A Blockchain-Enabled Swarm Intelligence Framework for Secure and Decentralized Optimization

Dr.K.SATHYA,Assistant Professor,Department of Computer Science,Suguna college of Arts and Science,CBE-14

Abstract—Swarm intelligence (SI) algorithms solve difficult optimization problems through the collective behavior of simple agents. Their distributed nature is attractive for Internet of Things (IoT), edge computing, and multi-organization optimization, yet practical deployments face trust, auditability, data-integrity, and coordination challenges. This paper proposes a blockchain-enabled swarm intelligence framework (BESIF) that combines off-chain swarm computation with an on-chain coordination and audit layer. Particle-swarm-style agents perform local search using signed observations and exchange bounded candidate summaries. A permissioned blockchain records agent registration, task configuration, candidate commitments, validator decisions, and iteration checkpoints through policy-enforcing smart contracts. Large observations and model artifacts remain off-chain; their cryptographic hashes and access metadata are anchored on-chain. The framework is specified through a constrained optimization model, a hybrid algorithm, threat assumptions, and a reproducible evaluation protocol. Analytical discussion identifies the expected trade-offs: blockchain can improve tamper evidence and accountable coordination, but adds communication, consensus, storage, and privacy costs. No experimental measurements are claimed in this design paper; a benchmark plan and reporting templates are provided so that performance claims can be validated on a real deployment. The proposed architecture is intended as a reusable design for trust-aware distributed optimization rather than a claim that blockchain universally improves SI performance.

Index Terms—Swarm intelligence, blockchain, particle swarm optimization, decentralized optimization, smart contracts, IoT, edge computing

I. INTRODUCTION

Distributed optimization is increasingly used where data, computation, and decision authority are spread across devices or organizations. Examples include sensor placement, energy scheduling, routing, federated resource allocation, and anomaly-response coordination. Swarm intelligence (SI) is a family of population-based methods inspired by collective behavior in nature. Algorithms such as particle swarm optimization (PSO), ant colony optimization (ACO), and artificial bee colony (ABC) can explore large search spaces without requiring a single centralized controller [1]–[4].

However, distribution does not automatically create trust. In a multi-owner environment, an agent may submit a false fitness value, replay an old update, impersonate another participant, or alter a shared optimization history. A central coordinator can become a single point of failure and may be unacceptable when participants do not fully trust one another. Blockchain offers a replicated, append-only record maintained by a consensus protocol, while smart contracts can enforce deterministic participation rules [5]–[7].

This paper presents BESIF, a hybrid architecture in which the swarm performs the expensive numerical search off-chain and a permissioned blockchain manages identity, task rules, commitments, and audit events. The design deliberately avoids placing raw sensor data, complete particle vectors, or large models on-chain. Instead, cryptographic hashes and compact metadata provide integrity anchors. The main contributions are: (1) a layered architecture separating optimization from trust management; (2) a mathematical formulation for constrained, multi-agent optimization; (3) a hybrid PSO-based algorithm with contract-mediated admission and checkpointing; (4) a threat model and design trade-off analysis; and (5) a reproducible evaluation plan with metrics and reporting tables.

The rest of the paper is organized as follows. Section II reviews the relevant foundations. Section III describes the system and threat model. Section IV gives the mathematical formulation. Section V presents the proposed algorithm and contract logic. Section VI discusses security and privacy. Section VII specifies the evaluation methodology and reporting templates. Section VIII discusses limitations, and Section IX concludes the paper.

II. BACKGROUND AND RELATED WORK

A. Swarm intelligence

SI methods use a population of simple entities whose local interactions produce a collective search process. In PSO, each particle has a position and velocity; personal and neighborhood best positions guide its movement [1]. ACO uses pheromone trails and probabilistic path construction [2], while ABC models employed, onlooker, and scout bees [3]. These methods are attractive for distributed settings because candidate evaluation can often be parallelized.

B. Blockchain and smart contracts

A blockchain is a replicated data structure in which blocks are linked using cryptographic hashes and accepted according to a consensus protocol. Permissionless networks permit broad participation but may incur variable fees and latency. Permissioned networks restrict validator membership and are often more appropriate for known organizations requiring governance and predictable access control. Smart contracts execute agreed rules, but their code and inputs must be designed carefully: deterministic execution does not prove that an off-chain fitness value is truthful [5]–[7].

C. Related integration patterns

Prior research has explored blockchain for IoT security, data provenance, federated learning coordination, and decentralized marketplaces. These systems motivate the use of on-chain identities, event logs, and hash anchoring. SI research, meanwhile, emphasizes convergence behavior, exploration–exploitation balance, and sensitivity to population size and parameter selection. BESIF combines these strands but treats the ledger as a trust and accountability substrate—not as a replacement for the numerical optimizer.

D. Research gap

A common architectural mistake is to store every optimization state on-chain. This increases transaction volume, latency, and cost, while exposing sensitive search information. Another mistake is to assume that consensus validates the semantic correctness of a candidate. Consensus only establishes agreement about submitted

transactions under the protocol assumptions. BESIF therefore stores compact commitments, uses off-chain evaluation, and explicitly separates integrity verification from truth verification.

III. SYSTEM MODEL AND ARCHITECTURE

A. Participants

The system includes (i) task owner, which publishes the objective, constraints, budget, and termination policy; (ii) swarm agents, which maintain candidate solutions and evaluate them using authorized data; (iii) validator nodes, which order transactions and execute contract rules; and (iv) an off-chain artifact store, such as an access-controlled object store or content-addressed system, for large data and model files. A governance authority onboards organizations and revokes compromised identities.

B. Layered architecture

Fig. 1 shows the proposed architecture. The application layer defines a task and receives the selected solution. The swarm layer performs candidate generation, local fitness evaluation, and neighborhood exchange. The blockchain layer maintains identities, task versions, commitments, and audit events. The storage layer retains large artifacts off-chain. A candidate is accepted only if its task identifier, agent signature, iteration number, data-version hash, and policy checks are valid.

C. On-chain and off-chain split

On-chain records include agent public-key identifiers, task configuration hashes, candidate hashes, iteration checkpoints, validator decisions, and revocation events. Off-chain records include raw sensor readings, full particle vectors, fitness traces, and model artifacts. Each off-chain object is addressed by a cryptographic digest. The task owner can later verify that a retrieved object matches the digest recorded on-chain. Access control and encryption remain necessary because a hash does not provide confidentiality.

D. Communication model

Agents communicate through authenticated channels and submit transactions to a gateway. A local optimizer may operate asynchronously, but the contract accepts only monotonically increasing iteration numbers or explicitly permitted late updates. To reduce ledger load, agents batch multiple candidate commitments into a Merkle tree and submit the root. The detailed leaves remain off-chain and can be disclosed selectively for audit.

IV. MATHEMATICAL FORMULATION

A. Optimization objective

Let there be N agents and a decision vector x in a bounded feasible region $\Omega \subseteq R^d$. The task is to minimize f(x), subject to inequality constraints $g_j(x) \le 0$, j=1,…,m, and equality constraints $h_k(x)=0$, k=1,…,r. A common penalty formulation is $F(x)=f(x)+\lambda_g \Sigma_j \max(0,g_j(x))^2+\lambda_h \Sigma_k h_k(x)^2$, where λ_g and λ_h are nonnegative penalty weights. For multi-objective tasks, f may be replaced by a vector objective and a Pareto archive.

B. Particle update

For particle i at iteration t, position x_i^t and velocity v_i^t are updated using $v_i^{(t+1)}=\omega v_i^t+c_1r_1(p_i^t-x_i^t)+c_2r_2(l_i^t-x_i^t)$, and $x_i^{(t+1)}=\Pi_\Omega(x_i^t+v_i^{(t+1)})$. Here ω is inertia, c_1 and c_2 are acceleration coefficients, r_1 and r_2 are independent uniform random vectors in $[0,1]^d$, p_i is the personal best, l_i is the best candidate in the permitted neighborhood, and Π_Ω projects onto the feasible domain. A velocity clamp may be applied to prevent unstable steps.

C. Trust-weighted neighborhood

Let $q_j \in [0,1]$ denote a policy-derived trust score for agent j, based on verifiable factors such as valid signatures, timely submissions, and audit outcomes—not on subjective reputation alone. A trust-aware neighborhood can select l_i from the eligible set $N_i=\{j: q_j \ge \tau$ and status_j=active$\}$. Trust must not be treated as proof of honest fitness; independent re-evaluation, robust aggregation, or task-specific verification is needed where adversarial values are consequential.

D. Integrity commitment

For candidate record R_i^t, define $C_i^t=H(\text{taskID} \| \text{agentID} \| t \| H(x_i^t) \| H(\text{dataVersion}) \| H(F_i^t) \| \text{nonce})$, where H is a collision-resistant hash and ‖ denotes unambiguous serialization. The agent signs C_i^t using its private key. The contract verifies signature, task membership, iteration policy, and commitment uniqueness. A later auditor can recompute the commitment from disclosed data and compare it with the ledger entry.

E. Complexity

For N particles and d dimensions, a conventional PSO iteration requires O(Nd) arithmetic, excluding objective evaluation. If each agent submits one transaction per iteration, ledger transaction demand is O(NT) for T iterations. Batching B candidates into one Merkle-root transaction reduces the number of on-chain submissions to approximately O(ceil(N/B)T), while retaining off-chain proof material. Hashing and signature verification add costs proportional to the size and number of committed records.

V. PROPOSED BESIF ALGORITHM

A. Design principles

BESIF follows five principles: numerical work stays off-chain; all participants are authenticated; task rules are versioned; candidate history is tamper-evident; and sensitive information is minimized. The baseline optimizer is PSO because its update rule is compact and widely studied. The architecture can also host ACO, ABC, or hybrid optimizers by replacing the swarm-update module while retaining the ledger interface.

B. Algorithm steps

At initialization, the task owner registers a task hash, bounds, constraints, objective version, termination rule, and eligible agents. Each agent initializes particles and evaluates candidates locally. At each iteration, the agent updates its state, computes a candidate commitment, signs it, and submits either the commitment or a Merkle batch root. The contract verifies authorization and sequence constraints, then emits an acceptance or rejection event. Validators order accepted transactions. Agents read the agreed checkpoint and continue. At termination, the task owner selects or independently verifies the best candidate and records the final solution hash.

C. Pseudocode

Algorithm 1 summarizes the process. The pseudocode assumes a permissioned ledger with deterministic contract execution and authenticated agents. The objective itself may be private or

expensive, so the contract checks metadata and authorization rather than recomputing the objective.

D. Smart-contract responsibilities

The contract maintains task state, authorized agent keys, task version, iteration checkpoint, used nonces, and commitment events. It rejects unknown agents, revoked keys, malformed task identifiers, stale or duplicate iteration values, and submissions exceeding configured limits. It does not store private observations or claim that a committed fitness value is correct. Contract upgrades require governance controls, review, and an auditable migration procedure.

VI. SECURITY, PRIVACY, AND GOVERNANCE

A. Threat assumptions

The design considers malicious or faulty agents, replay attempts, identity spoofing, unauthorized task modification, compromised gateways, and a minority of Byzantine validators within the fault tolerance of the selected consensus protocol. It assumes secure cryptographic primitives, protected signing keys, and a governance process for validator admission. If the consensus threshold is exceeded or validator collusion occurs, ledger guarantees may fail.

B. Integrity and accountability

Digital signatures bind a submission to an agent identity. Hash commitments make later alteration detectable. Append-only events provide an audit trail of accepted and rejected updates. Sequence numbers and nonces reduce replay risk. These mechanisms establish provenance and consistency, not semantic truth: an agent can still sign a fabricated fitness score. High-assurance applications should add trusted execution, redundant evaluation, challenge periods, or verifiable computation where feasible.

C. Privacy

Raw data and full candidate vectors should remain off-chain unless they are demonstrably non-sensitive. Data should be encrypted in transit and at rest, with role-based access and key rotation. Hashes can still leak information when the underlying values have low entropy or can be guessed; salted commitments and carefully designed metadata may be required. Differential privacy can be considered when aggregate information is released, although it changes optimization accuracy.

D. Governance and availability

A permissioned deployment requires clear rules for onboarding, revocation, dispute resolution, software upgrades, and validator failure. Off-chain storage must be replicated and backed up; a ledger pointer does not guarantee that an artifact remains available. Gateways should rate-limit submissions and validate payload size before transaction creation. Disaster recovery should preserve both ledger state and off-chain content-addressed artifacts.

VII. EXPERIMENTAL DESIGN AND EVALUATION PLAN

A. Research questions

The evaluation should answer: RQ1, does BESIF preserve solution quality relative to the same optimizer without blockchain? RQ2, what latency and throughput overhead does ledger coordination introduce? RQ3, how does batching affect transaction count and end-to-end time? RQ4, how does the system behave under agent churn, delayed messages, and malicious submissions?

B. Baselines

Compare (i) centralized PSO with a trusted coordinator; (ii) distributed PSO without blockchain; (iii) BESIF with one commitment per agent per iteration; and (iv) BESIF with Merkle batching. Keep objective functions, population size, random seeds, hardware, and stopping criteria consistent. Include standard benchmark functions such as Sphere, Rastrigin, and Rosenbrock, followed by one domain-specific workload. Run at least 30 independent random seeds where computationally feasible.

C. Metrics

Report best objective value, median and interquartile range across runs, convergence evaluations, feasibility rate, wall-clock runtime, transaction throughput, confirmation latency, bytes written on-chain, off-chain storage volume, and energy or monetary cost where measurable. Security tests should report replay rejection, unauthorized-submission rejection, detection of altered artifacts, and recovery under node or agent failure.

D. Reproducibility

Publish parameter settings, code version, contract version, network topology, consensus configuration, hardware, dataset hashes, random seeds, and raw measurements. Separate optimizer time from consensus wait time. Report confidence intervals and nonparametric tests when distributions are skewed. Do not infer a performance advantage from a single run.

E. Reporting templates

Tables II and III are templates to be populated after implementation. They intentionally contain no invented measurements. The paper should not be submitted as an empirical evaluation until the tables are replaced with reproducible results.

VIII. DISCUSSION AND LIMITATIONS

The principal benefit of BESIF is accountable coordination across parties that need a shared history but do not want a single organization to control the record. It is most suitable when auditability, provenance, and multi-party governance justify the ledger overhead. For a single trusted operator or a latency-critical embedded controller, a conventional distributed database or signed event log may be simpler and faster.

The framework has limitations. First, consensus latency may slow frequent swarm updates; batching and periodic checkpoints mitigate but do not eliminate this cost. Second, a blockchain cannot validate arbitrary objective functions without additional proof mechanisms. Third, public metadata may reveal participation patterns. Fourth, permissioned consensus introduces governance and validator-collusion risks. Fifth, the PSO formulation is a baseline, not a universally optimal SI method. Algorithm selection must follow the problem structure.

Future work should implement the architecture on a permissioned platform, measure end-to-end performance under realistic network conditions, evaluate Byzantine-resistant aggregation, and investigate zero-knowledge proofs or trusted execution for selected classes of verifiable fitness computation. Adaptive checkpoint intervals and privacy-preserving multi-objective optimization are also promising directions.

IX. CONCLUSION

This paper proposed BESIF, a blockchain-enabled swarm intelligence framework that separates off-chain optimization from on-chain trust management. The design uses authenticated agents, smart-contract policy checks, cryptographic commitments, and optional Merkle batching to create an auditable record while keeping large and sensitive artifacts outside the ledger. A constrained optimization model, PSO update equations, hybrid algorithm, threat model, and reproducible evaluation plan were presented. Blockchain does not guarantee truthful objective values and does not automatically improve convergence; its value lies in shared integrity and governance when multiple parties require accountable coordination. Future implementation and controlled experiments are necessary to quantify the quality, latency, throughput, and cost trade-offs.

FIGURES AND ALGORITHM

Fig. 1. Proposed layered blockchain-enabled swarm architecture.

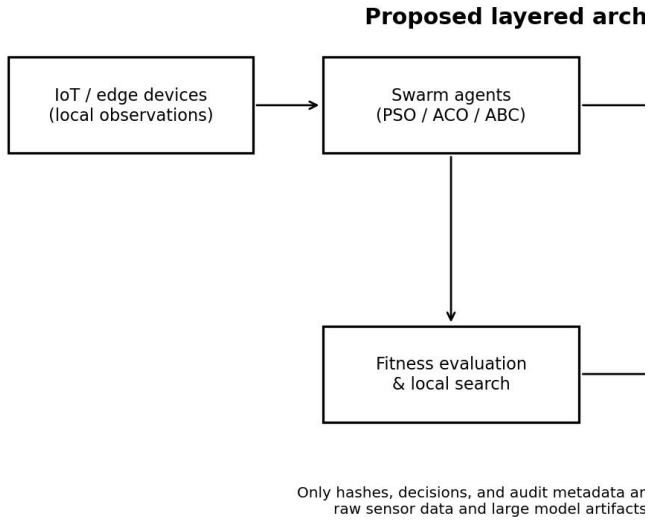
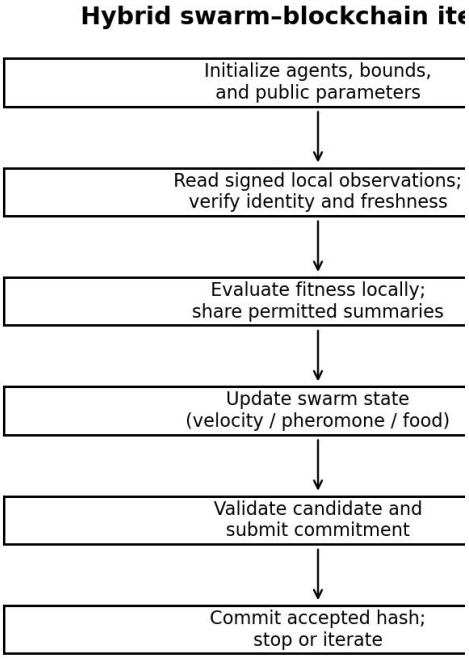


Fig. 2. Control flow for one swarm–ledger iteration.



Algorithm 1: Hybrid BESIF–PSO procedure

```
Input: task T, bounds Ω, N agents, swarm parameters
ω,c1,c2, iteration limit K
Register H(T), objective version, agent public keys,
and policy on the permissioned ledger.
Each agent initializes particles xᵢ, vᵢ; evaluates
F(xᵢ); stores personal best pᵢ.
For t = 1,…,K do
  Read the latest finalized checkpoint and eligible
neighborhood best lᵢ.
  For each local particle i: update vᵢ and xᵢ using
the PSO equations; project xᵢ onto Ω.
  Evaluate F(xᵢ) locally; update pᵢ and local/global
candidate summaries.
  Serialize task ID, agent ID, t, candidate hash,
data-version hash, fitness hash, and nonce.
  Sign the commitment; submit it directly or as a
Merkle-batch leaf/root.
  Contract checks identity, signature, task version,
nonce, iteration, and rate limits.
  Validators finalize accepted events; agents reject
non-finalized or invalid checkpoints.
  If termination rule is met, independently verify the
selected candidate and commit final hash.
End For; Output: selected solution, audit trail, and
reproducibility manifest.
```

TABLE I. ON-CHAIN AND OFF-CHAIN DATA PLACEMENT

Item	On-chain representation	Off-chain content
Task definition	Task ID, version hash, policy	Full objective details if confidential
Candidate	Commitment hash, agent ID, iteration	Vector, fitness trace, metadata
Dataset	Dataset/version digest	Raw observations and labels
Audit	Events, signatures, validator decision	Proof material and detailed logs
Model/artifact	Content hash and access pointer	Model weights and binary artifacts

TABLE II. EVALUATION METRICS

Metric	Definition
Solution quality	Best and median objective across independent runs
Feasibility	Percentage of runs satisfying all constraints
Convergence effort	Objective evaluations to reach fixed target
End-to-end latency	Task start to finalized solution (s)
Ledger throughput	Accepted transactions per second
Confirmation latency	Submission-to-finality time
Storage overhead	On-chain bytes and off-chain storage volume
Robustness	Acceptance/rejection under fault scenarios

TABLE III. RESULTS TEMPLATE — POPULATE AFTER EXPERIMENTS

Method	Best objective	Median objective	Runtime (s)	Tx/run
Centralized PSO	—	—	—	0
Distributed PSO	—	—	—	0
BESIF individual commits	—	—	—	—
BESIF Merkle batching	—	—	—	—

Note: em dashes indicate measurements not yet collected; they are not zero values.

REFERENCES

[1] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proc. IEEE Int. Conf. Neural Networks, Perth, Australia, 1995, pp. 1942–1948.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: Optimization by a colony of cooperating agents,” IEEE Trans. Syst., Man, Cybern. B, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[3] D. Karaboga, “An idea based on honey bee swarm for numerical optimization,” Technical Report TR06, Erciyes University, 2005.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems. New York, NY, USA: Oxford Univ. Press, 1999.

[5] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[6] K. Christidis and M. Devetsikiotis, “Blockchains and smart contracts for the Internet of Things,” IEEE Access, vol. 4, pp. 2292–2303, 2016.

[7] M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” in Proc. 3rd USENIX Symp. Operating Systems Design and Implementation, 1999, pp. 173–186.

[8] M. Conti, A. Di Pietro, L. V. Mancini, and A. Mei, “A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks,” in Proc. ACM MobiHoc, 2007, pp. 80–89.

[9] A. Y. S. Lamport, “The part-time parliament,” ACM Trans. Comput. Syst., vol. 16, no. 2, pp. 133–169, 1998.

[10] N. Szabo, “Formalizing and securing relationships on public networks,” First Monday, vol. 2, no. 9, 1997.