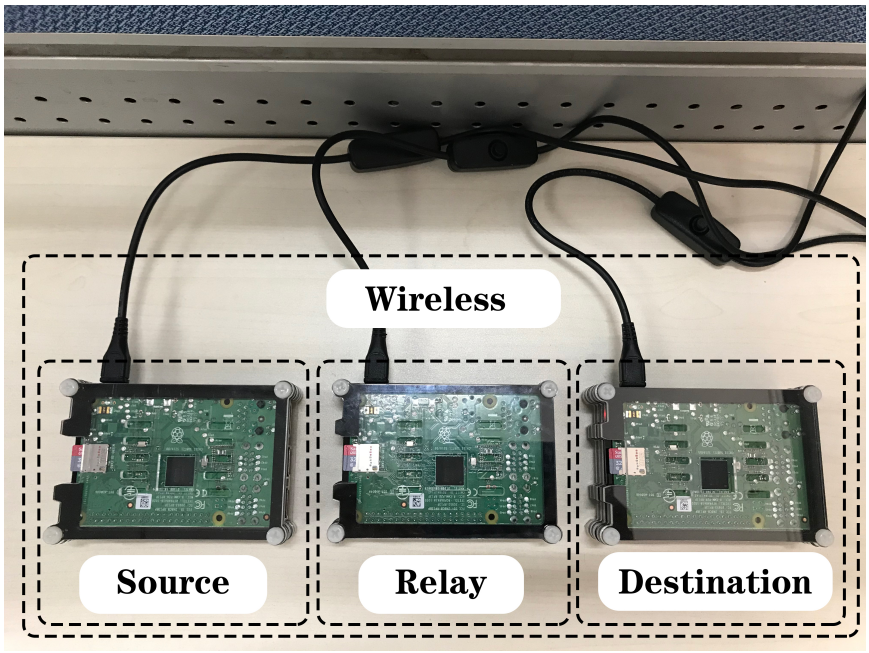
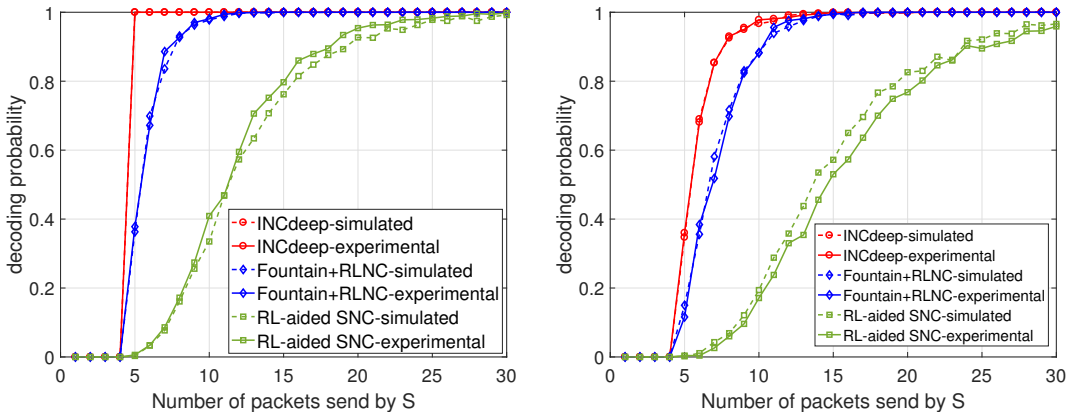


Figure 11: Testbed for Network coding on Raspberry Pi 3B+

6.2 Experimental Results

As seen from Fig. 12 and the small RMSE in Table 2, the results coincide with the simulation results and the decoding probability of INCdeep is higher than the benchmark coding algorithms. As seen from Table 3, the average CPU usage for INCdeep is higher than the traditional random network coding due to the computational complexity in DQN. However, the increased CPU usage gains more efficiency on decoding performance. It shows that the traditional Fountain+RNLC method has 63% performance reduction in decoding performance when the number of K packets sent by source over perfect link $PER = 0$. Meanwhile, comparing to RL-aid SNC, INCdeep performs better in both CPU usage and decoding performance. It proves the great potential of our proposed method on real applications.



(a) PER=0 (b) PER=0.1

Figure 12: Experimental and simulated decoding probability for single-relay network, $PER_{SR=RD} \in \{0, 0.1\}$, q=2 and K=5.

Table 2: RMSE for experimental and simulated results

	INCdeep	Fountain+RNLC	RL-aided SNC
PER=0	0	0.0109	0.0283
PER=0.1	0.0042	0.0153	0.0379
PER=0.3	0.0117	0.0237	0.0154

Table 3: The performance on CPU usage and decoding probability

	CPU usage	Decoding Prob. (K=5)	
		PER=0	PER=0.1
INCdeep	15.9% (+28.2%)	1.00 (0%)	0.86 (0%)
Fountain+RNLC	12.4% (0%)	0.37 (-63%)	0.70 (-18.6%)
RL-aided SNC	18.6% (+50%)	0.05 (-95%)	0.16 (-81.4%)

7 Conclusions

This paper addresses the problem of building adaptive network coding coefficients under the dynamic network. Existing linear network coding including deterministic network coding and RLNC, can not adapt to the dynamic network condition with good decoding performance. We propose INCdeep, an adaptive, DRL approach to automatically build adaptive coding coefficients. Extensive simulation results show that INCdeep has good generalization ability that adapts well in dynamic scenarios, and it achieves higher decoding probability and lower coding overhead compared with benchmark algorithms. Moreover, INCdeep converges fast and runtime costs within 0.8 ms and 1.6 ms to obtain coding coefficients for $K = 5$ and $K = 10$, respectively. Finally, experimental results coincide with the simulation results and also show that INCdeep is feasible to offer a practical deployment.

References

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE transactions on information theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] T. Shang, C. Zhang, K. Li, and J. Liu, "Nonlinear quantum network coding with classical communication resource," in *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–6.

[4] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *ISIT*, July 2003.

[5] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct 2006.

[6] A. F. Dana, R. Gowaikar, B. H. R. Palanki, and M. Effros, "Capacity of wireless erasure networks," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 789–804, Mar 2006.

[7] R. K. D. S. Lun, M. Medard and M. Effros, "On coding for reliable communication over packet networks," *Phys. Commun*, vol. 1, no. 1, pp. 3–20, 2008.

[8] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," 2003.

[9] M. V. Pedersen, D. E. Lucani, F. H. P. Fitzek, C. W. Sørensen, and A. S. Badr, "Network coding designs suited for the real world: What works, what doesn't, what's promising," in *IEEE ITW*, Sep 2013, pp. 1–5.

[10] D. Ferreira, R. A. Costa, and J. Barros, "Real-time network coding for live streaming in hyper-dense wifi spaces," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 773–781, Apr 2014.

[11] V. Mnih, Kavukcuoglu, Koray, Silver, David, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, and A. K. e. a. Fidjeland, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] (2016, jan.)google achieves ai "breakthrough" by beating go champion. bbc. [Online]. Available: https://www.bbc.com/news/technology-35420579

[13] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[14] X. Shi, M. Médard, and D. E. Lucani, "Whether and where to code in the wireless packet erasure relay channel," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 8, pp. 1379–1389, Aug 2013.

[15] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *ACM SIGCOMM Computer Commun, Rev., vol. 37, no. 12, Oct. 2007, pp. 169?80*, vol. 37, no. 12, pp. 169–180, Oct. 2007.

[16] D. J. C. MacKay, "Fountain codes," *IEE Communications*, vol. 152, pp. 1062–1068, 2005.

[17] M. Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, 2002, pp. 271–280.

[18] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM transactions on networking*, vol. 11, no. 5, pp. 782–795, 2003.

[19] C. Fragouli, E. Soljanin, and A. Shokrollahi, "Network coding as a coloring problem," in *In Proceedings*, no. CONF, 2004.

[20] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 41, no. 1. Citeseer, 2003, pp. 11–20.

[21] R.S.Sutton and A.G.Barto, "Reinforcement learning: An introduction," *MIT press Cambridge*, 1998.

[22] R. Gao, Y. Li, J. Wang, and T. Q. S. Quek, "Dynamic sparse coded multi-hop transmissions using reinforcement learning," *IEEE Communications Letters*, vol. Early Access, no. 1-5, June 2020.

[23] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, "Deep learning," *MIT press Cambridge*, 2016.

[24] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133 – 3174, May 2019.

[25] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[26] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE journal on selected areas in communications*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.

[27] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, "Network coding meets multimedia: A review," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1195–1212, Aug. 2013.

[28] A. Apavatjrut, C. Goursaud, K. Jaffrès-Runser, C. Comaniciu, and J. Gorce, "Toward increasing packet diversity for relaying lt fountain codes in wireless sensor networks," *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 52–54, 2011.

[29] Raspberry pi foundation. [Online]. Available: http://www.raspberrypi.org/

[30] B. H. Curtin, R. H. David, E. D. Dunham, C. D. Johnson, N. Shyamku-mar, T. A. Babbitt, and S. J. Matthews, "Designing a raspberry pi sensor network for remote observation of wildlife," in *the 6th Annual Symposium on Hot Topics in the Science of Security*, no. 17, April 2019, pp. 1–2.

[31] M. C. B. Gragasin, M. P. A. Talplacido, and N. A. Macabale, "Throughput evaluation of raspberry pi devices on multihop and multiflow wireless sensor network scenarios," in *International Conference on Signals and Systems (ICSigSys)*, 2017, pp. 256–260.

[32] M. Maksimović, V. Vujović, N. Davidović, V. Milosević, and B. Perisić, "Raspberry pi as internet of things hardware: performances and constraints," *design issues*, vol. 3, no. 8, 2014.

[33] S. Yattinahalli and R. M. Savithramma, "A personal healthcare iot system model using raspberry pi 3," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, April 2018, pp. 569–573.

[34] S. J. Matthews, *Harnessing single board computers for military data analytics.* K. Huggins, Ed. CRC Press, Taylor and Francis, 2018, no. 63-77, ch. 4.

[35] B. Araújo, J. Araújo, R. Silva, and D. Regis, "Machine vision for industrial robotic manipulator using raspberry pi," *13th IEEE International Conference on Industry Applications (INDUSCON)*, pp. 894–901, 2018.

[36] Tensorflow lite. [Online]. Available: https://tensorflow.google.cn/lite/guide/get_started