

DOIN: A Game-Theoretic Framework for Decentralized Multi-Model Optimization via Proof of Optimization

Harvey Bastidas

School of Systems and Computer Engineering
Universidad del Valle, Cali, Colombia
harvey.bastidas@correounivalle.edu.co

Abstract

We present DOIN, a Decentralized Optimization and Inference Network that repurposes blockchain consensus for verified machine learning optimization. Rather than expending computational resources on cryptographic puzzles as in Proof of Work (PoW), DOIN introduces *Proof of Optimization* (PoO), a novel consensus mechanism in which block generation is triggered by verified performance improvements across multiple simultaneous optimization domains. We formalize the system model, define a multi-domain weighting scheme called Verified Useful Work (VUW), and prove that honest optimization constitutes a Nash equilibrium under our incentive structure. A commit-reveal protocol with quorum-based verification using per-evaluator synthetic data provides robustness against front-running, parameter theft, and collusion attacks. We identify 25 attack vectors and implement 10 hardening measures, achieving security guarantees comparable to established blockchain systems while converting all consensus computation into productive ML optimization work. The reference implementation comprises five Python packages with 291 tests and a plugin architecture for extensibility.

Index Terms—decentralized optimization, blockchain consensus, proof of useful work, federated learning, game theory, machine learning, Nash equilibrium

I. INTRODUCTION

The training and optimization of machine learning models has become one of the most computationally intensive activities in modern computing. Large-scale optimization is typically performed in centralized data centers operated by a small number of organizations, creating bottlenecks in cost, access, and resilience [1]. Simultaneously, blockchain networks such as Bitcoin [2] collectively expend enormous computational resources—estimated at over 150 TWh annually—on cryptographic puzzles that produce no direct scientific or economic value beyond consensus security.

This paper addresses a natural question: *can the computational work required for blockchain consensus be redirected toward useful machine learning optimization?* We answer affirmatively by introducing the Decentralized Optimization and Inference Network (DOIN), a system that replaces Proof of Work with *Proof of Optimization* (PoO). In DOIN, nodes earn the right to propose blocks by demonstrating verified improvements to ML models across one or more optimization domains.

The key challenge in designing such a system is ensuring that claimed optimization improvements are genuine. Unlike cryptographic hash puzzles, where verification is trivial, verifying ML optimization quality requires evaluating model performance—a process that is itself computationally expensive and susceptible to manipulation. DOIN addresses this through a combination of commit-reveal protocols,

quorum-based verification with per-evaluator synthetic data, and an asymmetric reputation system that makes dishonesty a dominated strategy.

Our contributions are as follows:

- A formal system model for decentralized multi-domain ML optimization with configurable node roles (Section III).
- The Proof of Optimization consensus mechanism with Verified Useful Work (VUW) weighting and dynamic threshold adjustment (Section IV).
- A game-theoretic analysis proving that honest optimization is a Nash equilibrium under the DOIN incentive structure (Section V).
- A comprehensive security analysis covering 25 attack vectors with 10 implemented hardening measures (Section VI).
- A reference implementation with 291 tests demonstrating practical feasibility (Section VII).

II. RELATED WORK

A. Federated Learning

Federated learning (FL), introduced by McMahan et al. [3], enables distributed model training while preserving data locality. FedAvg and its variants [4] aggregate locally computed gradients at a central server. However, FL retains a centralized coordinator that represents a single point of failure

and trust. DOIN eliminates this dependency by replacing the central aggregator with blockchain-based consensus, making coordination fully decentralized.

B. Blockchain-Based Machine Learning

Several systems have explored combining blockchain with machine learning. FLChain [5] places federated learning on a blockchain but retains the FL aggregation model. DInEMMo [6] proposes decentralized inference but does not address optimization consensus. Ocean Protocol [7] decentralizes data marketplaces but not the optimization process itself. Unlike these systems, DOIN makes ML optimization the *consensus mechanism itself*, not merely an application running atop a blockchain.

C. Proof of Useful Work

The concept of replacing hash-based PoW with scientifically useful computation has been explored in several contexts. Primecoin [8] searches for prime number chains, while Gridcoin [9] rewards BOINC scientific computing contributions. Ball et al. [10] formalized proof of useful work theoretically. These systems, however, either use trivially verifiable problems or rely on external trust anchors. DOIN advances the field by providing a verification mechanism based on per-evaluator synthetic data generation that is both rigorous and domain-agnostic.

D. Decentralized Optimization

Distributed optimization has a rich history in the operations research and control communities [11]. ADMM-based distributed methods [12] and gossip-based protocols [13] enable optimization without a central coordinator. However, these methods assume cooperative participants. DOIN extends decentralized optimization to adversarial settings using game-theoretic incentive design.

III. SYSTEM MODEL

A. Network and Domains

We define the DOIN network as a tuple $(\mathcal{N}, \mathcal{D}, \mathcal{C})$ where:

- $\mathcal{N} = \{n_1, \dots, n_{|\mathcal{N}|}\}$: is the set of participating nodes
- $\mathcal{D} = \{d_1, \dots, d_{|\mathcal{D}|}\}$: is the set of active optimization domains
- $\mathcal{C} = (B_0, B_1, \dots)$: is the blockchain, an ordered sequence of blocks

Each domain $d \in \mathcal{D}$ represents an optimization problem defined by a model architecture, a loss function, and a dataset specification. Domains are registered on-chain and may be created by any node willing to stake a minimum deposit.

An *optimum* (plural: *optimae*) is a tuple $o = (d, \theta, p, n, t)$ representing the best-known parameters θ for domain d with verified performance p , submitted by node n at time t . The chain maintains a registry of current optimae for all active domains.

B. Node Roles

Each node $n \in \mathcal{N}$ may assume one or more of three roles, configurable at runtime:

Optimizer. An optimizer node performs gradient-based or black-box optimization on one or more domains. It submits candidate parameter improvements along with performance claims to the network.

Evaluator. An evaluator verifies performance claims by independently evaluating submitted parameters against synthetic test data. Evaluators are selected randomly for each verification task.

Relay. A relay node participates in block propagation and chain maintenance without performing optimization or evaluation. Relays earn reduced rewards for network maintenance.

C. Task Queue Architecture

DOIN employs a pull-based task queue with strict priority ordering. Each node maintains a local queue that sources tasks from the network. The priority hierarchy is:

$$\text{Priority: verification} > \text{inference} > \text{optimization} \quad (1)$$

Verification tasks take precedence to ensure timely block production. This architecture ensures that the system remains responsive to consensus requirements while utilizing remaining capacity for productive optimization work.

IV. PROOF OF OPTIMIZATION CONSENSUS

A. Block Generation Threshold

A node may propose a new block when the weighted sum of its verified performance increments across all domains exceeds a dynamic threshold T . Formally, a node n qualifies to produce a block when:

$$\sum_{d \in \mathcal{D}} \text{eff}(n, d) \geq T$$

where $\text{eff}(n, d)$ is the effective increment contributed by node n to domain d , defined below. Crucially, the sum ranges over *all* active domains, allowing a node to accumulate increments across multiple optimization problems to meet the threshold.

B. Verified Useful Work (VUW) Weighting

Not all optimization domains contribute equally to consensus. The VUW scheme assigns dynamic weights to each domain based on network demand and optimization difficulty. For domain d , the weight is:

$$w(d) = w_{\text{base}}(d) \cdot \varphi_{\text{demand}}(d) \cdot (1 + \varphi_{\text{progress}}(d)) \cdot v(d) \quad (3)$$

where:

- $w_{\text{base}}(d) \in \mathbb{R}^+$ is a governance-assigned base weight reflecting the domain's intrinsic importance.

- $\varphi_{\text{demand}}(d) = q(d) / \bar{q}$ is the demand factor, the ratio of pending inference queries for domain d to the network average.
- $\varphi_{\text{progress}}(d)$ captures optimization difficulty: domains where progress is harder receive higher weight, incentivizing work on challenging problems.
- $v(d) \in \{0, 1\}$ is the verification strength: $v(d) = 1$ if and only if the domain supports synthetic data verification; otherwise $v(d) = 0$.

The binary verification strength is a critical design choice: *domains without synthetic data verification contribute zero weight to consensus*. This ensures that all optimization work counted toward block production is independently verifiable, preventing gaming through unverifiable claims.

C. Effective Increment

The effective increment for node n on domain d combines the raw performance improvement with domain weight and node reputation:

$$\text{eff}(n, d) = \Delta p(n, d) \cdot w(d) \cdot \log(1 + r(n)) / \log(1 + 10) \quad (4)$$

where $\Delta p(n, d)$ is the verified raw performance increment (difference between the submitted performance and the current best-known performance for domain d), $w(d)$ is the VUW domain weight from (3), and $r(n) \in [0, \infty)$ is the reputation score of node n .

The logarithmic reputation scaling serves two purposes. First, it prevents high-reputation nodes from dominating block production, as the marginal value of additional reputation diminishes. Second, the normalization by $\log(1 + 10)$ ensures that a node with reputation 10 (a reasonable established participant) contributes its raw weighted increment without attenuation.

D. Dynamic Threshold Adjustment

The threshold T is adjusted periodically to maintain a target block time τ_{target} , analogous to Bitcoin's difficulty adjustment. After each adjustment epoch of E blocks:

$$T_{\text{new}} = T_{\text{old}} \cdot (\tau_{\text{actual}} / \tau_{\text{target}})^{-1} \cdot \text{clamp}(0.25, 4.0) \quad (5)$$

where τ_{actual} is the mean observed block time over the epoch. The clamp factor bounds the adjustment to prevent instability from sudden changes in network optimization capacity. If blocks are produced too quickly (optimization is easy), the threshold increases; if too slowly, it decreases.

E. Fork Choice Rule

DOIN employs a *heaviest-chain* rule rather than the longest-chain rule used in Bitcoin. The chain weight is defined as the cumulative verified optimization work:

$$W(C) = \sum_{B \in C} \sum_{d \in D} \text{eff}(B, d)$$

This rule ensures that the canonical chain is the one containing the most verified optimization work, aligning the fork choice incentive with the system's primary objective of maximizing useful computation.

V. GAME-THEORETIC ANALYSIS

This section constitutes the core theoretical contribution of this paper. We model DOIN as a strategic game and prove that honest participation is a Nash equilibrium.

A. Game Formulation

Define the DOIN game as $G = (N, S, u)$ where N is the set of nodes (players), $S = S_1 \times \dots \times S_{|N|}$ is the joint strategy space, and $u: S \rightarrow \mathbb{R}^{|N|}$ is the payoff function. Each node n_i selects a strategy $s_i \in S_i = \{\text{honest, inflate, deflate, collude, free-ride}\}$.

B. Commit-Reveal Protocol

To prevent front-running and parameter theft, DOIN employs a two-phase commit-reveal protocol. In the commit phase, an optimizer publishes:

$$c = H(\theta \parallel \eta)$$

where θ represents the optimized parameters, η is a random nonce, H is a cryptographic hash function, and \parallel denotes concatenation. In the subsequent reveal phase, the optimizer publishes (θ, η) , and the network verifies that $H(\theta \parallel \eta) = c$.

This prevents attack vector #13 (front-running), where an adversary observing an optimization submission could rush to claim it, and attack vector #19 (parameter theft), where submitted parameters could be copied before attribution is established.

C. Quorum Verification with Per-Evaluator Synthetic Data

Upon a valid reveal, the protocol selects a quorum of K evaluators from the pool of M eligible evaluators using a deterministic but unpredictable selection function:

$$Q = \text{Select}_K(M, H(c \parallel h_{\text{tip}}))$$

where h_{tip} is the hash of the current chain tip at selection time. Each selected evaluator $e_j \in Q$ independently generates synthetic evaluation data from a deterministic seed:

$$\text{seed}_j = H(c \parallel d \parallel e_j \parallel h_{\text{tip}})$$

This design has three critical properties. First, each evaluator uses *different* synthetic data, preventing an optimizer from overfitting to a single test distribution. Second, the optimizer cannot predict the seeds at optimization time because it does not know (a) which evaluators will be selected, as this depends on the chain tip at selection time, and (b) the

chain tip at selection time, as it changes with each block. Third, the seeds are deterministic given the inputs, allowing any node to independently verify the evaluation results.

Theorem 1 (Quorum Unpredictability). *For an optimizer with fraction $f < 1/2$ of network evaluation capacity, the probability of predicting all K evaluation seeds is bounded by f^K .*

Proof. The seed for each evaluator depends on the evaluator's identity and the chain tip. Controlling the chain tip requires majority evaluation capacity. For $f < 1/2$, each evaluator selection is independent with probability at most f of being a colluding node. The probability that all K evaluators are controlled is f^K . For $K = 5$ and $f = 0.3$, this is $0.3^5 \approx 0.00243$. ■

D. Incentive Model

Let p_{claimed} denote the performance reported by the optimizer and p_{verified} the median performance measured by the evaluator quorum. Define the performance gap as $g = p_{\text{claimed}} - p_{\text{verified}}$. The reward fraction is a piecewise function:

$$\rho(g) = \begin{cases} \min(1 + |g| \cdot \beta, \rho_{\max}), & \text{if } g < 0 \\ 1.0, & \text{if } g = 0 \\ 1 - g/\varepsilon \cdot (1 - \rho_{\min}), & \text{if } 0 < g \leq \varepsilon \\ 0, & \text{if } g > \varepsilon \end{cases}$$

where β is the bonus scaling factor, ρ_{\max} is the maximum bonus multiplier (e.g., 1.1), ρ_{\min} is the minimum partial reward fraction (e.g., 0.1), and ε is the tolerance threshold. The capping of the bonus at ρ_{\max} is essential: it removes the incentive to deliberately under-report performance, as the bonus from doing so is bounded.

E. Reputation System

Each node n maintains a reputation score $r(n)$ that evolves via exponentially weighted moving average (EMA) with asymmetric updates:

$$r_{t+1}(n) = \alpha \cdot r_t(n) + (1 - \alpha) \cdot \delta(n, t)$$

where $\alpha = 2^{-1/H}$ with half-life $H = 1$ week (measured in blocks), and:

$$\delta(n, t) = \begin{cases} +R_{\text{reward}}, & \text{if verification succeeds} \\ -3 \cdot R_{\text{reward}}, & \text{if verification fails} \end{cases}$$

The 3:1 penalty-to-reward asymmetry ensures that a single failed verification erases the reputation gains from three successful ones. A node must maintain $r(n) \geq r_{\min}$ to

participate in consensus, creating a meaningful barrier to Sybil attacks: new identities start with zero reputation and cannot immediately influence the system.

The EMA decay with a one-week half-life serves an additional purpose: it prevents *reputation farming*, where a node builds reputation through honest behavior and then exploits it in a single large attack. Historical reputation contributions decay exponentially, so maintaining high reputation requires sustained honest participation.

F. Nash Equilibrium Analysis

Theorem 2 (Honest Optimization is a Nash Equilibrium). *Under the DOIN incentive structure with quorum size $K \geq 3$ and penalty ratio $\gamma = 3$, the strategy profile where all nodes play honest is a Nash equilibrium.*

Proof. We show that no unilateral deviation from honest is profitable.

Case 1: Deviation to inflate. A node reports $p_{\text{claimed}} > p_{\text{actual}}$. With probability $1 - f^K$ (near certainty for $K \geq 3$), at least one honest evaluator detects the discrepancy. If $g > \varepsilon$, the reward is zero and reputation decreases by $3R_{\text{reward}}$. Expected payoff:

$$u(\text{inflate}) = f^K \cdot R + (1 - f^K) \cdot (-3R) < 0 < R = u(\text{honest}) \quad (13)$$

Case 2: Deviation to deflate. A node reports $p_{\text{claimed}} < p_{\text{actual}}$ to collect the bonus. By equation (10), the bonus is capped at ρ_{\max} . For $\rho_{\max} = 1.1$:

$$u(\text{deflate}) \leq 1.1R$$

However, deflation reduces the effective increment by equation (4), requiring more optimization rounds to meet the block threshold. The net expected payoff per unit time is lower than honest reporting.

Case 3: Deviation to collude. Collusion requires controlling all K evaluators. By Theorem 1, this probability is f^K . For a coalition controlling fraction f of evaluators with $K = 5$:

(11)	f	0.1	0.2	0.3	
	f^5	10^{-5}	3.2×10^{-4}	2.4×10^{-3}	1.

The expected penalty from failed collusion dominates any potential gain.

Case 4: Deviation to free-ride. A free-riding node submits no optimization work. By equation (2), it cannot meet the block threshold and earns zero rewards. Thus $u(\text{free-ride}) = 0 < R = u(\text{honest})$.

Since no unilateral deviation is profitable, the all-honest strategy profile is a Nash equilibrium. ■

G. Payoff Matrix

Table II summarizes the expected payoffs for a single optimizer node against an honest network, assuming $K = 5$, $f = 0.1$, $\gamma = 3$, and $\rho_{\max} = 1.1$.

TABLE II: EXPECTED PAYOFF PER OPTIMIZATION ROUND

Strategy	Reward	Rep. Change	E[Payoff]
honest	1.0R	$+R_{\text{rep}}$	1.0R
inflate	≈ 0	$-3R_{\text{rep}}$	$-3R$
deflate	$\leq 1.1R$	$+R_{\text{rep}}$	$\leq 1.1R^*$
collude	≈ 0	$-3R_{\text{rep}}$	$-3R$
free-ride	0	0	0

*Net payoff per unit time is lower due to reduced effective increment.

VI. SECURITY ANALYSIS

We conducted a systematic threat analysis identifying 25 attack vectors organized by attack surface. Table III summarizes the 10 primary hardening measures and the attacks they mitigate.

A. Attack Taxonomy

The identified attacks span five categories:

- 1) *Consensus Attacks*: 51% attack, selfish mining, long-range rewrite, nothing-at-stake. These target the blockchain consensus layer.
- 2) *Optimization Fraud*: Performance inflation, gradient poisoning, overfitting to evaluation data, result fabrication. These exploit the optimization verification process.
- 3) *Identity Attacks*: Sybil attacks, reputation farming, identity recycling. These manipulate the identity and reputation systems.
- 4) *Network Attacks*: Eclipse attacks, DoS, transaction censorship, network partitioning. These target the communication layer.
- 5) *Economic Attacks*: Front-running, parameter theft, sandwiching, market manipulation, under-reporting. These exploit the economic incentive structure.

B. Hardening Measures

TABLE III: HARDENING MEASURES AND MITIGATED ATTACKS

Measure	Attacks Mitigated
Commit-reveal protocol	Front-running, parameter theft
Random quorum selection	Collusion, rubber-stamping
Asymmetric reputation	Sybil, reputation farming
Resource limits	DoS, OOM exhaustion
Finality checkpoints	Long-range rewrite
EMA reputation decay	Reputation farming
Min reputation threshold	Sybil identity flooding
External checkpoint anchoring	51% attack
Heaviest-chain fork choice	Selfish mining
Deterministic evaluation seeds	Hidden randomness exploit

C. Formal Security Properties

Theorem 3 (Verification Soundness). *If the fraction of Byzantine evaluators $f < K/(2M)$, then the probability of a fraudulent optimization passing verification is negligible in K .*

Proof sketch. Each evaluator independently generates synthetic data and evaluates the submitted parameters. For a fraudulent result to pass, the median of K evaluations must fall within tolerance ϵ of the claimed performance. With at least $\lceil K/2 \rceil$ honest evaluators, the median is determined by honest evaluations, which will reflect the true performance with high probability. The probability of all honest evaluators coincidentally producing results within ϵ of a false claim decreases exponentially with K . ■

Theorem 4 (Reputation Convergence). *Under the EMA update rule (11) with asymmetric penalties, the expected reputation of a node playing a mixed strategy with dishonesty probability $q > 1/(1 + \gamma)$ converges to below the minimum threshold r_{\min} .*

Proof sketch. The expected per-round reputation change is $\delta_{\text{exp}} = (1 - q) \cdot R_{\text{reward}} - q \cdot \gamma \cdot R_{\text{reward}} = R_{\text{reward}}(1 - q(1 + \gamma))$. For $\gamma = 3$ and $q > 0.25$, $\delta_{\text{exp}} < 0$, and the EMA converges to a negative fixed point, eventually falling below r_{\min} . ■

VII. IMPLEMENTATION

The reference implementation of DOIN consists of five Python packages, designed with a modular plugin architecture using setuptools entry points for extensibility.

A. Package Architecture

The system is organized into the following packages:

doin-core: The blockchain data structures, consensus engine, reputation system, and task queue. This package implements the PoO consensus protocol, VUW weighting, and fork choice rule.

doin-node: The peer-to-peer networking layer, block propagation, and node lifecycle management. Implements the commit-reveal protocol and quorum selection.

doin-optimizer: Optimization engine supporting gradient-based and black-box optimization methods. Provides the interface between ML frameworks and the DOIN protocol.

doin-evaluator: Evaluation engine implementing synthetic data generation, model evaluation, and result aggregation. Handles the per-evaluator seed derivation from equation (9).

doin-plugins: Plugin registry and reference implementations for domain-specific optimization tasks, including neural architecture search, hyperparameter optimization, and reinforcement learning.

B. Plugin System

DOIN's extensibility is achieved through Python `setuptools` entry points. New optimization domains, evaluation metrics, and consensus parameters can be added without modifying core packages. Each plugin registers three entry points: a domain definition (model architecture and loss function), a synthetic data generator, and an evaluation function.

C. Testing

The implementation includes 291 tests across all five packages, covering unit tests for individual components, integration tests for the full optimization-verification-consensus pipeline, and property-based tests for the game-theoretic invariants (e.g., that honest strategies always yield non-negative expected payoffs).

VIII. CONCLUSION AND FUTURE WORK

We have presented DOIN, a decentralized network that converts blockchain consensus computation into productive machine learning optimization. The Proof of Optimization consensus mechanism, with its VUW weighting and dynamic threshold adjustment, ensures that all computational work contributing to block production is verifiable and useful. Our game-theoretic analysis demonstrates that honest optimization is a Nash equilibrium under the DOIN incentive structure, and our security analysis addresses 25 identified attack vectors with 10 hardening measures.

Several directions for future work remain. First, the theoretical analysis could be extended to consider repeated games with discounting, providing stronger guarantees about long-term behavior. Second, the VUW weighting scheme could incorporate mechanism design principles to optimally allocate optimization effort across domains. Third, privacy-preserving techniques such as secure multi-party computation

could be integrated to protect proprietary model architectures while maintaining verifiability. Fourth, large-scale empirical evaluation on real optimization workloads would validate the theoretical predictions regarding convergence and security. Finally, the integration of zero-knowledge proofs for optimization verification could reduce evaluator computational overhead while maintaining the same security guarantees.

DOIN demonstrates that the apparent tension between blockchain security and computational utility can be resolved through careful mechanism design. By aligning consensus incentives with productive optimization work, DOIN offers a path toward decentralized AI infrastructure that is both secure and computationally efficient.

REFERENCES

- [1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. ACL*, 2019, pp. 3645–3650.
- [2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.
- [4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. MLSys*, 2020.
- [5] M. A. Ferrag, L. Shu, H. Djallel, and K.-K. R. Choo, "Blockchain technologies for the internet of things: Research issues and challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2188–2204, 2019.
- [6] H. Bastidas, "DInEMMo: Decentralized inference and evaluation of multi-model optimization," Universidad del Valle, Tech. Rep., 2025.
- [7] T. McConaghy et al., "Ocean Protocol: A decentralized substrate for AI data and services," Ocean Protocol Foundation, Tech. Rep., 2020.
- [8] S. King, "Primecoin: Cryptocurrency with prime number proof-of-work," 2013. [Online]. Available: <http://primecoin.io/bin/primecoin-paper.pdf>
- [9] R. Halford, "Gridcoin: Crypto-currency using Berkeley Open Infrastructure for Network Computing as a proof of work," 2014.
- [10] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of work from worst-case assumptions," in *Proc. CRYPTO*, 2018, pp. 789–819.
- [11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
- [12] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, 2014.
- [13] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, 2009.