Self-Creating Random Walks for Decentralized Learning Under Pac-Man Attacks

Xingran Chen, *Member, IEEE*, Parimal Parag, *Senior Member, IEEE*, Rohit Bhagat, *Student Member, IEEE*, and Salim El Rouayheb, *Senior Member, IEEE*

***Abstract*—Random walk (RW)-based algorithms have long been popular in distributed systems due to low overheads and scalability, with recent growing applications in decentralized learning. However, their reliance on local interactions makes them inherently vulnerable to malicious behavior. In this work, we investigate a termination-based attack, termed the "Pac-Man" attack, in which a malicious node probabilistically terminates any RW that visits it. This stealthy behavior gradually eliminates active RWs from the network, effectively halting the learning process without triggering failure alarms. To counter this threat, we propose the CREATE-IF-LATE (CIL) algorithm, which is a fully decentralized mechanism that is resilient to termination-based Pac-Man attacks by enabling the self-creation of RWs and preventing RW extinction under such attacks. Our theoretical analysis shows that the CIL algorithm guarantees several desirable properties, such as 1) non-extinction of the RW population, 2) almost sure boundedness of the RW population, and 3) convergence of RW-based stochastic gradient descent even in the presence of Pac-Man with a quantifiable deviation from the true optimum. Moreover, the learning process experiences at most a linear time delay due to Pac-Man interruptions and RW regeneration. Our extensive empirical results on both synthetic and public benchmark datasets validate our theoretical findings.**

***Index Terms*—Decentralized learning, random-walk (RW) stochastic gradient descent, adversarial attacks, resilience, robustness.**

I. INTRODUCTION

DECENTRALIZED algorithms are becoming increasingly important in modern large-scale networked applications. These algorithms enable a network of nodes/agents, each with access only to local data and a limited local view of the connection graph, to collaborate in solving global computational tasks without relying on centralized coordination. Among the most widely studied approaches are random-walk (RW)-based algorithms [1], [2] and gossip-based algorithms [3], [4]. RW-based algorithms operate by letting one or more tokens perform RWs over the graph; local updates are carried out only at the nodes currently holding tokens.[1] Gossip-based algorithms, by contrast, rely on frequent broadcast communication in the local neighborhood of the nodes where they exchange model snapshots and update locally based on received messages.

While consensus-based methods are powerful, their repeated local broadcasts can lead to large communication overhead in large-scale systems [3], [4]. This motivates the study of RW-based algorithms, which offer a communication-efficient alternative for large networks due to their simplicity and scalability [1], [2]. By propagating information through tokens that traverse the network, RW-based algorithms require only local updates, no global coordination, and low communication overhead. These features have led to their successful application across a wide range of domains [5], [6], [7], [8], [9]. A particularly compelling application is decentralized machine learning, where RWs are used to aggregate learning on data distributed across networked agents [10], [11]. While our focus in this article is on decentralized machine learning, the techniques we develop are broadly applicable to other domains that rely on RW-based methods.

Despite their advantages, RW learning algorithms are susceptible to failure and security threats [12], [13]. In particular, they are vulnerable to a class of attacks that evade classical defenses against poisoning and Byzantine behavior [14], [15] by executing locally correct computations while silently terminating random walks, ultimately halting the learning process. In this work, we consider a specific instance of such behavior, which we term the *Pac-Man attack*, in which a malicious node probabilistically terminates any random walk that visits it. We refer to this adversary as *Pac-Man*, evoking the arcade character known for devouring everything in its path.

A Pac-Man node (malicious node) *mimics* honest behavior under standard fault-tolerance mechanisms [16], [17], [18], [19], [20], [21], such as retransmissions, replication, or timeout-based recovery, thereby making it difficult to detect or isolate via conventional means. In fact, it can terminate any RW that visits it, effectively removing it from the network.

Received 29 December 2025; revised 11 July 2026; accepted 5 October 2026. Date of publication 15 October 2026; date of current version 30 October 2026. This work was supported in part by the National Science Foundation of China under Grant 62401111; in part by the Scheme for Promotion of Academic and Research Collaboration under Grant SPARC/2025-2026/P4239; in part by Qualcomm Inc. through Qualcomm University Relations 6G India; in part by the Anusandhan National Research Foundation under Grant CRG/2023/008854; in part by the Centre for Networked Intelligence (A Cisco Corporate Social Responsibility (CSR) Initiative), Indian Institute of Science (IISc); and in part by the Army Research Laboratory (ARL) under Grant W911NF-24-2-0172. *(Corresponding author: Xingran Chen.)*

Xingran Chen is with the Engineering Systems and Design Pillar, Singapore University of Technology and Design, Singapore 487372 (e-mail: xingranc@ieee.org).

Parimal Parag is with the Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, Karnataka 560012, India (e-mail: parimal@iisc.ac.in).

Rohit Bhagat and Salim El Rouayheb are with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854 USA (e-mail: rb1395@scarletmail.rutgers.edu; sye8@scarletmail.rutgers.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/JSAC.2026.3743835, provided by the authors.

Digital Object Identifier 10.1109/JSAC.2026.3743835

[1]Throughout this paper, we use the terms "random walk" and "token" interchangeably.

1558-0008 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

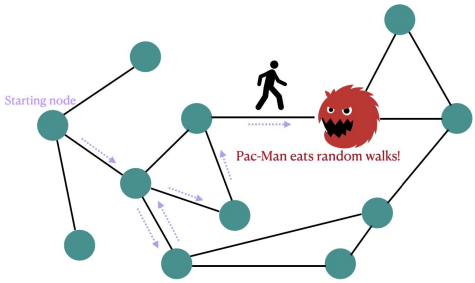


Fig. 1. An illustration of the Pac-Man attack: A malicious node (shown in red) intercepts and "eats" (terminates) any visiting RW with a positive probability. Despite behaving like a benign node to its neighbors, it prevents the random walk from continuing, leading to eventual extinction of all walks in the network.

Even when multiple redundant RWs are used to perform learning across the network, a single Pac-Man node can eventually lead to the extinction of all walks—effectively halting the learning process without triggering any explicit failure or alarm. In this work, we focus on the case of a single Pac-Man node to facilitate clarity of exposition and present the key insights. This restriction does not limit the generality of our results. Both the proposed algorithm and the theoretical framework can be extended to settings with multiple Pac-Man nodes, as illustrated in Fig. 2(a), and discussed in Remark 4 and Appendix G (see the Supplementary Material).

Algorithm 1 CREATE-IF-LATE (CIL) Algorithm

Input: The graph $\mathcal{G}$, the thresholds $\{A_u\}_{u\in\mathcal{B}}$, the initial recording $L_0^{(u)}=0$ for $u\in\mathcal{B}$, the creation probability q, and the initial location of RWs $\mathcal{Z}_0$.
for $t\geqslant 0$ **do**
  for $u\in\mathcal{B}$ **do**
    if $u\in\cup_{j\in\mathcal{Z}_t}X_j(t)$ **then**
      $L_t^{(u)}\leftarrow t$.
    else
      if $t-L_t^{(u)}\leqslant A_u$ **then**
        $L_t^{(u)}\leftarrow L_t^{(u)}$.
      else
        With probability q, node u generates a new RW by replicating the RW that last visited it at time $L_t^{(u)}$.
        If a new RW is created, $L_t^{(u)}\leftarrow t$; otherwise $L_t^{(u)}\leftarrow L_t^{(u)}$.
      end if
    end if
  end for
end for

Motivated by this limitation, we propose a new decentralized self-regulation mechanism based on creation rather than duplication. Our algorithm, termed CREATE-IF-LATE (CIL), is designed to maintain long-term resilience against Pac-Man attacks without relying on system parameter estimation or global coordination (see Algorithm 1). In CIL, each benign node independently decides whether to create a new RW using only local timing information, namely the time elapsed since the last visit of any RW. If this interval exceeds a predetermined threshold, the node infers a possible RW termination and probabilistically creates a new RW by replicating the most recently visited one. This locally triggered mechanism enables resilient RW creation in a fully decentralized manner.

A novel aspect of our analysis lies in the fact that, although the RW population process in our work, bears resemblance to classical branching processes, such as the Galton-Watson process [22, Chapter I], the standard assumptions underlying these models do not hold in our setting. In particular, the RW creation rate is density-dependent and depends on the graph topology as well as the joint system state, including the locations of all RWs and the visitation histories of honest nodes. These features introduce intricate dependencies that violate the independence assumptions required by classical branching process theory, and as such standard tools and results do not directly apply.

A. Contributions

In the following, we summarize the main contributions of this article.

(i) **Resilience to termination-based Pac-Man attacks through local decisions, without the need to estimate system parameters**. Unlike duplication-based methods such as DECAFORK, CIL guarantees long-term survivability of the RW population and prevents permanent extinction (see Fig. 5 in Section VII). Consequently, computational tasks carried by RWs remain operational even in an adversarial environment and do not depend on an accurate estimation of the number of alive RWs.

(ii) **Novel theoretical framework and analytical results in an adversarial setting**. We contribute to developing a novel and rigorous theoretical framework to analyze the evolution of the number of active RWs under CIL, explicitly accounting for the strong interdependence among RWs. We establish that the RW population under CIL is almost surely bounded (Theorem 1) and provide a theoretical regime for selecting the algorithm parameter such that the expected peak population stays below any prescribed level (see Theorem 2). Together, these results guarantee both the long-term stability and controllability of the RW population in the presence of adversarial attacks.

(iii) **Integration with decentralized learning and proof of convergence**. The CIL mechanism can be seamlessly integrated into decentralized random walk stochastic gradient descent (RW-SGD) [10], [23], [24]. We prove that RW-SGD converges under the CIL mechanism, even in the presence of an adversary (see Theorem 3). We further characterize the bias induced by premature RW termination, showing how adversarial behavior skews the optimizer and leads to a bounded deviation from the true global optimum (Proposition 1). Moreover, we establish that the inactive time introduced by the adversary is at most a constant fraction of the current time (Proposition 2), ensuring that the effective

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

learning iterations remain at least linearly increasing in time. Numerical experiments on both synthetic and public benchmark datasets validate our theoretical results.

Notation: We denote the set of first N consecutive positive integers as $[N] \triangleq \{1, \ldots, N\}$. For any finite set $\mathcal{V}$, we denote the collection of probability measures on set $\mathcal{V}$ by $\mathcal{M}(\mathcal{V}) \triangleq \left\{\alpha \in [0,1]^{\mathcal{V}} : \sum_{v\in\mathcal{V}} \alpha_v = 1\right\}$.

B. Related Work

The closest related works are RW-based decentralized optimization algorithms with self-duplication mechanisms in a fully decentralized manner [15], [25], [26]. In DECAFORK [15], [25], each node records RW visit times and estimates the number of active RWs. When the estimated population falls below a predefined threshold, duplication is triggered to replenish the RW population. Similarly, in AC [26] each node records the time interval between consecutive RW visits and duplicates a RW when the observed interval exceeds a prescribed threshold.

Both approaches aim to maintain a sufficient RW population using only local observations. However, the underlying setting and robustness objective are different. The original DECAFORK framework is motivated by non-adversarial catastrophic events, rather than by an adversary that strategically terminates incoming RWs. In addition, the robustness of DECAFORK and AC is highly sensitive to parameter choices and the RW population may eventually collapse (see Fig. 8 in Section VII and Fig. 2(b) in [26]). That is, DECAFORK and AC do not remain consistently resilient under the Pac-Man attack.

The Pac-Man attack shares similarities with other attacks in distributed communication and learning systems; however, it differs in both the entity it targets and its effect on information propagation. The closest analogy is a blackhole attack [27], [28], [29], in which a malicious intermediate node drops packets intended for a particular destination without changing the underlying network topology. In contrast, an RW has no predetermined destination and propagates model updates through its continued movement across the network. Thus, a Pac-Man node terminates an entire persistent computational process, eliminating its future trajectory and all subsequent model updates it would have performed, rather than merely preventing a single packet from reaching its destination. Moreover, because the Pac-Man node remains in the graph and may otherwise behave normally, it can be difficult to detect using only local observations.

The Pac-Man attack also differs from random and localized node-removal attacks [30], [31], [32], which alter the communication topology and may fragment the network. Pac-Man nodes remain active in the graph and instead disrupt learning by terminating visiting RWs. Another related disruptive behavior is that of stragglers, which arise from heterogeneous computation or communication resources and cause some nodes to respond substantially more slowly than others [33], [34], [35], [36], [37], [38]. Stragglers delay the dissemination of information, whereas Pac-Man nodes eliminate the RWs that carry and propagate it.

II. PROBLEM FORMULATION

We consider a decentralized system consisting of N agents that collaboratively perform a computational task without any central coordination.

A. Graph and Random Walks

These agents are modeled as nodes in a *connected* undirected graph, where each agent possesses its own local data and can communicate only with neighboring agents.[2] Without loss of generality, we assume that the Pac-Man agent is indexed as node 1. The communication topology graph is constructed as a finite graph $\mathcal{G} \triangleq (\mathcal{V}, E)$, where the set of agents/nodes is $\mathcal{V} \triangleq [N]$, and $E \subseteq \binom{\mathcal{V}}{2}$ denotes the set of edges. In addition, let $\mathcal{B} \triangleq [N]\setminus\{1\}$ denote the set of benign nodes.

In this system, computation is carried out via RWs on the graph. We assume that there is a fixed and identical transition probability matrix P for each RW on this graph. Each RW carries a token message that is processed and passed along the network. At each discrete time step, only the node currently holding a token performs local computation and updates the message. After computation, the current node forwards the token to a randomly selected neighbors based on the transition probability matrix P. This process continues until a predefined stopping criterion is satisfied. When multiple RWs are present, each is uniquely identifiable (e.g., via an index j) to track their individual progress through the network.

B. Threat Model: The Pac-Man Attack

We are interested in ensuring the resiliency of RW-based algorithms—that is, their ability to prevent total extinction and continue operating effectively even when a malicious node attempts to terminate the RWs. Specifically, we focus on a threat we refer to as the *Pac-Man attack*: a malicious node, termed *Pac-Man*, terminates all incoming RWs without performing the required computation or forwarding the result to a neighbor. We restrict our attention in this paper to the case of a single Pac-Man node to facilitate the presentation of the proposed algorithm and its theoretical analysis. This setting already captures the core difficulty of the problem: even one such adversary is sufficient to cause eventual extinction of all random walks with probability one, thereby completely halting the learning process. Extensions to settings with multiple Pac-Man nodes are discussed in subsequent sections.

The Pac-Man can be particularly dangerous due to its ability to remain hidden through a deceptive behavior.

(i) The Pac-Man node can reply positively to all network-level fault-tolerance checks and controls, for example retransmissions, making it difficult to distinguish it from benign nodes despite its malicious actions.

(ii) To avoid detection, the Pac-Man node terminates incoming RWs independently with a probability $\zeta \in (0, 1]$,

[2]Without loss of generality, we restrict our analysis to connected graphs. For disconnected graphs, the proposed theoretical framework can be applied separately to each connected component, and the analysis proceeds analogously.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

referred to as the termination probability. This randomized behavior allows the Pac-Man to conceal itself among benign nodes. If the Pac-Man node terminates all incoming RWs, it would never propagate RWs to its neighbors, making it easily identifiable as malicious since no RWs would be observed from that node over a long time horizon. In contrast, by terminating incoming RWs only with a positive probability, the Pac-Man can intermittently forward RWs, thereby blending in with normal nodes. Under decentralized operation, where each node relies solely on local observations, this stochastic behavior significantly delays detection by neighboring nodes.

In decentralized systems, traditional fault-tolerance mechanisms are challenging when facing such an adversary, because unlike conventional failures—such as link disconnections, node crashes, or consistently faulty behavior [39]—a Pac-Man remains active and selectively disrupts system operation while appearing compliant. Our goal is to design a mechanism that prevents a Pac-Man node from causing a system-wide failure by eliminating all RWs.

Another commonly used fault-tolerance technique is the introduction of redundancy. However, static redundancy alone is *ineffective* in this adversarial setup. Simply starting with multiple RWs does not guarantee their survival: for any small termination probability $\zeta > 0$, all RWs will eventually be terminated with probability 1, causing the task to fail. Thus, redundancy cannot serve as a long-term solution.

C. Random Walk Stochastic Gradient Descent

In this paper, the global task is defined as the standard distributed optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^m} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^m} \mathbb{E}_{u\sim\pi}\left[f_u(\mathbf{x})\right], \tag{1}$$

where each node u possesses a local function $f_u(\mathbf{x})$, π is a target sampling distribution, and m is a positive integer.

Under local communication constraints, solving problem (1) using RW-based stochastic gradient descent (RW-SGD) algorithms has proven to be highly effective [23], [40], [41]. This decentralized approach exploits local interactions within a network to collaboratively achieve global optimization objectives.

Given a target sampling distribution π (assuming $\pi_u > 0$ for all $u \in \mathcal{V}$) and a connected graph $\mathcal{G}$, the Metropolis-Hastings algorithm [41] provides a principled way to construct a RW on $\mathcal{G}$ with a transition matrix P such that π is the stationary distribution of the Markov chain defined by P.[3] The RW-SGD is outlined as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma_t \hat{g}_{v_t}(\mathbf{x}_t), \tag{2}$$

where v_t is the node visited by the random walk at time t, and $\hat{g}_{v_t}(\mathbf{x}_t)$ denotes the gradient or sub-gradient at that node. Here, the sequence $\{v_t\}_{t\in\mathbb{N}}$ is a RW that evolves according to P.

[3]The transition matrix P depends on both the target sampling distribution π and the graph $\mathcal{G}$. We re-parameterize it as $P_{\pi,\mathcal{G}} \in \mathbb{R}_+^{N\times N}$, and simply write P when there is no risk of confusion.

In the absence of the Pac-Man node, RW-SGD is known to converge to the global optimum under standard assumptions [23], [40], [41]. However, the introduction of a Pac-Man node fundamentally alters the dynamics: by probabilistically terminating incoming RWs, it becomes unclear whether RW-SGD remains convergent. Even if convergence occurs, it is not known a priori whether RW-SGD converges to the true optimum or to a biased solution.

D. Designable Properties of the Decentralized Mechanism

As stated in Sections II-B and II-C, the entire goal is to design a *decentralized mechanism* such that (i) the global computational task can be carried out via RWs even in the presence of a Pac-Man node, and (ii) when the proposed decentralized mechanism is integrated with RW-SGD, it remains effective and convergent.

At the beginning of time slot t, let $\mathcal{Z}_t$ denote the set of indices of active RWs, and define the random variable $Z_t \triangleq |\mathcal{Z}_t|$ as the total number of active RWs at that moment. For each RW $j \in \mathcal{Z}_t$ at time t, we denote its location at time t by $X_j(t) \in \mathcal{V}$. At the initial time $t = 0$, we denote the number of initial RWs as $Z_0 = z_0$, where z_0 is a predetermined scalar. To achieve our entire goal, the proposed algorithm must have the following desirable properties.

1) No Permanent Extinction: To ensure that the global task can be accomplished, we must avoid permanent extinction of the RW population almost surely:

$$\Pr\left(\exists t_0,\, \forall t \geqslant t_0,\, Z_t = 0 \,|\, Z_0 = z_0\right) = 0. \tag{3}$$

2) No Blowup: To ensure system stability, we must avoid uncontrolled growth in the number of RWs. The algorithm should keep the RW population bounded almost surely:

$$\Pr\left(\sup_t Z_t < \infty \,\middle|\, Z_0 = z_0\right) = 1. \tag{4}$$

3) Convergence: All active RWs under RW-SGD must converge to the same minimizer, in the following sense:

$$\lim_{t\to\infty} \mathbb{E}\left\|\mathbf{x}_t^{(j_t)} - \tilde{\mathbf{x}}^\star\right\| = 0,\ \forall j_t \in \mathcal{Z}_t, \tag{5}$$

where $\tilde{\mathbf{x}}^\star$ denotes the limiting point attained by the RW-SGD iterates across active RWs. If the limit in (5) exists, we further aim to characterize the approximation error with respect to the true optimizer $\mathbf{x}^\star$, by bounding the deviation $\|\tilde{\mathbf{x}}^\star - \mathbf{x}^\star\|$.

Remark 1: It is worth noting that, (5) characterizes the convergence behavior along any active RW. This is a substantially stronger conclusion than results based on averaging over all active RWs or over all benign nodes, as is commonly done in gossip-based algorithms [3].

III. The Create-If-Late Algorithm

We present the Create-If-Late (CIL) algorithm, a decentralized mechanism for adaptively creating RWs with the desirable properties of (a) no permanent extinction, (b) no blowup, and (c) convergence, as formalized in (3), (4) and (5). We first describe the CIL algorithm and then summarize our main theoretical results.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

A. Create-If-Late Algorithm

We outline the CIL algorithm in Algorithm 1. The core idea is that each benign node creates a new RW if it has not been visited for too long. In particular, each benign node $u \in \mathcal{B}$ maintains a variable $L_t^{(u)}$, which records the last time *before* t that node u was visited by any RW. Using this timestamp, node u computes the elapsed time since its most recent visit as $t - L_t^{(u)}$. If this interval exceeds a predetermined threshold A_u, node u infers that some RW may have been lost, and with probability q, creates an *identical* copy of the most recently visited RW stored in its local cache (lines 9, 10, 11 in Algorithm 1).[4] The newly created RW is then propagated independently according to the transition matrix P. If multiple RWs arrive at node u simultaneously, the node retains any one of them at random.

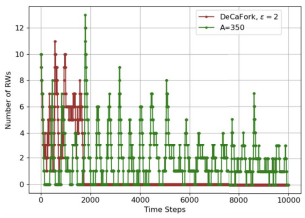
Remark 2: We observe that, under the construction of the CIL algorithm, whenever there are no RWs on the graph, at least one new RW will be generated within at most $\min_{u\in\mathcal{B}} A_u$ time slots from the last time all RWs got terminated. Consequently, the RW population can never become permanently extinct, and the objective of achieving no permanent extinction defined in (3) is inherently satisfied.

Remark 2 is illustrated in Fig. 2(a), where we compare the resilience of the proposed CIL algorithm with the baseline DECAFORK algorithm in [15] and [25]. The figure plots the evolution of the number of RWs over time. Under the CIL algorithm, even when the creation threshold is inappropriately chosen, the RW population may temporarily go extinct (hit 0) but eventually recovers after a finite period (green curve). In contrast, when the algorithm parameter in the DECAFORK algorithm is inappropriately chosen, the RW population goes extinct permanently (brown curve). This behavior highlights a key robustness advantage of the proposed algorithm over the baseline. This phenomenon is not unique to complete graphs; similar behavior is observed on other graph topologies as shown in Fig. 8 in Section VII.

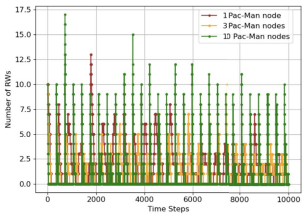
Next, we numerically demonstrate that the proposed algorithm remains effective in the presence of multiple Pac-Man nodes. In Fig. 2(b), we consider a complete graph with 100 nodes, where all benign nodes share the same creation threshold $A = 350$. We examine three scenarios with 1, 3, and 10 Pac-Man nodes, respectively. The results show that even in the presence of a large number of Pac-Man nodes (up to 10% of the total nodes), the RW population is able to recover after a finite time, following the elimination of all RWs by the Pac-Man nodes. This numerical result also highlights the robustness of the proposed algorithm against multiple Pac-Man nodes. As discussed in Sections V and VI, our theoretical framework can be straightforwardly extended to accommodate multiple Pac-Man nodes. For the remainder of this article, we focus on the case of a single Pac-Man node when presenting our results to improve the clarity of exposition.

Finally, the algorithm design directly affects resource consumption, such as communication cost. When the creation threshold A is large or the creation probability q is small, new RWs are created less frequently. This reduces resource consumption but requires a longer time to complete computational tasks. Conversely, when the creation threshold A is small or the creation probability q is large, new RWs are created more frequently, resulting in faster task completion at the expense of higher communication cost.

[4]Similar timeout-based mechanisms have been studied for handling stragglers in decentralized learning, where the timeout threshold affects the trade-off between latency and learning performance [33].



(a) CIL and DECAFORK



(b) Multiple Pac-Man nodes

Fig. 2. (a): Comparison of the number of RWs under the CIL and DECAFORK algorithms in a complete graph. (b): Number of RWs under the CIL algorithm in a complete graph with multiple Pac-Man nodes.

B. Main Results

We summarize here our main results, which provide theoretical guarantees of no permanent extinction, no blowup, and convergence. The details of analysis are provided in Sections V and VI.

1) No Permanent Extinction: As stated in Remark 2, by construction, the CIL algorithm prevents the RW population from becoming permanently extinct with probability one; that is, (3) holds.

2) No Blowup: With a Pac-Man node present in the graphs, the CIL algorithm ensures that the RW population size remains bounded with probability one; that is, (4) holds, (see Theorem 1). Moreover, we provide a theoretical regime for selecting the creation probability q such that the expected peak population, $\sup_{t\geqslant 0}\mathbb{E}[Z_t]$, becomes independent of the network size N (see Theorem 2).

3) Convergence: Under the CIL mechanism, RW-SGD is shown to converge; that is, (5) holds (see Theorem 3). In addition, we show that it converges to a biased optimum, and we characterize the error between this skewed optimizer and the true optimizer, $\|\tilde{\mathbf{x}}^\star - \mathbf{x}^\star\|$ (see Proposition 1).

IV. PRELIMINARIES: NOTATIONS, DEFINITIONS, AND ASSUMPTIONS

In this section, we introduce the notation, definitions, and assumptions used in the system model and its transition dynamics.

Definition 1 (Communication topologies and RWs): A communication topology is defined as a finite directed graph $\mathcal{G} \triangleq (\mathcal{V}, E)$ with the set of nodes $\mathcal{V} = [N]$ and the set of edges $E \subseteq \binom{\mathcal{V}}{2}$. Each RW $X_j : \Omega \to \mathcal{V}^{\mathbb{Z}_+}$ on this graph is assumed to be *i.i.d.* and can be defined by the common transition probability matrix $P : \mathcal{V} \to \mathcal{M}(\mathcal{V})$, where the probability of transition from node u to node v in one time step at time $t \in \mathbb{Z}_+$, is

$$P_{uv} \triangleq \Pr\left(X_j(t+1) = v \,|\, X_j(t) = u\right).$$

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

We will call the RW on graph $\mathcal{G}$ aperiodic if transition probability matrix P is aperiodic.

Remark 3: Without loss of generality, we assume that $P_{uv} > 0$ for all nodes v connected to node u in graph $\mathcal{G}$. Therefore, transition matrix P is irreducible iff graph $\mathcal{G}$ is connected.

Definition 2 (Timing conventions): We define the timing conventions of the system as follows:

(1) At the beginning of time slot t, let $\mathcal{Z}_t$ denote the set of indices of active RWs. Each active RW $j \in \mathcal{Z}_t$ is associated with a birth time $\theta_j \geqslant 0$ and an initial location $u_j \triangleq X_j(\theta_j) \in \mathcal{V}$. Specifically, let $Z_t = |\mathcal{Z}_t|$.

(2) At the end of time slot t, each active RW $j \in \mathcal{Z}_t$ moves from its current location to a randomly selected neighbor in the next time slot. We denote by $X_j(t+1)$ the location of RW j after this movement at time $t+1$.

(3) Upon arrival at location $X_j(t+1)$ at time $t+1$, termination operation at node 1 and creation operations at other nodes is performed (if applicable).

Under the model assumption, the Pac-Man node is fixed at location 1. We next present a formal definition of the system including the Pac-Man node.

Definition 3: Consider a communication topology $\mathcal{G} = (\mathcal{V}, E)$, as defined in Definition 1, where node 1 acts as the Pac-Man, such that if a RW visits node 1, it is sent to a death node 0[5] with a termination probability $\zeta \in (0,1]$. That is, we augment graph $\mathcal{G}$ to $\mathcal{G}' = (\mathcal{V}', E')$ where $\mathcal{V}' \triangleq \mathcal{V} \cup \{0\}$ and $E' \triangleq E \cup \{(1,0)\}$. In the presence of this Pac-Man, the original RW transition probability matrix P now changes to P', where for each state $u, v \in \mathcal{V}'$

$$P'_{uv} \triangleq \begin{cases} P_{uv}, & u \in \mathcal{B}, v \in \mathcal{V}, \\ (1-\zeta)P_{uv}, & u = 1, v \in \mathcal{V}, \\ \zeta, & u = 1, v = 0, \\ 1, & u = 0, v = 0 \\ 0, & u = 0, v \in \mathcal{V}. \end{cases} \tag{6}$$

Based on Definition 3, the transition matrix P' can be characterized by two cases: $\zeta = 1$ and $0 < \zeta < 1$. When $\zeta = 1$, the Pac-Man node becomes an absorbing state, whereas when $0 < \zeta < 1$, the Pac-Man node is non-absorbing. In particular,

(1) When $\zeta = 1$, the Pac-Man node eats every incoming RW, and the local data at the Pac-Man node cannot be utilized, making it an absorbing state. From Definition 3, the death node 0 is also absorbing. Therefore, we merge the Pac-Man node and the death node into a single absorbing one, which we continue to denote as node 1. The corresponding transition matrix P' becomes

$$P' = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{bmatrix} \triangleq \begin{bmatrix} 1 & \mathbf{0}_{1\times(N-1)} \\ R^{(1)} & Q^{(1)} \end{bmatrix}. \tag{7}$$

We observe that $Q^{(1)} \in [0,1]^{\mathcal{B}\times\mathcal{B}}$ is sub-stochastic matrix where $Q^{(1)}_{uv} = P_{uv}$ for each $u, v \in \mathcal{B}$, and $R^{(1)} \in [0,1]^{\mathcal{B}\times 1}$ column vector where $R^{(1)}_u = P_{u1}$ for each benign node $u \in \mathcal{B}$.

(2) When $0 < \zeta < 1$, the local data at the Pac-Man node cannot be reliably utilized, because any RW visiting it is terminated with probability strictly less than 1. In this case, only the death node 0 is absorbing. For clarity, we denote the location of the death node as 0. According to Definition 3 and denoting $\bar{\zeta} \triangleq 1 - \zeta$, the corresponding transition matrix P' becomes

$$P' = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \zeta & \bar{\zeta}P_{11} & \cdots & \bar{\zeta}P_{1N} \\ 0 & P_{21} & \cdots & P_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & P_{N1} & \cdots & P_{NN} \end{bmatrix} \triangleq \begin{bmatrix} 1 & \mathbf{0}_{1\times N} \\ R^{(\zeta)} & Q^{(\zeta)} \end{bmatrix}. \tag{8}$$

We observe that

$$R^{(\zeta)} \triangleq \begin{bmatrix} \zeta \\ \mathbf{0}_{(N-1)\times 1} \end{bmatrix}, \qquad Q^{(\zeta)} \triangleq \begin{bmatrix} \bar{\zeta}P_{11} & \bar{\zeta}S \\ R^{(1)} & Q^{(1)} \end{bmatrix}.$$

The row vector $S \in [0,1]^{1\times\mathcal{B}}$ such that $S_{1v} = P_{1v}$ for each benign node $v \in \mathcal{B}$.

Remark 4: If $\zeta = 1$, the analysis extends straightforwardly to the setting with multiple Pac-Man nodes. In this case, all Pac-Man nodes can be treated as a single "super node", which acts as an absorbing state of the system. The resulting analysis is essentially identical to that of the single Pac-Man node case. If $\zeta < 1$, the analysis becomes more involved, since the transition probability to the absorbing state (i.e., the death node 0) depends on the identity of the Pac-Man node through (a) their corresponding termination probabilities and (b) their connectivity to the benign nodes. Nevertheless, the analysis can still be carried out within the theoretical framework proposed for the single Pac-Man node setting, although the analysis would be more tedious for a larger number of Pac-Man nodes.

Definition 4 (Robustly Connected Graph): A graph $\mathcal{G}$ is robustly connected if

(i) every pair of benign nodes in $\mathcal{B}$ is connected by a path that avoids the Pac-Man node, and

(ii) the Markov chain corresponding to each active RW is aperiodic.

Remark 5: According to Definition 4, a robustly connected graph cannot be partitioned into two disjoint components by Pac-Man. In addition, in a robustly connected graph, the Markov chain corresponding to each active RW is irreducible and aperiodic.

Definition 5: For $\zeta \in (0,1]$, let $Q^{(\zeta)}$ be the matrix defined in (7) and (8). Let $\alpha^{(\zeta)}$ denote the (unique) maximum eigenvalue of $Q^{(\zeta)}$,[6] and let $\nu^{(\zeta)}$ denote the associated left normalized positive eigenvectors with unit sum.

Assumption 1: Each RW has an *i.i.d.* evolution on this graph with transition probability matrix P, conditioned on their initial locations.

Assumption 2: The graph $\mathcal{G}$ defined in Definition 1 is a robustly connected graph.

[5]The death node is a virtual node used only for a clearer presentation.

[6]We assume a unique maximum eigenvalue for simplicity. If $Q^{(\zeta)}$ has multiple dominant eigenvalues, the arguments can be extended using a standard Jordan decomposition.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

Next, we adopt the common assumptions used in standard distributed optimization problems, as follows.

Assumption 3: Each local function $f_u(\mathbf{x})$ in (1) with $u \in \mathcal{V}$ is μ-strongly convex and L-smooth.

Assumption 4: Bounded norm of the local gradient at the global optimum $\mathbf{x}^\star$, i.e. $\sup_{u\in\mathcal{V}} \|\nabla f_u(\mathbf{x}^\star)\|^2 \leqslant \sigma^2$, where $\mathbf{x}^\star$ is the minimizer of (1).

In the remainder of this article, we assume that Assumptions 1–4 hold.

For clarity of presentation, the theoretical results in Section V and Section VI are presented for a single Pac-Man node, and Theorem 2 is further specialized to a complete-graph setting. These assumptions are made only to simplify the notation and analysis. The results extend to multiple Pac-Man nodes on a general connected graph with minor modifications, provided that the robustly connected graph condition in Definition 4 and Assumption 2 holds; that is, removing the Pac-Man nodes does not disconnect the network. Further details are provided in Appendix G (see the Supplementary Material).

V. Population Boundedness With Pac-Man

In this section, we introduce one Pac-Man node, and we examine the long-term behavior of the RW population process $\{Z_t\}_{t\in\mathbb{N}}$.

In the following theorem, we show that, regardless of the creation thresholds $\{A_u : u \in \mathcal{B}\}$ chosen by the benign nodes, the RW population remains almost surely bounded over time. This boundedness is crucial for ensuring algorithmic stability (in terms of population) and preventing the network from being flooded with RWs.

Theorem 1: On any finite graph $\mathcal{G}' = (\mathcal{V}', E')$ of Definition 3, with $z_0 \geqslant 1$, $A_u \geqslant 1$, $q \leqslant 1$, and $\zeta \in (0, 1]$, the CIL algorithm ensures that $\limsup_{t\to\infty} Z_t < \infty$ almost surely.[7]

Proof: (Roadmap) Let $\mathcal{F}_t$ denote the history of the evolution of all RWs until time t. First, we prove that the d-step drift of Z_t is bounded as $\mathbb{E}[Z_{t+d} - Z_t \mid \mathcal{F}_t] \leqslant -c\zeta Z_t + (N-1)d$, where d is the graph diameter and c is the smallest probability of reaching the Pac-Man within d steps from any node $u \in \mathcal{B}$. The formal definitions of d and c are provided in Definition 9 in Appendix A (see the Supplementary Material). This inequality implies that the expected drift in the number of RWs after d steps is negative when Z_t is large. This prevents Z_t from diverging. The full proof is in Appendix A (see the Supplementary Material). A discussion of the extension to multiple Pac-Man nodes is provided in Appendix G-A (see the Supplementary Material). □

A natural follow-up question is whether we can characterize the maximum number of random walks at across all times t. As such, we next analyze the peak of expected number of RWs, defined as

$$\bar{Z}^\star = \sup_{t\geqslant 0} \mathbb{E}[Z_t]. \tag{9}$$

[7]We abstract away network delays and other lower-layer communication issues to focus on the proposed random-walk-based algorithm and its resilience to termination. These issues are assumed to be handled by the underlying network protocols.

We focus on the special case in which the creation thresholds satisfy $A_u = 1$ for all $u \in \mathcal{B}$. Under this choice, a new RW is created immediately whenever a benign node is not visited by a RW, thereby making RW creation as frequent as possible. Consequently, this setting corresponds to the worst-case scenario for RW proliferation. The expected peak number derived under this setting therefore provides a universal upper bound for all threshold choices with $A_u > 1$. Since larger thresholds lead to less frequent RW creation, the resulting bound may be conservative for general values of A_u.

Theorem 2: Consider a complete graph with N nodes and a *uniform* target sampling distribution $\pi = \left(\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}\right)$. Suppose there are z_0 initial RWs, each node has a creation threshold $A_u = 1$, and the creation probability is $q \in (0, 1]$. The expected peak population under the CIL algorithm satisfies

$$\bar{Z}^\star = O\left(\frac{qN^2}{\zeta}\right),$$

where $\bar{Z}^\star$ is defined in (9).

Proof: (Roadmap) By unrolling the recursion derived from the transition probabilities P_{uv} and the termination probability ζ, we obtain the following upper bound for every $t \geqslant 0$:

$$\mathbb{E}[Z_t] \leqslant \frac{q}{\zeta}(N-1)N + \left(1 - \frac{\zeta}{N}\right)^t \left(z_0 - \frac{q}{\zeta}(N-1)N\right).$$

Consequently,

$$\bar{Z}^\star \leqslant \max\left\{z_0, \frac{q}{\zeta}N^2\right\}. \tag{10}$$

The full proof is given in Appendix B (see the Supplementary Material). A discussion of the extension to multiple Pac-Man nodes and general connected graphs is provided in Appendix G-B (see the Supplementary Material). □

Corollary 1: Under the same setting in Theorem 2, under the CIL algorithm, for any $0 \leqslant \alpha \leqslant 2$, if $z_0 = \Theta(1)$ and $\zeta = \Theta(1)$, $q = N^{-\alpha}$, then $\bar{Z}^\star = O(N^{2-\alpha})$.

Proof: By substituting the appropriate values of (z_0, ζ, q) into (10), we obtain the desired bounds. □

From a practical perspective, Corollary 1 provides an explicit rule for selecting the creation probability q according to the desired RW population. In many practical settings, the initial number of RWs is typically $z_0 = \Theta(1)$, while the termination probability ζ, determined by the Pac-Man, satisfies $\zeta \in (0, 1]$ and is usually not too small, implying $\zeta = \Theta(1)$. Under these conditions, if the system designer wishes to keep the expected peak RW population at the order of $O(N^{2-\alpha})$, where $0 \leqslant \alpha \leqslant 2$, then q can be selected as $q = N^{-\alpha}$. For example, $q = N^{-2}$ keeps the expected peak RW population bounded independently of N, $q = N^{-1}$ allows it to grow linearly with N, and $q = \Theta(1)$ may result in quadratic growth. In implementation, one may use $q = \min\{1, rN^{-\alpha}\}$, where α specifies the desired population scaling and the constant $r > 0$ can be tuned according to the available resource budget.

A similar population-control principle also applies to other connected graphs. Although the required scaling of the creation probability may differ from $N^{-\alpha}$, the expected peak number of RWs, $\bar{Z}^\star$, can still be controlled through an appropriate choice of q. Therefore, rather than selecting the creation

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

probability heuristically, one may first specify a desired growth rate for the RW population and then choose q accordingly.

VI. Convergence of the Learning Algorithm

In this section, we study how the CIL algorithm affects the convergence of RW-SGD in the presence of a Pac-Man node. Since the Pac-Man node terminates incoming RWs with probability ζ, the local data at the Pac-Man node is only partially exploited in the learning process. This raises two questions: (i) whether RW-SGD still converges, and (ii) if it does, whether the resulting limit point $\tilde{\mathbf{x}}^\star$ (defined in (5)) coincides with the true optimizer $\mathbf{x}^\star$.

It is important to note that each single RW will eventually be terminated by the Pac-Man node with probability one. To address the two questions above, rather than relying on a single RW, we need to analyze the behavior of a *chain* of RWs. Specifically, when a benign node has not been visited for too long, it creates a new RW by copying the model of the most recently visiting RW. We refer to this most recent RW as the *parent* and newly created RW as its *child*.

Definition 6: (Chain of RWs) Start with an initial RW $j_0 \in [z_0]$. At each $s \geqslant 0$, if j_s has at least one child, let j_{s+1} be a uniformly chosen child of j_s; otherwise, set $j_{s'} = -1$ for all $s' \geqslant s$ to mark termination. We refer to $\{j_s : s \in \mathbb{N}\}$ as a chain of RWs.

An illustrative example of a chain of RWs is shown in Fig. 3. Each parent-child pair is separated by a random waiting period that is at least as large as the corresponding creation threshold, due to the creation probability q. However, these waiting periods do not affect learning performance, as they simply correspond to *idle* times during which no updates are applied at the node. Thus, we can **conceptually condense** all waiting periods and treat the entire chain as an *effective single* RW that inherits all model updates along the parent–child lineage. We therefore study the limiting behavior of such chains. Since every benign node can create new RWs, at least one infinite parent-child chain must almost surely exist.

In the presence of the Pac-Man node, a chain of RWs does not follow the original transition matrices P (or P'). The effective transition probability matrix now depends explicitly on the termination probability ζ. We denote this effective transition matrix by $P^{(\zeta)}_{\text{chain}}$, and its stationary distribution by $\pi^{(\zeta)}_{\text{chain}}$, where the explicit forms of $P^{(\zeta)}_{\text{chain}}$ and $\pi^{(\zeta)}_{\text{chain}}$ are given in (12) and (13), respectively.

Theorem 3: Under the CIL mechanism, RW-SGD converges to the minimizer of the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^m} \tilde{f}(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^m} \mathbb{E}_{u\sim\pi^{(\zeta)}_{\text{chain}}}[f_u(\mathbf{x})], \tag{11}$$

where

$$\pi^{(\zeta)}_{\text{chain}} = \begin{cases} [0, \nu^{(1)}]\zeta = 1, \\ \nu^{(\zeta)}\zeta \in (0,1), \end{cases} \tag{12}$$

and $\nu^{(\zeta)}$ is defined in Definition 5.

Proof: (Roadmap) We first show that, as long as the initial RW j_0 remains active, the distribution of the chain coincides with that of j_0. Next, we demonstrate that this distribution converges to the quasi-stationary distribution [42] of RW j_0, which can be computed following the approach in [43]. Consequently, under the CIL algorithm, RW-SGD converges to the surrogate optimization problem (11). The full proof is given in Appendix C (see the Supplementary Material). A discussion of the extension to multiple Pac-Man nodes is provided in Appendix G-C (see the Supplementary Material). □

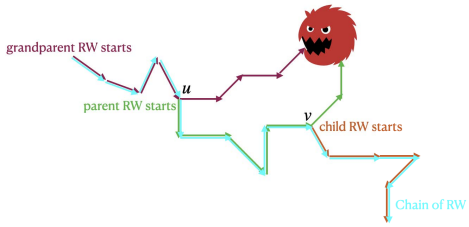


Fig. 3. An example of a chain of RWs. When a benign node u has not been visited for A_u consecutive time slots, it creates a new RW (green). Similarly, when node v has not been visited for A_v time slots, it creates another new RW (orange). The blue trajectory connects these RWs and forms a chain.

Now we derive the explicit form of the effective transition probability matrix $P^{(\zeta)}_{\text{chain}}$, which will be used to quantify the limiting model from the true global optimum $\mathbf{x}^\star$. As stated before, without loss of generality, we *condense the waiting time* between the each pair of parent and child (i.e., the flat segments in Fig. 9). Then, $P^{(\zeta)}_{\text{chain}}$ takes the following form (the proof is provided in Appendix D) (see the Supplementary Material):

$$\left[P^{(\zeta)}_{\text{chain}}\right]_{uv} = \frac{Q^{(\zeta)}_{uv}}{\sum_v Q^{(\zeta)}_{uv}} \tag{13}$$

and $Q^{(\zeta)}$ is defined in (7) and (8).

In the standard RW-SGD algorithm, it is well known that if the stepsize η_t decreases with the number of local-update iterations t and tends to 0, the algorithm converges to a deterministic point; if η_t remains constant, it converges to a random variable [23], [40], [41].

Proposition 1 (Shift of Optima): Define $P^{(\zeta)}_{\text{chain}}$ as in (13), and let $\gamma^{(\zeta)}_{\text{chain}}$ be its spectral gap. Denote by $\|\cdot\|_{\text{TV}}$ be the total variation distance, and let $\mathbf{x}_0$ be the starting point. Let $\pi^{(\zeta)}_{\text{chian}}$ be the stationary distribution defined in (12). Under the CIL mechanism:

(1) If the step size η_t decreases with the number of iterations and tends to 0, then RW-SGD converges to the minimizer $\tilde{\mathbf{x}}^\star$ of (11). Moreover,

$$\frac{1}{L}\|\nabla f(\tilde{\mathbf{x}}^\star)\| \leqslant \|\tilde{\mathbf{x}}^\star - \mathbf{x}^\star\| \leqslant \frac{1}{\mu}\|\nabla f(\tilde{\mathbf{x}}^\star)\|.$$

(2) If the step size keeps constant, i.e., $\eta_t = \eta < \frac{1}{L}$, then, the expected error satisfies

$$\lim_{T\to\infty} \mathbb{E}\left[\|\tilde{\mathbf{x}}_T - \mathbf{x}^\star\|\right] \leqslant \frac{\eta L\sigma^2}{\gamma^{(\zeta)}_{\text{chain}}\mu^2} + \frac{\left\|\tilde{\nu}^{(\zeta)} - \pi\right\|^2_{TV}\sigma^2 L}{\mu^3}.$$

Proof: These inequalities are derived from existing results, and the full proof is given in Appendix E (see the Supplementary Material). A discussion of the extension to multiple Pac-Man nodes is provided in Appendix G-D (see the Supplementary Material). □

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

Remark 6: An interesting problem is whether the sampling bias caused by Pac-Man attacks can be reduced using importance sampling [44] and update reweighting [45]. However, in a fully decentralized network, the location of the Pac-Man node and the global transition probability matrix are unknown. Standard reweighting factors are therefore unavailable, while estimating them from local observations may introduce substantial error and amplify adversarial updates. Developing reliable bias-correction methods for this setting is left for future work.

Since temporary extinctions may cause the RW population to become inactive, not every physical time slot contributes to learning. To characterize its real-time convergence behavior, we focus on the number of local-update iterations completed by time t. For an infinite chain of RWs, let Iter_t denote this number. The ratio $\frac{\mathbb{E}[\mathrm{Iter}_t]}{t}$ measures the expected fraction of time steps that effectively contribute to learning, and therefore partially characterizes the convergence speed in real time. When $\mathrm{Iter}_t = t$, every time step produces a local update, so the algorithm progresses at the same rate as classical RW-SGD. When $\mathrm{Iter}_t < t$, some time steps are lost due to RW termination and regeneration, and the real-time convergence is slowed down is slowed down according to the effective update ratio.

To characterize the communication overhead, let C_t denote the total number of transmissions on the graph up to time t. Since each local update is associated with one RW transmission, Iter_t also counts the number of transmissions generated by a chain. Note that C_t counts all transmissions generated by all active RWs on the graph. Therefore, we have $\mathrm{Iter}_t \leqslant C_t$.

Proposition 2 (Convergence speed & communication overhead): Under the same setting in Theorem 2, with $A_u = A \geqslant 1$, the CIL algorithm guarantees that the expected number of local-update iterations up to time t satisfies

$$1 \geqslant \frac{1}{t}\mathbb{E}[\mathrm{Iter}_t], \quad \lim_{t\to\infty}\frac{1}{t}\mathbb{E}[\mathrm{Iter}_t] \geqslant \frac{\frac{N}{\zeta}}{\frac{N}{\zeta}+A-1+\frac{1}{q}}, \tag{14}$$

$$\lim_{t\to\infty}\frac{1}{t}\mathbb{E}[C_t] \geqslant \frac{\frac{N}{\zeta}}{\frac{N}{\zeta}+A-1+\frac{1}{q}}. \tag{15}$$

Proof: (Roadmap) The upper bound $\mathbb{E}[\mathrm{Iter}_t] \leqslant t$ is immediate, since at most one learning iteration can occur per time slot. To obtain the lower bound, we analyze a virtual worst-case system in which at most one RW can be active RW at any time. Whenever a new RW would be created while another is active, the creation is suppressed; when no RW is active and a creation trial succeeds, a single RW is started. The expected number of local-update iterations in this restricted system therefore provides a lower bound on $\mathbb{E}[\mathrm{Iter}_t]$. Since $\mathrm{Iter}_t \leqslant C_t$, a lower bound on $\mathbb{E}[C_t]$ follows immediately. The full proof is provided in Appendix F (see the Supplementary Material). □

Remark 7 (Communication–convergence tradeoff): The lower bounds in (14) and (15) reflect the role of the regeneration parameters and partially reveal a communication–convergence tradeoff. After a temporary extinction, RW regeneration is governed by the threshold A and the creation probability q. A larger A or a smaller q makes regeneration more conservative, which lengthens the expected inactive period and decreases the lower bounds on both the effective update ratio $\mathbb{E}[\mathrm{Iter}_t]/t$ and the normalized communication cost $\mathbb{E}[C_t]/t$. Conversely, a smaller A or a larger q leads to more aggressive regeneration, which increases the fraction of time slots used for learning but also increases the communication cost. Thus, aggressive regeneration improves real-time learning progress at the expense of higher communication overhead, whereas conservative regeneration reduces communication but slows down real-time convergence.

Remark 8 (Selection of the creation threshold A): A complete design rule for choosing A remains an open problem. The optimal choice of A may depend on the interaction among RW termination, recovery time, communication cost, and real-time convergence progress. Thus, it is intractable to analyze the optimal A directly. In practice, A may be tuned adaptively by monitoring empirical observations. Such an online procedure does not give a closed-form optimal threshold, but it may provide an acceptable data-driven choice of A. We leave a rigorous adaptive selection rule for A as future work.

Remark 9: Equation (14) implies that although the extinction events slow down convergence, the delay (the length of horizontal segments) grows at most linearly with t. This means that RW-SGD performs at least a linear number of updates with respect to t, and therefore will eventually converge given a long time horizon.

VII. Simulations

In this section, we present simulation results to validate the theoretical findings in Sections V and VI.

A. Simulation Setup

1) Graph Settings: We consider 4 connected graphs of 100 nodes, with 1 Pac-Man node and 99 benign nodes, including: (i) a regular graph with degree $d = 99$ (complete graph), (ii) a random regular graph with degree $d = 8$ (expander graph [46]), (iii) a regular graph with degree $d = 2$ (ring topology), and (iv) an Erdős–Rényi graph with the edge probability $p = 0.1$. The target sampling distribution is set to be uniform, i.e., $\pi = \left(\frac{1}{100}, \frac{1}{100}, \cdots, \frac{1}{100}\right)$, and the transition matrix P is obtained by the Metropolis-Hastings algorithm [41]. We assume $A_u = A$ for $u \in \mathcal{B}$. We set both the forking and termination probabilities to 1, i.e., $q = \zeta = 1$. The initial number of RWs $z_0 = 10$.

2) Learning Settings: For the distributed learning problem defined in (1), we evaluate our approach on both synthetic and public benchmark datasets.

Synthetic dataset. We consider a decentralized linear regression task, where each node u minimizes a local mean squared error (MSE) loss of the form:

$$f_u(\mathbf{x}) = (\mathbf{w}^T\mathbf{x} + b - y_u)^2,$$

where $\mathbf{x}$ is the input feature, y_u is the target label at node u, $\mathbf{w}$ is the weight vector, and b is the bias. We assume each node

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

holds only a single data point. This local objective $f_u(\mathbf{x})$ is strongly convex and L-smooth.

Public benchmark dataset. We use the standard MNIST handwritten digit dataset [47]. The dataset is evenly partitioned into 100 disjoint subsets, with each node is assigned a unique subset. We consider both *i.i.d.* and non-*i.i.d.* data partitioning schemes for distributing data across network nodes. In the *i.i.d.* case, the dataset is uniformly and independently divided into 100 disjoint subsets, one for each node. In the non-*i.i.d.* case, data heterogeneity is introduced by sampling from a Dirichlet distribution [48]: The concentration parameter α controls the degree of heterogeneity: as $\alpha \to 10$, the partitioning approaches the *i.i.d.* case; as $\alpha \to 0$, the data becomes highly non-*i.i.d.* Throughout this paper, we set $\alpha = 0.5$, which corresponds to a moderate level of non-*i.i.d.* partitioning. Under both partitioning schemes, whenever a RW visits a node, it uniformly samples a mini-batch of size $B = 256$ from the node's local data to perform a training update. Each node u minimizes the empirical cross-entropy loss over its local dataset $\mathcal{D}_u$:

$$f_u(w) = \frac{1}{|\mathcal{D}_u|} \sum_{(\mathbf{x},y)\in\mathcal{D}_u} \ell(w;\mathbf{x},y),$$

where w denotes the model parameters, $\ell(w;\mathbf{x},y)$ is the standard cross-entropy loss function, and $\mathcal{D}_u$ is the local subset assigned to node u. We adopt the Adam optimizer from the PyTorch library (`torch.optim.Adam`) for model training.

3) Baseline: To the best of our knowledge, the algorithm most closely related to our work is the DECAFORK algorithm [15], [25]. DECAFORK is a duplication-based algorithm, whereas our proposed algorithm is creation-based.[8]

Another baseline we consider is the classical GOSSIP-BASED SGD [3], [4]. In this scheme, each node transmits its locally updated model to all of its neighbors at every iteration, and model parameters are updated via neighborhood averaging. We incorporate the same adversarial setting as before: the Pac-Man node independently terminates each incoming model update with the termination probability ζ. As a result, the Pac-Man node is unable to reliably incorporate all information from its neighbors, leading to biased and incomplete aggregation.

Fig. 4 compares CIL and GOSSIP-BASED SGD in terms of communication overhead, where the communication overhead is measured by the number of transmissions. The loss curve of CIL (blue curve) decays much faster than that of GOSSIP-BASED SGD (orange curve), showing that CIL achieves faster convergence under the same level of communication overhead. In this experiment, we set $\zeta = 0.8 < 1$ and choose the creation threshold as $A = 55$, and we observe that the number of RWs fluctuates around the initial value $z_0 = 10$. The cumulative communication overhead up to time t is calculated by $\sum_{\tau\le t} Z_\tau$.[9] In contrast, for GOSSIP-BASED SGD, the number of communications (on a complete graph) at each time step is fixed and equals $N(N-1)$. Thus, the horizontal axis in Fig. 4 represents the cumulative communication overhead measured in transmissions. The results suggest that GOSSIP-BASED SGD requires substantially more transmissions to achieve the same loss level, whereas CIL can preserve and propagate informative updates more efficiently through active RWs.

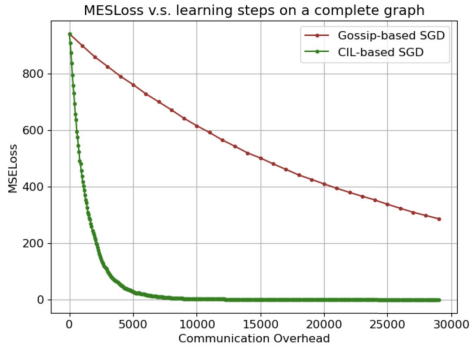


Fig. 4. Loss v.s. learning steps on a complete graph: comparison between CIL-based and gossip-based SGD.

Due to its slow convergence, we find GOSSIP-BASED SGD to be an unsuitable baseline in the Pac-Man setting, and we therefore exclude it from the remaining simulation results.

B. Boundedness

In the beginning of the subsection, we validate Theorem 1, which asserts that the number of RWs remains bounded under the CIL algorithm. As shown in Fig. 5, the RW population process $\{Z_t\}_{t\in\mathbb{Z}}$ remains bounded on all tested graphs. The y-axis represents the number of RWs, and the x-axis denotes the time steps. We present a single realization of $\{Z_t\}_{t\in\mathbb{Z}}$. On each graph, when the threshold A is small (noting that the definitions of "small" and "large" vary by graph), the RW population fluctuates at a relatively high level region. In contrast, for large A, the RW population may temporarily go extinct but subsequently recovers. In all cases, the population remains bounded, confirming the theoretical analysis.

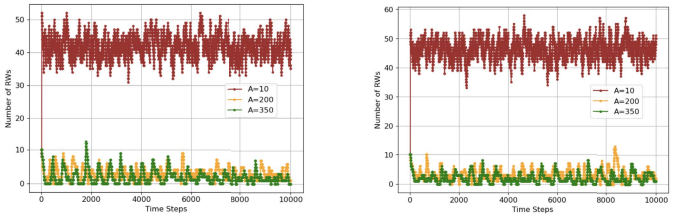
Next, we fix the creation threshold $A = 10$ and vary the termination probability ζ instead of fixing it at 1. In Fig. 6, we set $\zeta \in \{0.1, 0.01, 0.001\}$ and present a *single* realization of the RW population process $\{Z_t\}_{t\in\mathbb{Z}}$. From the figure, we observe that the population size remains bounded across different graph types, even when the termination probability ζ is very small. This observation is consistent with Theorem 1.

The estimated $\bar{Z}^\star$ and the upper bound from Theorem 2 are illustrated in Table I and Fig. 7. Although Theorem 2 formally applies only to the complete graph, we additionally evaluate $\bar{Z}^\star$ numerically on other graphs with small diameter, including regular and Erdős–Rényi graphs. Ring graphs are excluded from this experiment because their creation probability q requires a different scaling regime and is therefore not directly comparable.

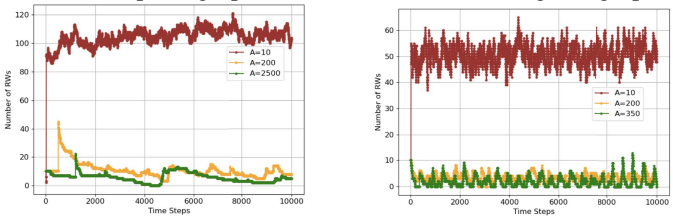
[8]Another duplication-based algorithm introduced in [15] and [25] is the MISSINGPERSON algorithm, which has been less explored in the literature. We do not include the MISSINGPERSON algorithm as a baseline in this paper for two reasons: (i) the primary difference between DECAFORK and MISSINGPERSON lies in their duplication mechanisms, and DECAFORK has been shown to be more robust than MISSINGPERSON in [15] and [25]; (ii) when applied to decentralized learning settings, both algorithms exhibit similar convergence performance, as reported in [26].

[9]In addition, as discussed in Fig. 7, the creation probability q can be appropriately chosen to keep the RW population small, thereby reducing the communication overhead.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

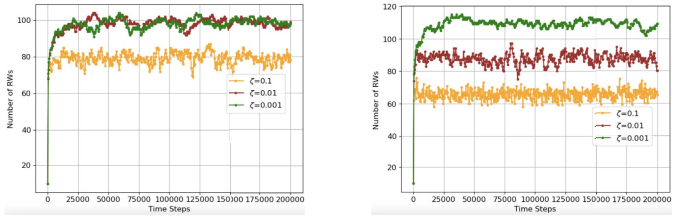


(a) complete graph (b) random regular graph

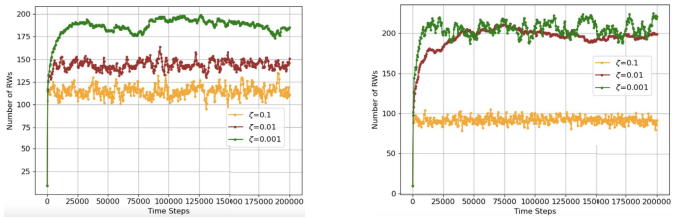


(c) ring topology (d) Erdős–Rényi graph

Fig. 5. Number of RWs over time on different graphs.



(a) complete graph (b) random regular graph



(c) ring topology (d) Erdős–Rényi graph

Fig. 6. Number of RWs over time on different graphs when $\zeta \in \{0.1, 0.01, 0.001\}$.

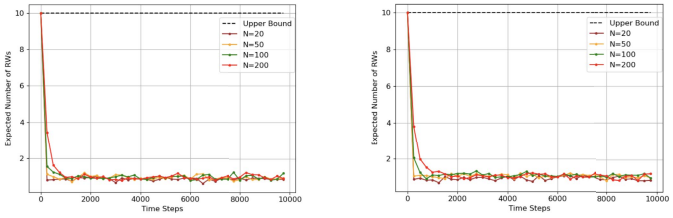
TABLE I

THE EXPECTED PEAK NUMBER OF RWS, $\bar{Z}^\star$, EVALUATED ON COMPLETE, REGULAR, AND ERDŐS–RÉNYI GRAPHS

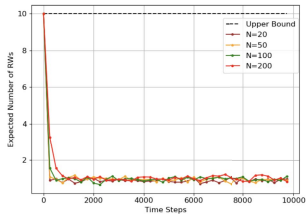
	$q = 1$	$q = \frac{1}{N}$	$q = \frac{1}{N^2}$
$\bar{Z}^\star$ in Complete	205.3	45.2	1.13
$\bar{Z}^\star$ in Regular	211.9	44.6	1.25
$\bar{Z}^\star$ in Erdős–Rényi	208.1	47.5	1.37
Upper Bound Theorem 2	$O(10^4)$	$O(10^2)$	$O(1)$

In Table I, we fix the initial population $z_0 = 1$, termination probability $\zeta = 1$, and creation threshold $A = 1$. For different choices of the creation probability q, the empirical $\bar{Z}^\star$ remains strictly below the theoretical upper bound, thereby validating Theorem 2.

Fig. 7 shows that under the scaling $q = \frac{1}{N^2}$ and with $z_0 = 10$, the expected number of RWs $\mathbb{E}[Z_t]$ (and consequently $\bar{Z}^\star$) on complete, regular, and Erdős–Rényi graphs is always below the bound in (10) for all tested network size $N \in \{20, 50, 100, 200\}$. This provides strong empirical evidence

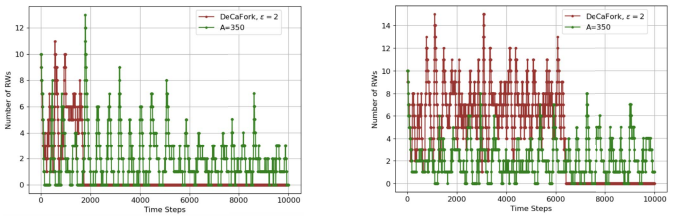


(a) complete graph (b) random regular graph

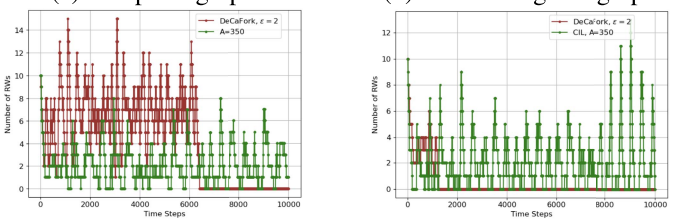


(c) Erdős–Rényi graph

Fig. 7. Expected number of RWs over time on different graphs under the scaling $q = \frac{1}{N^2}$ for $N \in \{20, 50, 100, 200\}$, compared with the upper bound in (10).



(a) complete graph (b) random regular graph



(c) ring topology (d) Erdős–Rényi graph

Fig. 8. Comparison of the number of RWs under the CIL and DECAFORK algorithms.

that the peak RW population can be made independent of the network size N even beyond the complete-graph setting. It is worth noting that the bound in (10) is inherently conservative, since it is derived under the worst-case choice $A_u = 1$ for all $u \in \mathcal{B}$, which maximizes the RW creation rate. Therefore, (10) serves as a universal upper bound for all threshold choices with $A_u > 1$, but may be loose for larger thresholds, where RW creation becomes less frequent.

In the end of this subsection, we present a comparison between the number of RWs generated by the CIL algorithm and by the baseline DECAFORK algorithm. To ensure a fair comparison, the two mechanisms adopt the same RW transition matrix P, so their movement behavior remains identical. The results are shown in Fig. 8. We observe that the CIL algorithm outperforms the DECAFORK algorithm in terms of robustness. Under the DECAFORK algorithm, the permanent extinction of the RW population is highly sensitive to the algorithm parameter ϵ. If ϵ is not chosen properly, the RW population eventually goes extinct (brown curves). In

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

contrast, under the CIL algorithm, the permanent extinction never occurs. Regardless of how large the creation threshold A is, if the population goes extinct, after a certain period of time, the RW population will always revive.

C. Convergence

We now evaluate the performance of the proposed CIL algorithm against the baseline DECAFORK. We begin with experiments on the synthetic dataset. As noted in Remark 1, we do not average the models across all benign nodes as gossip-based algorithms do [3]; we also do not average across all active RWs. Instead, we select *one* chain of RWs as defined in Definition 6 and take the expectation only over the intrinsic randomness of the algorithm, such as the stochastic-gradient noise and the RW dynamics. Accordingly, when plotting the loss curve, we randomly choose one chain of RWs and report its loss. We clarify that the loss curves in Figs. 9, 11, and 12 are used to compare the convergence behavior of a single chain of RWs. The communication overhead associated with each plotted loss curve is the overhead incurred along that particular chain of RWs. When the threshold A is small, more RWs may be generated on the graph, which can indeed increase the total communication overhead of the overall system. However, this additional graph-level overhead is not reflected in the per-chain loss curve.

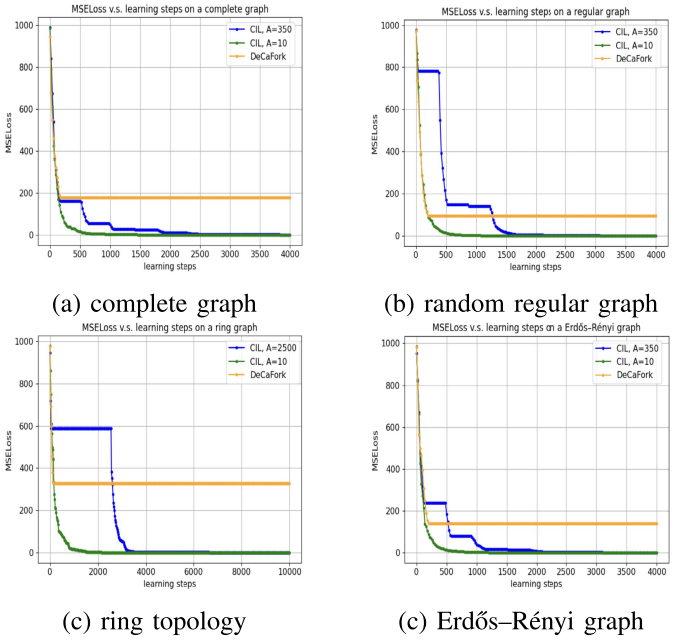
Fig. 9 presents the convergence performance of RW-SGD using the proposed CIL algorithm, compared with the DECAFORK baseline. In each sub-figure, the y-axis represents the value of the global loss (log scale), while the x-axis denotes the number of time steps. Across different graph topologies[10]:

(i) For the CIL with a small creation threshold (e.g, $A = 10$), the loss curve consistently decreases over time and eventually approach zero, indicating effective convergence. When $A = 10$, extinction events under the CIL algorithm occur only rarely, so with high probability the loss curve contains no flat segments. In this experiment, no extinction events happen.

(ii) For CIL with a large creation threshold (e.g, $A = 350$), the loss curve exhibits multiple horizontal segments (green curve). During each flat segment, the RW population becomes extinct, and the learning process remains inactive. After a certain time period, once at least one node has waited for A time slots since its last visit, new RWs are created and the learning process resumes.

(iii) As shown in Fig. 8, the robustness of the DECAFORK algorithm depends strongly on the choice of the parameter ϵ. The DECAFORK algorithm can delay RW extinction but does not completely prevent it. In contrast, CIL avoids permanent extinction by creating new RWs when needed, at the cost of additional communication overhead. We observe that, when ϵ is not chosen properly, i.e., $\epsilon = 3$, the RW population may eventually become extinct. In this case, no RW remains to carry out local updates, and the learning task fails to complete. It is worth noting that, based on the design of DECAFORK [15], [25], it has a positive probability of RW extinction. Therefore, regardless of how carefully the hyperparameter ϵ is chosen, there always remains a nonzero risk that all RWs become extinct, thereby terminating the training process.

[10]In the ring topology, we set the initial number of RWs to $z_0 = 1$ instead of $z_0 = 10$. This is because an RW is much less likely to hit the Pac-Man node in a ring topology than in complete, regular, and Erdős–Rényi graphs; using a smaller initial RW population makes the impact of the Pac-Man attack more visible.



(a) complete graph (b) random regular graph

(c) ring topology (c) Erdős–Rényi graph

Fig. 9. Loss function v.s. learning steps on different graphs.

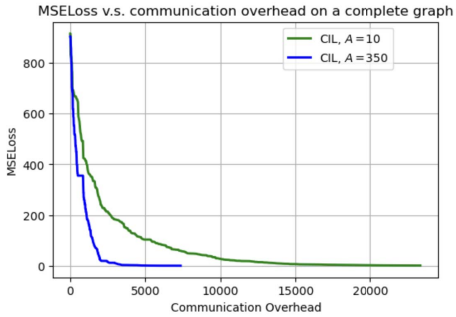
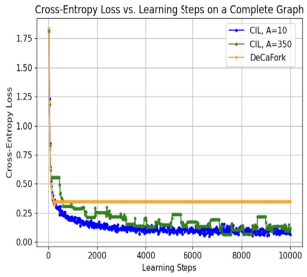


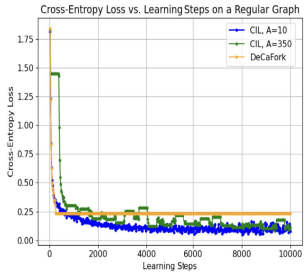
Fig. 10. Loss v.s. communication overhead for the CIL algorithm with different thresholds A on a complete graph.

From Fig. 9, one might conclude that the CIL algorithm performs better with a smaller threshold A, since it converges faster in terms of time steps. This observation may suggest choosing a very small threshold, such as $A = 1$, to accelerate the learning process. However, a smaller threshold can result in substantially higher communication overhead. As shown in Fig. 10, to achieve the same loss level, the CIL algorithm with $A = 10$ incurs greater communication overhead than that with $A = 350$. This observation illustrates the tradeoff between convergence speed and communication overhead discussed in Proposition 2 and Remark 7. Although Fig. 10 presents results only for the complete graph, similar phenomena are also observed for regular, ring, and Erdős–Rényi graphs.

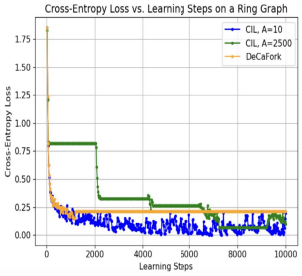
Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.



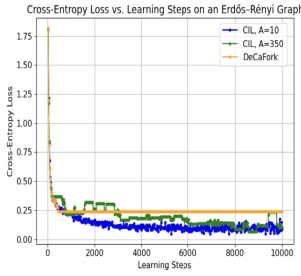
(a) complete graph



(b) random regular graph

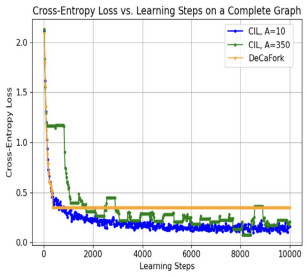


(c) ring topology

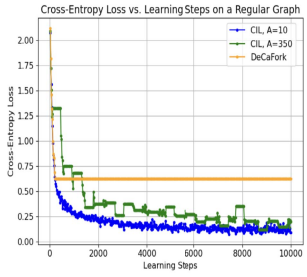


(c) Erdős–Rényi graph

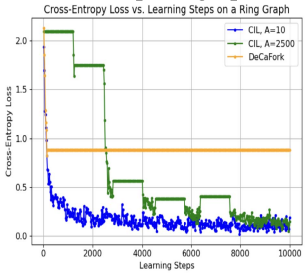
Fig. 11. Loss function v.s. learning steps on different graphs under *i.i.d.* partitioning of the public benchmark dataset.



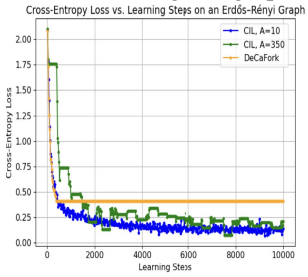
(a) complete graph



(b) random regular graph



(c) ring topology



(c) Erdős–Rényi graph

Fig. 12. Loss function v.s. learning steps on different graphs under non-*i.i.d.* partitioning of the public benchmark dataset.

TABLE II

$\|\tilde{\mathbf{x}}^* - \mathbf{x}^*\|$ AND ITS BOUNDS ON DIFFERENT GRAPHS

graph type	Complete	Regular	Ring	Erdős–Rényi
$\frac{1}{L}\|\nabla f(\tilde{\mathbf{x}}^\star)\|$	0.018	0.013	0.042	0.015
$\|\tilde{\mathbf{x}}^* - \mathbf{x}^*\|$	0.024	0.018	0.051	0.022
$\frac{1}{\mu}\|\nabla f(\tilde{\mathbf{x}}^\star)\|$	0.041	0.033	0.139	0.037

TABLE III

TESTING ACCURACIES ON DIFFERENT GRAPHS UNDER I.I.D. PARTITIONING

	Complete	Regular	Ring	Erdős–Rényi
Standard SGD	0.9727	0.9784	0.9715	0.9763
CIL, small A	0.9749	0.9792	0.9736	0.9771
CIL, large A	0.9758	0.9720	0.9767	0.9713
DeCaFork	0.9795	0.9742	0.9776	0.9731

The distance between the new convergence point and the original optimal solution, along with the corresponding theoretical bounds, is reported in Table II. This validates the theoretical guarantees established in Proposition 1 (1).

We then conducted experiments on the real-world dataset, considering both *i.i.d.* partitioning and non-*i.i.d.* partitioning scenarios.

Fig. 11 and Fig. 12 show the convergence behaviors of RW-SGD under the CIL with different creation thresholds ($A = 10$ and $A = 350$) and the DECAFORK, on the public benchmark dataset [47], using *i.i.d.* and non-*i.i.d.* data partitioning across nodes, respectively. In each figure, the subplots depict the global loss (on the y-axis) over time steps (on the x-axis). A moving average with a window size of 20 is applied to smooth the curves and mitigate the variances, to better reveal the underlying trends. Across both partitioning settings, the convergence patterns are consistent with those observed on synthetic dataset. The loss curve under CIL with $A = 10$ decreases quickly to near-zero levels and remain stable thereafter, confirming that the algorithm converges. The global loss under CIL with $A = 350$ also converges, but contains several horizontal segments and proceeds at a slower rate. Under the DECAFORK algorithm, the RW population may eventually become extinct, in which case no RW remains to perform local updates and the training process terminates prematurely.

Finally, we evaluate the learning performance by using testing accuracies of the final model. Given the target distribution $\pi = \left(\frac{1}{100}, \ldots, \frac{1}{100}\right)$, Metropolis-Hastings algorithm [41] implies that $\nu^{(1)}$ is uniform over the 99 benign nodes in complete, random regular, and ring graphs, and approximately uniform in the Erdős–Rényi graph due to its near-regular structure. This indicates that in the presence of a Pac-Man node, active RWs visit benign nodes uniformly or near-uniformly.

Table III presents the performances of final models under *i.i.d.* data partitioning. In the "Standard SGD" baseline, a single RW performs standard SGD on a graph without a Pac-Man node. From this table, we observe that the performances are nearly identical across all graph types, regardless of whether a Pac-Man node is present. This is because, although a Pac-Man node is present, the benign nodes share the same data distribution (due to the *i.i.d.* partitioning) and are accessed uniformly (or approximately so) by active RWs. As a result, the active RWs observe an approximately representative sample of the global data distribution, and the final models still perform well.

Table IV shows the results under non-*i.i.d.* partitioning. Here, the presence of a Pac-Man node impacts performance. Two trends emerge: (i) the performance in the "Standard SGD" baseline outperforms those in the Pac-Man cases, as the RW in the baseline has full access to all nodes and thus to the complete data distribution; in contrast, the active

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
TESTING ACCURACIES ON DIFFERENT GRAPHS UNDER NON- I.I.D. PARTITIONING

	Complete	Regular	Ring	Erdős–Rényi
Standard SGD	0.9738	0.9772	0.9729	0.9794
CIL, small A	0.9659	0.9661	0.9683	0.9542
CIL, large A	0.9677	0.9634	0.9612	0.9661
DeCaFork	0.9655	0.9613	0.9522	0.9638

RWs in the Pac-Man cases never access the Pac-Man node, and the remaining benign nodes—now with heterogeneous data (due to the non-*i.i.d.* partitioning)—no longer provide a representative sample of the entire dataset, resulting in degraded performance. (ii) The performances in the Pac-Man cases (under the CIL and DECAFORK algorithms) are nearly identical across all graph types. This is because their active RWs achieve similar quasi-stationary distributions. As a result, they sample similarly skewed subsets of data, which leads to a similar performances of the final models.

VIII. CONCLUSION

In this work, we focused on RW-based decentralized learning algorithms, which are susceptible to terminations and Pac-Man like attack. This stealthy and highly disruptive threat can gradually degrade the decentralized operations without triggering any detectable failure signals. We proposed a new creation based algorithm CIL that maintains RW population, and thereby improving the resilience of RW-based SGD under termination-based Pac-Man attacks. The proposed CIL algorithm is a fully decentralized duplication mechanism based solely on local visitation intervals and hence does not need to depend on system parameter estimation. We provided a rigorous analysis of the performance of this algorithm, highlighting its strengths. Extensive experiments on synthetic and public benchmark datasets validate our findings.

An interesting future direction is to develop principled methods for selecting the creation threshold A and characterizing the fundamental trade-offs among communication cost, recovery time, and convergence speed. Another interesting future direction is to establish tighter graph-dependent bounds on the peak RW population and better understand the role of graph topology in RW population dynamics. In addition, designing random-walk-based decentralized learning algorithms under combined threat models, where Pac-Man termination attacks coexist with other adversarial behaviors such as Byzantine attacks is a promising avenue for future work.

ACKNOWLEDGMENT

Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies.

REFERENCES

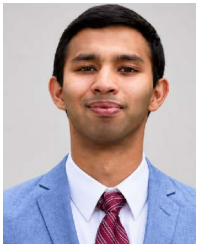
[1] L. Lovász, "Random walks on graphs," *Combinatorics*, vol. 2, pp. 1–46, Jun. 1993.
[2] D. A. Levin and Y. Peres, *Markov Chains and Mixing Times*, 2nd ed. Providence, RI, USA: American Mathematical Society, 2017.
[3] J. Tsitsiklis and D. Bertsekas, "Distributed asynchronous optimal routing in data networks," *IEEE Trans. Autom. Control*, vol. AC-31, no. 4, pp. 325–332, Apr. 1986.
[4] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized SGD with changing topology and local updates," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, Jul. 2020, pp. 5381–5393.
[5] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the Web," *Stanford InfoLab*, Stanford, CA, USA, Tech. Rep. 1999-66, 1999.
[6] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 355–368, Mar. 2007.
[7] Y. Liu, S. Ji, and P. Mittal, "SmartWalk: Enhancing social network security via adaptive random walks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 492–503.
[8] L. Backstrom and J. Leskovec, "Supervised random walks: Predicting and recommending links in social networks," in *Proc. 4th ACM Int. Conf. Web search data mining*, Feb. 2011, pp. 635–644.
[9] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 793–803.
[10] B. Johansson, M. Rabi, and M. Johansson, "A simple peer-to-peer algorithm for distributed optimization in sensor networks," in *Proc. 46th IEEE Conf. Decis. Control*, Dec. 2007, pp. 4705–4710.
[11] G. Ayache, V. Dassari, and S. E. Rouayheb, "Walk for learning: A random walk approach for federated learning from heterogeneous data," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 929–940, Apr. 2023.
[12] Y. Lai et al., "Coupled-space attacks against random-walk-based anomaly detection," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 9315–9329, 2024.
[13] X. Zhang, H. Xie, P. Yi, and J. C. S. Lui, "Enhancing Sybil detection via social-activity networks: A random walk approach," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1213–1227, Mar. 2023.
[14] H. Kawamura, S. Shiina, H. N. Aung, and H. Ohsaki, "Robustness of random walk on a graph against adversary attacks," in *Proc. IEEE 48th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2024, pp. 1080–1088.
[15] M. Egger, G. Ayache, R. Bitar, A. Wachter-Zeh, and S. E. Rouayheb, "Self-duplicating random walks for resilient decentralized learning on graphs," in *Proc. GLOBECOM - IEEE Global Commun. Conf.*, Dec. 2024, pp. 2960–2965.
[16] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Comput. Surveys*, vol. 22, no. 4, pp. 299–319, Dec. 1990.
[17] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998.
[18] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[19] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Comput. Surveys*, vol. 34, no. 3, pp. 375–408, Sep. 2002.
[20] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, no. 2, pp. 225–267, Mar. 1996.
[21] J. Walrand and P. Varaiya, *High-Performance Communication Networks*, 2nd ed., San Mateo, CA, USA: Morgan Kaufmann, 1999.
[22] T. E. Harris, "The theory of branching processes," *RAND Corp.*, Monica, CA, USA, Tech. Rep. R-381-PR, 1964.
[23] T. Sun, Y. Sun, and W. Yin, "On Markov chain gradient descent," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 9918–9927.
[24] X. Mao, K. Yuan, Y. Hu, Y. Gu, A. H. Sayed, and W. Yin, "Walkman: A communication-efficient random-walk algorithm for decentralized optimization," *IEEE Trans. Signal Process.*, vol. 68, pp. 2513–2528, 2020.
[25] M. Egger, R. Bitar, G. Ayache, A. Wachter-Zeh, and S. El Rouayheb, "Self-regulating random walks for resilient decentralized learning on graphs," *IEEE Trans. Inf. Theory*, vol. 72, no. 9, pp. 6864–6878, Sep. 2026.
[26] X. Chen, P. Parag, R. Bhagat, Z. Liu, and S. El Rouayheb, "Random walk learning and the Pac-Man attack," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2026, pp. 1–6.
[27] J. Sen, M. G. Chandra, P. Balamuralidhar, S. G. Harihara, and H. Reddy, "A distributed protocol for detection of packet dropping attack in mobile Ad hoc networks," in *Proc. IEEE Int. Conf. Telecommun. Malaysia Int. Conf. Commun.*, May 2007, pp. 75–80.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.

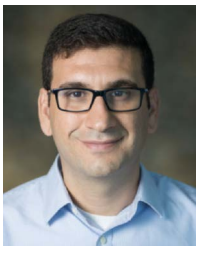
[28] S. Pal, B. Sikdar, and J. Chow, "Real-time detection of packet drop attacks on synchrophasor data," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Nov. 2014, pp. 896–901.
[29] K. Edemacu, M. Euku, and R. Ssekibuule, "Packet drop attack detection techniques in wireless Ad hoc networks: A review," *Int. J. Netw. Secur. Appl.*, vol. 6, no. 5, pp. 75–86, Sep. 2014.
[30] L. Palmieri, C. Boldrini, L. Valerio, A. Passarella, M. Conti, and J. Kertész, "Robustness of decentralised learning to nodes and data disruption," *Comput. Commun.*, vol. 241, Sep. 2025, Art. no. 108250.
[31] M. Chujyo, S. Liu, and F. Toriumi, "Impact of seed node position on network robustness under localized attacks," *PLoS ONE*, vol. 21, no. 9, Sep. 2026, Art. no. e0358769.
[32] Y. Berezin, A. Bashan, M. M. Danziger, D. Li, and S. Havlin, "Localized attacks on spatially embedded networks with dependencies," *Sci. Rep.*, vol. 5, no. 1, p. 8934, Mar. 2015.
[33] Y. Yakimenka, C.-W. Weng, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Straggler-resilient differentially private decentralized learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 5, pp. 407–423, 2024.
[34] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari, and R. Pedarsani, "Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity," *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 197–205, Jun. 2022.
[35] I. Tziotis, Z. Shen, R. Pedarsani, H. Hassani, and A. Mokhtari, "Straggler-resilient personalized federated learning," *Trans. Mach. Learn. Res.*, Oct. 2023.
[36] G. Xiong, G. Yan, S. Wang, and J. Li, "Straggler-resilient decentralized learning via adaptive asynchronous updates," in *Proc. 25th Int. Symp. Theory, Algorithmic Found., Protocol Design Mobile Netw. Mobile Comput.*, Oct. 2024, pp. 434–439.
[37] J. He, M. Xiao, M. Skoglund, and H. V. Poor, "Straggler-resilient asynchronous ADMM for distributed consensus optimization," *IEEE Trans. Signal Process.*, vol. 73, pp. 2496–2510, 2025.
[38] D. Liang, J. Zhang, E. Chen, Z. Li, R. Li, and H. Yang, "Towards straggler-resilient split federated learning: An unbalanced update approach," in *Proc. Adv. Neural Inf. Process. Syst.*, 2025, pp. 35702–35741.
[39] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*. Reading, MA, USA: Addison-Wesley, 2011.
[40] M. Even, "Stochastic gradient descent under Markovian sampling schemes," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 202, 2023, pp. 9412–9439.
[41] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM J. Optim.*, vol. 20, no. 3, pp. 1157–1170, Jan. 2010, doi: 10.1137/08073038x.
[42] P. Collet, S. Martínez, and J. S. Martín, *Quasi-Stationary Distributions: Markov Chains, Diffusions and Dynamical Systems (Probability and Its Applications)*. Cham, Switzerland: Springer, 2012.
[43] J. N. Darroch and E. Seneta, "On quasi-stationary distributions in absorbing discrete-time finite Markov chains," *J. Appl. Probab.*, vol. 2, no. 1, pp. 88–100, Jun. 1965.
[44] H. Kahn and A. W. Marshall, "Methods of reducing sample size in Monte Carlo computations," *J. Operations Res. Soc. Amer.*, vol. 1, no. 5, pp. 263–278, Nov. 1953.
[45] D. G. Horvitz and D. J. Thompson, "A generalization of sampling without replacement from a finite universe," *J. Amer. Stat. Assoc.*, vol. 47, no. 260, pp. 663–685, Dec. 1952.
[46] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, no. 4, pp. 439–561, Oct. 2006.
[47] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[48] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.
[49] R. Durrett, *Probability: Theory and Examples*, 5th ed. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[50] R. D. Foley and D. R. McDonald, "Yaglom limits can depend on the starting state," *J. Appl. Probab.*, vol. 54, no. 3, pp. 726–734, 2017.
[51] P. Billingsley, *Probability and Measure* (Wiley Series in Probability and Mathematical Statistics), 3rd ed., Hoboken, NJ, USA: Wiley, 1995.
[52] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[53] Z. Liu, S. El Rouayheb, and M. Dwyer, "The entrapment problem in random walk decentralized learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2024, pp. 3660–3665.

Xingran Chen (Member, IEEE) received the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania. He is currently an tenure-track Assistant Professor with the Engineering Systems and Design Pillar, Singapore University of Technology and Design (SUTD). Prior to joining SUTD, he was a Post-Doctoral with Rutgers University and an Assistant Professor with the University of Electronic Science and Technology of China. His research focuses on fundamental limits and algorithm design for decentralized learning and decision-making in networked systems. He has served the research community as an Advisory TPC Member for IEEE INFOCOM 2027 and a TPC Member for IEEE ICC 2027, IEEE ITW 2027, WiOpt 2026, IEEE ICC 2024, and IEEE GLOBECOM Workshop 2020. He received the IEEE Communications Society and Information Theory Society Joint Paper Award in 2023.

Parimal Parag (Senior Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from IIT Madras in 2004, and the Ph.D. degree in electrical engineering from Texas A&M University in 2011. He has been an Associate Professor with the Department of Electrical Communication Engineering, Indian Institute of Science, since December 2014. Prior to that, he was a Senior System Engineer (R&D) with Assia Inc., Redwood City, from 2011 to 2014. He is the co-author of the 2018 IEEE ISIT Student Best Paper. His research interests include the design and analysis of large scale distributed systems. He was a recipient of the 2017 Early Career Award from the Science and Engineering Research Board.



Rohit Bhagat (Student Member, IEEE) is currently pursuing the double bachelor's degrees in electrical and computer engineering and in computer science with Rutgers University, New Brunswick. His research interests include federated machine learning, distributed optimization, and adversarial robustness, with a focus on implementing algorithms for robust and scalable decentralized learning systems.



Salim El Rouayheb (Senior Member, IEEE) received the Diploma degree in electrical engineering from Lebanese University, Lebanon, in 2002, the M.S. degree from American University of Beirut, Lebanon, in 2004, and the Ph.D. degree in electrical engineering from Texas A&M University, College Station, TX, USA, in 2009.

He was a Post-Doctoral Research Fellow with the University of California, Berkeley, from 2010 to 2011, and a Research Scholar with Princeton University from 2012 to 2013. From 2013 to 2017, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ, USA, where he held the Walter Tyson Junior Faculty Chair in 2019. His research interests include the intersection of information theory, security, privacy, and distributed computing, with a particular emphasis on information-theoretic privacy and security, coding for distributed systems, and resilient decentralized and federated learning.

Prof. Rouayheb received the NSF CAREER Award in 2016 and the Google Faculty Award in 2018.

Authorized licensed use limited to: J.R.D. Tata Memorial Library Indian Institute of Science Bengaluru. Downloaded on November 08,2026 at 04:37:16 UTC from IEEE Xplore. Restrictions apply.