Bi-Scale Evaluation Particle Swarm Optimization for Neuron-Level Enhanced Backdoor Attacks

Huan-Yu Chen
College of Cryptology and Cyber Science
Nankai University
Tianjin, 300350, China

Guanxiong Ha (Corresponding Author)
College of Cryptology and Cyber Science
Nankai University
Tianjin, 300350, China

Chunfu Jia
College of Cryptology and Cyber Science
Nankai University
Tianjin, 300350, China

Jun Zhang
College of Artificial Intelligence
Nankai University
Tianjin, 300350, China

Zhi-Hui Zhan (Corresponding Author)
College of Artificial Intelligence
Nankai University
Tianjin, 300350, China
zhanapollo@163.com

***Abstract*—Existing backdoor attacks usually suffer from poor generalization across different neural network architectures and datasets. Enhancing the generalization capability of existing attack models is termed as enhanced backdoor attacks problem (EBAP). The essence of EBAP is to precisely select a subset of key neurons from the attack model and perturb their parameters, so as to enhance the performance of the attack model. To efficiently solve the EBAP, this paper models the EBAP as a 0-1 integer programming problem for fine-tuning neural-level parameters, and proposes a bi-scale evaluation particle swarm optimization (BSEPSO) algorithm based on a bi-scale evaluation (BSE) mechanism. As evaluating the fitness value of an EBAP solution is expensive, the BSE mechanism adopts a fast evaluation and a full evaluation to work cooperatively to reduce the computational cost. This way, the BSEPSO algorithm achieves efficient optimization in high-dimensional neural spaces while effectively balancing evaluation accuracy and computational overhead. Experimental results demonstrate that across extensive testing on 2 datasets and 15 mainstream attack models, BSEPSO significantly enhances attack success rate (ASR) while strictly controlling clean accuracy (CA) fluctuations within safety thresholds, validating its effectiveness as a universal attack enhancement framework.**

Keywords—Backdoor Attack, Enhanced Backdoor Attacks Problem, 0-1 Integer Programming Problem, Bi-Scale Evaluation, Particle Swarm Optimization, Evolutionary Computation.

I. INTRODUCTION

The widespread adoption of neural networks has raised increasing concerns about their security, with backdoor attacks posing a major threat to artificial intelligence (AI) systems due to their high stealthiness [1][2]. In such attacks, adversaries implant triggers during training so that the model maintains high clean accuracy (CA) on benign inputs while producing predefined erroneous outputs when triggered.

Although existing attack methods like BadNets [1] and Trojan [1] demonstrate strong performance on specific benchmark models and datasets, they still face the challenge of performance degradation in complex and dynamic deployment environments. First, attack effectiveness is highly sensitive to model architecture. Many existing attacks fail to maintain consistent attack success rates (ASR) across all victim models. Second, attacks exhibit poor transferability between datasets. A backdoor effective on dataset A may fail on more complex datasets B, exposing the instability of backdoor logic embedded within models [3].

To overcome these limitations, enhancing backdoor attacks within existing attack models has gradually become a research hotspot. This problem is referred to as the enhanced backdoor attacks problem (EBAP). The core idea of solving the EBAP is to fine-tune the internal parameters of an infected model, significantly improving ASR while maintaining CA [4]. Since the essence of backdoor attacks lies in establishing mappings between triggers and specific neuron activations, this enhancement problem can be modeled as a localization and perturbation challenge targeting critical neurons. Specifically, it involves precisely selecting the optimal subset from a parameter space containing tens of thousands of neurons and determining their perturbation steps. Therefore, the EBAP can be regarded as a large-scale 0-1 programming problem [5].

Evolutionary computation (EC) [6][7], with its unique advantage of efficiently performing global optimization in non-convex, high-dimensional search spaces, has been widely applied to solving various of problems [8]-[10] and 0-1 integer programming problem [11][12]. This family of algorithms encompasses various nature-inspired paradigms, such as particle swarm optimization (PSO) [13][14], genetic algorithm (GA) [15][16], and ant colony optimization (ACO) [17][18]. With a limited computational budget, PSO has faster initial convergence and higher search efficiency compared to GA and ACO [19]. For using EC algorithms to solve 0-1 programming problem, Li et al. [20] proposed a coevolutionary multi-objective discrete PSO (CMODPSO) for the gateway placement optimization problem, which is a typical 0-1 integer programming scenario. Samma et al. [21] proposed a reinforcement learning-based memetic PSO (RLMPSO), integrating PSO's global search capability and memetic algorithm's local refinement to adapt to 0-1 decision spaces. Yang et al. [22] employed the bare bones PSO-adaptive chaotic jump-bi-directional feature fixation (BBPSO-ACJ-BDFF) to address the feature selection problem.

This work was supported in part by the National Key Research and Development Program of China under Grant 2024YFF0509600, in part by the National Natural Science Foundation of China (NSFC) under Grant 62576175 and Grant U23B2039, in part by the Tianjin Science and Technology Major Special Project under Grant 25ZXSFSN00070, in part by the Natural Science Foundation of Tianjin Key Program under Grant 25JCZDJC01040, in part by the Tianjin Top Scientist Studio Project under Grant 24JRRCRC00030, in part by the Tianjin Belt and Road Joint Laboratory under Grant 24PTLYHZ00250, in part by the Fundamental Research Funds for the Central Universities under Grant 63261167, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIT) (No. RS-2025-00555463 and No. RS-2025-25456394).

Given its excellent performance, this paper adopts BBPSO-ACJ-BDFF as the base algorithm for solving EBAP.

However, directly applying BBPSO-ACJ-BDFF to solve the EBAP faces the difficulty in the trade-off between evaluation efficiency and accuracy [23]. Entire dataset evaluations are computationally expensive, while sampling-based evaluations degrade optimization reliability. To address the aforementioned challenges, this paper introduces a bi-scale evaluation (BSE) mechanism to balance the cost of sampling-based evaluation (i.e., fast evaluation) and entire dataset evaluation (i.e., full evaluation). Based on the BSE mechanism, a bi-scale evaluation PSO (BSEPSO) algorithm is proposed. The personal best position ($pbest$) and global best position ($gbest$) updates in the BSEPSO algorithm rely on the BSE mechanism. Moreover, BSEPSO further incorporates an elite pool strategy to preserve high-quality genes during the evolutionary process and enhance the global search capability by using selective domain fixed strategy.

Experimental results demonstrate that BSEPSO exhibits exceptional universality across 15 main-stream attack models and 2 datasets. It significantly enhances the ASR for most attack models while maintaining the CA within the no-degradation threshold.

In the remainder of this paper, Section II models the EBAP. Section III details the proposed BSEPSO. Section IV presents the experiments. Finally, Section V concludes the paper.

II. Problem Formulation

Backdoor attacks implant covert malicious behaviors into neural networks by establishing strong associations between specific triggers and a target class. After training, the infected model behaves normally on clean inputs, while producing attacker-specified outputs when the trigger is present [2]. Rather than modifying the backdoor training process itself, this work focuses on the enhancement stage, in which attack effectiveness is primarily determined by the activation and interaction of a subset of critical neurons.

However, modern neural networks typically contain millions of neurons, rendering direct heuristic search over the entire neuron space computationally prohibitive. To address this challenge, a gradient-based sensitivity analysis is employed to construct a reduced and tractable search space.

Specifically, we first compute the gradient of the poisoning loss function L_p with respect to the model parameters. The poisoning loss L_p is defined as the cross-entropy between the model's prediction and the target label y_t for poisoned samples, using the poisoned test set $\mathcal{D}^p$ [1]. L_p is calculated as

$$L_p = -\frac{1}{|\mathcal{D}^p|}\sum_{(x,y_t)\in\mathcal{D}^p}\log\big(P(f_\theta(x')=y_t)\big), \quad (1)$$

in which the f_θ is the original model. The magnitude of this gradient $|\nabla_{\theta_j}L_P|$ quantifies the neuron's sensitivity to backdoor features, which is calculated as

$$\nabla_{\theta_j}L_p = \frac{\partial L_p}{\partial \theta_j}. \quad (2)$$

A larger gradient magnitude indicates that changes in these neurons' model parameters exert a more significant influence. Then, neurons are sorted by gradient magnitude from largest to smallest, selecting the top K neurons to form the candidate set $Scand$. Once the search space is reduced to set $Scand$., the EBAP becomes a K dimensional 0–1 integer programming problem, where each neuron must be either selected or excluded for perturbation.

EBAP's objective is to identify an optimal subset of neurons that maximizes the ASR, while constraining CA degradation [3]. Based on this dual-objective optimization principle, the specific evaluation metrics in this paper are as follows:

The first objective focuses on evaluating gains in ASR. We apply the binary perturbation scheme corresponding to particle $\boldsymbol{X_i}$ to the model parameters θ. Finally, we obtain the enhanced model $f_{\theta+\delta\cdot X_i}$ [2]. The equation for ASR is

$$\mathrm{ASR}\big(f_{\theta+\delta\cdot X_i}\big) = \frac{1}{|\mathcal{D}^p|}\sum_{x\in\mathcal{D}^p}\mathbb{I}\left(f_{\theta+\delta\cdot X_i}(x') = y_t\right). \quad (3)$$

Among these, the $\mathbb{I}(\cdot)$ is used to count the total number of samples where the attack succeeded, and the x' is input.

The second objective focuses on evaluating the retention level of CA. We define the performance penalty term by evaluating the computational model's performance on the cleaned test set $\mathcal{D}^c$ [2]. The equation for CA is

$$\mathrm{CA}\big(f_{\theta+\delta\cdot X_i}\big) = \frac{1}{|\mathcal{D}^c|}\sum_{x\in\mathcal{D}^c}\mathbb{I}\left(f_{\theta+\delta\cdot X_i}(x) = y\right). \quad (4)$$

Combining the above two points, the fitness function $F(X_i)$ ultimately defined in this paper is as

$$max\ F(X_i) = \widetilde{\mathrm{ASR}}(f_{\theta+\delta\cdot X_i}) - \lambda\cdot max(0, \eta - \widetilde{\mathrm{CA}}(f_{\theta+\delta\cdot X_i})), \quad (5)$$

where the η is the predefined fitness threshold, and the λ is the penalty factor used to impose a substantial fitness reduction on solutions falling below the threshold. It is worth noting that both ASR and CA evaluations within the fitness function employ a fast evaluation. The specific sampling mechanism and the collaborative relationship between the fast and full evaluation scales will be detailed in Section IV. In this paper, $\widetilde{\mathrm{ASR}}(f_{\theta+\delta\cdot X_i})$ is the fast evaluation ASR and $\widetilde{\mathrm{CA}}(f_{\theta+\delta\cdot X_i})$ is the fast evaluation CA. Regarding enhancements to backdoor attacks, the following conditions must be met.

Constraint 1: To ensure model utility and stealthiness, we impose a CA threshold constraint as

$$\mathrm{CA}(f_{\theta+\delta\cdot X_i}) \geq max(0, \mathrm{CA}_{orig} - \Delta\mathrm{CA}). \quad (6)$$

The CA_{orig} denotes the CA of the original backdoor attack model, while the ΔCA represents the maximum permissible accuracy loss (set to 0.02 in this paper).

Constraint 2: For ensuring heuristic search yields positive gains, ASR hard constraint is set as

$$\mathrm{ASR}(f_{\theta+\delta\cdot X_i}) \geq \mathrm{ASR}_{orig}, \quad (7)$$

in which the ASR_{orig} denotes the ASR of the original backdoor attack model.

Constraint 3: To control the scope of model modifications, the number of perturbed neurons must be controlled. It is defined as

$$N_{min} \leq \|X_i\|_0 \leq N_{max}. \quad (8)$$

Here, the $\|X_i\|_0$ is the total number of activated neurons. The N_{min} represents the minimum selectable number and the N_{max} denotes the maximum selectable number.

Constraint 4: To achieve reasonable perturbation of key neurons, this paper designs a linear dynamic perturbation as

$$\delta = \delta_0 \cdot \left(1 - 0.11 \cdot \frac{g}{G_{max}}\right). \quad (9)$$

Here, the g denotes the current evaluation count, G_{max} represents the maximum evaluation count, and the initial perturbation scale δ_0 is set to 0.02. It should be noted that in this paper, perturbations are only used in addition. And the values of the perturbation vector δ constrain is set as

$$\delta_{min} \leq \delta \leq \delta_{max}, \quad (10)$$

where the δ_{max} and the δ_{min} define the upper and lower bounds for perturbation search.

III. The Proposed BSEPSO Algorithm

A. Particle Encoding

Given the reduced candidate neuron set *Scand* with *K* neurons obtained from gradient-based sensitivity analysis, BSEPSO adopts a binary encoding scheme to represent each particle. This paper uniformly numbers the neurons in each layer of the neural network according to their topological order to construct the neuron set $\boldsymbol{S} = \{s_1, s_2, \ldots, s_K\}$. After generating *N* particles, each particle $\boldsymbol{X_i}$ is encoded as a vector of the same dimension as $\boldsymbol{S}$ [5][12]. $\boldsymbol{X_i}$ is defined as

$$\boldsymbol{X}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,K}], \quad (11)$$

where each bit each bit $x_{i,j}$ is a binary value. $x_{i,j}$ of the ith particle corresponds one-to-one with neuron s_j. $x_{i,j} = 1$ indicates applying a perturbation, while $x_{i,j} = 0$ preserves its original parameters unchanged. Neuron selection becomes a 0–1 program under this encoding.

To initialize the BSEPSO, we randomly generate an initial position $x_{i,j}$ with values 0/1 for each of the *K* dimensions within each particle. In this paper, position updates do not depend on the velocity $v_{i,j}$. The velocity v has been discarded [22]. Then we use the domain partitioning dividing the particle composed of *K* dimensions into *R* domains. And each domain consists of *T* neurons [22].If the number of neurons in the last neighborhood is less than *T*, randomly generated placeholders are added to complete the set. We still divide the population $\boldsymbol{X}$ into two subpopulations uniformly: the searching more neurons subpopulation SSu and the searching less neurons subpopulation SSl.

B. BSE Mechanism

During the optimization process, fitness evaluation is one of the most computationally intensive operations. To resolve this problem, this paper proposes a BSE mechanism. By organically combining fast evaluations with full evaluations, it achieves a balance between computational efficiency and evaluation accuracy.

Fast Evaluation: Fast evaluation is performed on small batches of test samples to enable frequent fitness calculations and guide optimization directions. Specifically, $\mathcal{D}_{sub}^{p}$ is a small batch selected from the full poisoned test set $\mathcal{D}^{p}$, and $\mathcal{D}_{sub}^{c}$ is a corresponding small batch derived from the full clean test set $\mathcal{D}^{c}$. Under fast evaluation, ASR and CA can be calculated by

$$\widetilde{\mathrm{ASR}}(f_{\theta+\delta\cdot X_i}) = \frac{1}{|\mathcal{D}_{sub}^{p}|}\sum_{x\in\mathcal{D}_{sub}^{p}} \mathbb{I}\,(f_{\theta+\delta\cdot X_i}(x') = y_t), \quad (12)$$

$$\widetilde{\mathrm{CA}}(f_{\theta+\delta\cdot X_i}) = \frac{1}{|\mathcal{D}_{sub}^{c}|}\sum_{x\in\mathcal{D}_{sub}^{c}} \mathbb{I}\,(f_{\theta+\delta\cdot X_i}(x) = y). \quad (13)$$

The computational complexity of the quick evaluation is $O(\mathcal{D}_{sub}^{p/c}\times C)$. *C* represents the computational cost of a single sample. It is significantly lower than the complexity of the full evaluation $O(\mathcal{D}^{p/c}\times C)$.

Full Evaluation: Full evaluation is conducted across the entire test set for constraint validation and final decision-making. The ASR under full evaluation is given by Equation (3), and the CA is given by Equation (4).

The BSE mechanism balances computational efficiency and assessment accuracy by coordinating fast evaluation with full evaluation. This mechanism is applied in three distinct phases: the initialization process, the iteration process, and the termination process.

(1) Initialization process: First, all particles are fast-evaluated, and the particle with the highest fast-evaluated fitness among the personal best solutions is selected as the initial $gbest$. This candidate is then subjected to full evaluation on the entire test set to compute $\mathrm{CA}(f_{\theta+\delta\cdot X_i})$ and $\mathrm{ASR}(f_{\theta+\delta\cdot X_i})$. The perturbation solution is accepted as the first perturbation solution *best_solution* only if it satisfies both requirements Constraint 1 and Constraint 2. If these requirements are not met, it retains $gbest$ for subsequent search, while no feasible $best\_solution$ is recorded at this stage.

(2) Iteration process: When a particle achieves a higher fitness than the current $gbest$, it is first subjected to a fast evaluation to pre-screen infeasible candidates. Only particles satisfying $\widetilde{\mathrm{ASR}}(f_{\theta+\delta\times X_i}) \geq \mathrm{ASR}_{orig}$ and $\widetilde{\mathrm{CA}}(f_{\theta+\delta\times X_i}) \geq max(0, \mathrm{CA}_{orig}\text{-}\Delta\mathrm{CA})$ proceed to the next stage. The surviving candidate is then fully evaluated on the entire dataset to compute exact $\mathrm{ASR}(f_{\theta+\delta\cdot X_i})$ and $\mathrm{CA}(f_{\theta+\delta\cdot X_i})$. A particle is considered feasible only if it meets Constraint 1 and Constraint 2. Among feasible candidates, the *best_solution* is updated based on relative ASR improvement: if no feasible solution has been recorded, the first validated candidate is accepted; otherwise, the solution yielding a higher ASR replaces the previous *best_solution*.

(3) Termination process: The algorithm performs a final pareto non-domination verification [24]. Final solutions must not only satisfy all constraints but also adhere to the non-domination principle. Specifically, it must strictly avoid a double-loss scenario where ASR fails to improve while CA suffers deterioration. If the final solutions fail this reliability check, the algorithm triggers a circuit breaker mechanism, restoring the original model state without applying any perturbations.

C. Particle Update

BSEPSO fuses the basic update logic of BBPSO with the chaotic jump strategy of ACJ [22], ensuring both local exploration of high-quality solution regions and effective escape from local optima. The update rules are shown as

$$temp_{i,d} = \begin{cases} N\left(\frac{pbest_{i,d}+gbest_d}{2}, \left|pbest_{i,d} - gbest_d\right|\right) & ,if\ r<P_{cj,i} \\ pbest_{i,d}\cdot(1+(2z_k-1)) & ,otherwise \end{cases}, \quad (14)$$

$$x_{i,d} = \begin{cases} 0\,, & temp_{i,d} < 0.5 \\ 1\,, & temp_{i,d} \geq 0.5 \end{cases}, \quad (15)$$

where $\mathcal{N}(\mu, \pi^2)$ denotes a gaussian distribution with mean μ and standard deviation π. The r is a sampled random value in the interval [0,1]. The $pbest_{i,d}$ represents the dth dimension of ith particle's individual best position and the $gbest_d$ indicates the dth dimension of the population's global best position. $P_{cj,i}$ and z_k are calculated by

$$z_k = \begin{cases} 0.13, & k = 0 \\ 4z_{k-1}(1 - z_{k-1}), & k \geq 1, \end{cases} \quad (16)$$

$$P_{cj,i} = \frac{1}{1+e^{2-s_i}}, \quad (17)$$

in which the s_i is the number of stagnant generations. When the fitness of particles fails to improve across multiple generations, $P_{cj,i}$ automatically increases enhancing the probability of triggering chaotic jumps.

D. pbest And gbest Update

We use BSE mechanism to govern both $pbest$ and $gbest$. $pbest$ updates if fitness improves and the candidate passes the BSE iteration conditions. $gbest$ updates if fitness beats the current $gbest$ and the same BSE iteration conditions are satisfied.

We also employ the BSE mechanism to retain or discard particles. Specifically, particle retention is categorized into no violation, minor constraint violation, and major constraint violation. Minor constraint violations occur when the $0.02 < \mathrm{CA}(f_{\theta+\delta\cdot X_i}) - \mathrm{CA}_{orig} \leq 0.1$. In this case, the new particle is retained to maintain exploration potential in that region, but $gbest$ and $pbest$ are not updated. Severe constraint violations occur when the $\mathrm{CA}(f_{\theta+\delta\cdot X_i}) > 0.1$ or $\mathrm{ASR}(f_{\theta+\delta\cdot X_i}) < \mathrm{ASR}_{orig}$. In this case, the new particle is discarded and the old particle is retained. If no violation occurs, the new particle is retained and update $gbest$ and $pbest$.

E. Elite Pool

To address the premature convergence issue encountered during the search for optimal neuron combinations, the BSEPSO maintains an elite pool ε of fixed size. A newly generated solution X_{new} is only added to the elite pool if it satisfies both of the following conditions. It includes the fitness $F(X_{new})$ exceeds the minimum fitness in the current elite pool and the hamming distance ratio (HDR) with all existing solutions in the elite pool satisfies $HDR(X_{new}, \varepsilon) > \tau$, where τ is the diversity threshold. The *HDR* is defined as

$$HDR(X_{new}, \varepsilon) = \frac{1}{|\varepsilon|}\sum_{e\in\varepsilon}\frac{\sum_{j=1}^{N}|x_{new,j} - x_{e,j}|}{N}, \quad (18)$$

where X_e is a particle in the elite pool. $x_{new,j}$ and $x_{e,j}$ represent the jth-position values of the new particle and the elite pool particle X_e. This process continuously introduces new viable solutions into the elite pool, providing high-quality templates for subsequent weak particle reboots.

Every $reset_{period}$ evaluations (for about 10 generations), the algorithm identifies the lowest-fitness particles in the population and restarts them to avoid wasted computation in poor regions. Specifically, a high-quality solution is randomly selected from the elite pool ε as a template, and a small number of random bit flips are applied to its binary encoding. This preserves the core genetic traits of elite solutions while maintaining exploratory diversity. The restarted particles then replace the original weak particles in the population.

F. Selective Domain Fixed

Subpopulation SSu needs to search more selected neurons and SSl needs to search less selected neurons [22].In SSu, if all neurons in a neighborhood with $pbest$ are selected, the entire neighborhood state is fixed to 1. In subsequent iterations, neurons within this neighborhood no longer change. In SSl, if all neurons in a neighborhood with $pbest$ are unselected, a partial fixation strategy is employed: based on the current iteration progress, a fixed proportion of all-0 neighborhood states are fixed to 0. The fixed ratio *fixratio* calculation equation is as

$$fixratio = r_{min} + (r_{max} - r_{min}) \cdot min\,(1.0, \frac{g}{G_{max}}). \quad (19)$$

The r_{min} is the minimum fixed ratio and the r_{max} is the maximum fixed ratio.

G. Comprehensive Algorithm

The complete pseudocode of the BSEPSO is given as **Algorithm 1**.

Algorithm 1 BSEPSO

Input: The original model $f(\theta)$, the allowed CA drop ΔCA, the tolerable ASR drop threshold σ, the particle x.
Output: The enhanced model $f(\theta^*)$, the globally best particle $gbest$, the perturbation value δ^*.

Begin
$swarm = [random\ 0/1\ vector\ with\ N_{min} \leq ||\cdot||_0 \leq N_{max}]_1^N$;
$pbest = swarm$, $current\_gen = 0$, $fe\_count = 0$;
$fitness\_pbest = [CalculateFitness(x)\ for\ x\ in\ swarm]$;
$gbest = argmax(fitness\_pbest)$, $dir\_list = [random\ 0/1]_1^N$;
$s\_list = [0]_1^N$, $best\_solution = None$, $best\_asr = ASR_{orig}$;
While *fe_count* $< G_{max}$ **do**
  $new\_swarm = []$; δ_g = max (Eqs. (9)), δ_{min});
  For *i*=1 to *N* **do**
    mask = DomainFixed(*swarm*[*i*], *pbest*[*i*], *gbest, dir_list*[*i*], *T*) ;
    new_p=UpdatePosition(*swarm*[*i*],*pbest*[*i*],*gbest,dir_list*[*i*],*mask*) ;
    fe_count+= 1; $\widetilde{\mathrm{ASR}}$= $QuickEvaluate(new\_p, \mathcal{D}_{sub}^{p}, \delta_g)$;
    $\widetilde{\mathrm{CA}}$= $QuickEvaluate(new\_p, \mathcal{D}_{sub}^{c}, \delta_g)$;
    $fit\_new = CalculateFitness(new\_p, \widetilde{\mathrm{ASR}}, \widetilde{\mathrm{CA}})$;
    IF ($fit\_new > fitness\_pbest[i]$) **AND** ($\widetilde{\mathrm{ASR}} > \mathrm{ASR}_{orig}$) **AND** ($\widetilde{\mathrm{CA}} \geq \mathrm{CA}_{orig} - \Delta\mathrm{CA}$) **Then**
      Apply δ_g *to* $f(\theta)$; $\mathrm{CA} = FullEvaluate(new\_p, D^c, \delta_g)$;
      $\mathrm{ASR} = FullEvaluate(new\_p, D^p, \delta_g)$; *Restore*;
      IF ($\mathrm{CA} \geq \mathrm{CA}_{orig} - \Delta\mathrm{CA}$) **AND** ($\mathrm{ASR} \geq \mathrm{ASR}_{orig}$) **AND** ((*best_solution* == *None*) **OR** ($\mathrm{ASR} \geq best\_asr - \sigma$)) **Then**
        $pbest[i], fitness\_pbest[i], s\_list[i] = new\_p, fit\_new, 0$;
        $gbest, best\_solution, best\_ASR = new\_p, \delta_g, \mathrm{ASR}$;
      End if
      IF (*fit_new* > *fitness_pbest*[*i*]) **OR** ($\widetilde{\mathrm{CA}} \geq \mathrm{CA}_{orig} - 0.1$) **Then**
        $new\_swarm.append(new\_p)$;
      ELSE
        $new\_swarm.append(swarm[i])$;
      End if
    End for
    $swarm = new\_swarm$;
    IF $current\_gen\%reset_{period} == 0$ **Then**
      $swarm = ResetFromElite(swarm)$;
    End if
    $current\_gen$+= 1
  End while
  IF *best_solution* != *None* **Then**
    Apply best_solution to $f(\theta)$;
    IF ($\widetilde{\mathrm{CA}} < \mathrm{CA}_{orig}$) **AND** ($\mathrm{ASR} < \mathrm{ASR}_{orig}$) **Then**
      Restore;
      $best\_solution = None$;
    End if
  End if
  $Return\ (f(\theta), gbest, best\_solution)$;
End

TABLE I
PERFORMANCE OF 15 ATTACK MODELS ENHANCED BY THE BSEPSO ALGORITHM ON CIFAR-10 AND TINY-IMAGENET

Attack Method	CIFAR-10				Tiny-ImageNet			
	ASR(%)		CA(%)		ASR(%)		CA(%)	
	Original	Optimized	Original	Optimized	Original	Optimized	Original	Optimized
BadNets	94.44(0.76)	**97.02(0.54)(+)**	91.66(0.17)	90.48(0.43)(–)	99.97(0.03)	99.97(0.03)(=)	37.86(0.25)	37.86(0.25)(=)
Blended	99.86(0.10)	99.86(0.10)(=)	93.51(0.20)	93.51(0.20)(=)	99.64(0.03)	**99.66(0.06)(+)**	38.20(0.56)	37.82(0.60)(–)
Blind	99.87(0.10)	**99.96(0.04)(+)**	82.82(2.32)	81.98(2.10)(–)	19.99(1.70)	**99.87(0.39)(+)**	0.55(0.02)	0.50(0.00)(–)
Bpp	96.53(2.39)	**99.63(0.47)(+)**	90.97(0.72)	90.54(0.72)(–)	99.02(0.46)	99.02(0.46)(=)	41.46(0.31)	41.46(0.31)(=)
Sig	97.81(0.06)	**98.26(0.65)(+)**	84.59(0.09)	84.00(0.88)(–)	NA	NA (=)	NA	NA (=)
Wanet	94.85(1.09)	**95.37(1.20)(+)**	88.62(3.35)	87.93(2.95)(–)	99.80(0.13)	**99.83(0.08)(+)**	40.34(0.74)	40.06(0.49)(–)
Ctrl	99.52(0.06)	**99.62(0.14)(+)**	84.85(0.10)	84.64(0.39)(–)	NA	NA (=)	NA	NA (=)
Ftrojan	100.00(0.01)	100.00(0.01)(=)	93.58(0.07)	**93.60(0.06)(+)**	71.29(49.30)	**72.16(41.46)(+)**	35.89(2.49)	35.60(2.41)(–)
Input-aware	96.47(1.86)	**98.47(0.78)(+)**	89.99(0.70)	89.10(0.59)(–)	99.85(0.08)	**99.88(0.08)(+)**	38.88(0.31)	38.55(0.54)(–)
Lc	99.46(0.24)	99.46(0.24)(=)	84.71(0.21)	84.71(0.21)(=)	NA	NA (=)	NA	NA (=)
Lf	99.10(0.09)	**99.15(0.10)(+)**	93.26(0.21)	92.87(0.79)(–)	99.98(0.0150)	**99.98(0.0145)(+)**	37.16(0.15)	37.05(0.40)(–)
Lira	83.77(14.39)	**86.08(11.89)(+)**	34.69(21.88)	**35.19(19.32)(+)**	94.26(2.10)	**100.00(0.00)(+)**	0.50(0.00)	0.50(0.00)(=)
Refool	94.00(0.23)	**94.06(0.18)(+)**	92.17(0.23)	91.92(0.62)(–)	97.59(0.74)	**98.18(0.49)(+)**	36.26(0.29)	35.06(0.58)(–)
Ssba	97.63(0.58)	97.63(0.58)(=)	92.99(0.21)	92.99(0.21)(=)	98.92(0.15)	**98.95(0.13)(+)**	37.45(0.21)	36.75(0.80)(–)
Trojan	100.00(0.01)	**100.00(0.00)(+)**	93.38(0.09)	92.83(0.75)(–)	99.97(0.01)	99.97(0.01)(=)	37.98(0.40)	37.98(0.40)(=)
+/=/–	11/4/0		2/3/10		9/6/0		0/7/8	

"+", "=", and "-" respectively indicate that the BSEPSO algorithm achieves higher, equal, or lower performance than the original ASR/CA models after enhancing various attack models across different datasets.

IV. EXPERIMENTAL STUDIES

A. Experimental Design

Since most current backdoor attack studies conduct experiments on the two representative datasets CIFAR-10 [25] and Tiny-ImageNet [26], this paper also employs them..

The experiments are conducted on CIFAR-10 and Tiny-ImageNet using PreActResNet-18 [27] and ResNet-18 [28] as victim models, respectively. These models are trained for 100 and 50 epochs with a consistent poisoned ratio of 10%, effectively balancing training efficiency and performance across datasets of varying complexities.

The experiment selects 15 classic and representative backdoor attack models, including BadNets, Blended, Blind, Bpp, Sig, Wanet, Ctrl, Ftrojan, Input-aware, Lc, Lf, Lira, Refool, Ssba, and Trojan [3]. These models represent foundational core methods in the backdoor attack domain [1][2][3], encompassing mainstream attack paradigms such as data poisoning, model fine-tuning and implicit triggering. This enables the evaluation of BSEPSO's optimization performance across diverse attack scenarios, thereby avoiding biased conclusions derived from single-paradigm approaches.

Key parameters for the experimental setup are the penalty factor λ is 6. The Predefined fitness threshold η is $\mathrm{CA}_{orig} - 0.02$. The maximum and minimum selectable neuron number N_{max} and N_{min} are 100 and 0. The ASR drop threshold σ is set to 0.0005. The total number of particles N is set to 30 and the maximum evaluation count G_{max} is set to 1000. The domain window size T is 3. The size of the elite pool ε is 8. In the experiments, each attack model is run 3 times on each dataset and each obtained attack model result undergoes 3 rounds of BSEPSO backdoor attack enhancement.

B. Experimental Results

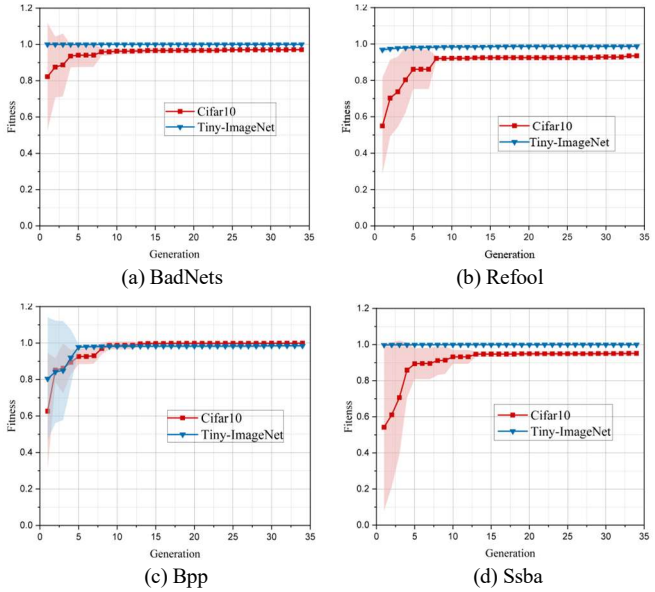
The 15 classic backdoor attack models involved in the experiments are reproduced using the benchmarking platform [3]. Table I illustrates the changes in ASR and CA across 15 attack models on two datasets. In Table I, The black bolded results indicates that ASR or CA performance achieves improvement after enhancement. NA denotes failure to meet the poisoning rate requirement [3]. Results are shown as mean "mean (standard deviation)".

Experimental results demonstrate that the BSEPSO algorithm achieves overall enhancement of backdoors across 15 attack models. Its core performance manifests as consistently elevated or sustained high ASR values, the standard deviation of ASR for each attack model also decreases, CA loss is controlled within reasonable bounds, and there is no substantial degradation in normal performance due to attack enhancement. It improves the generalization capability of attack models. Among the 15 models, 14 demonstrate improved ASR performance on at least one dataset. Meanwhile, the CA of the vast majority of models does not show a significant decline after augmentation. Only a few models exhibit minor fluctuations, while the CA loss of the remaining models is kept within 2%. Surprisingly, the CA of some attack models even increases.

The results also reveal that BSEPSO demonstrates particularly remarkable performance recovery for models characterized by high randomness and unstable baseline attack effectiveness. For instance, the original ASR of Blind on the Tiny-ImageNet dataset is only 19.99%, but after enhancement by BSEPSO, it reaches 99.87%. This holds significant implications for adjusting and restoring attack models with limited execution attempts when they randomly produce bad results.

Furthermore, the experimental results demonstrate that the effectiveness of BSEPSO is independent of the dataset complexity. Whether on the relatively straightforward CIFAR-10 dataset or the more challenging Tiny-ImageNet dataset, BSEPSO consistently achieves significant backdoor enhancement across the majority of attack methods.

To further elucidate the optimization dynamics and performance advantages of the BSEPSO algorithm, this paper selects four representative attack models BadNets, Bpp, Refool and Ssba from a set of 15 attack models. Experimental convergence curves are plotted, as shown in Fig. 1.



(a) BadNets (b) Refool

(c) Bpp (d) Ssba

Fig. 1 : The change curve of fitness for the BSEPSO on 2 different datasets from evaluation times 0 to 1000.

Fig. 1 clearly shows that all models converge rapidly and stabilize at optimal fitness levels before iteration 15. The shaded areas in the figure represent standard deviations, indicating initial fluctuation differences among models across datasets due to variations in initial fitness quality. However, these fluctuations rapidly diminish as iterations progress.

V. Conclusion

This paper transforms the EBAP into a 0-1 integer programming problem and constructs a corresponding mathematical model. To address this problem, we propose BSEPSO, which employs the BSE mechanism to achieve an efficient balance between efficiency and accuracy, successfully resolving the EBAP. Experimental results fully validate that BSEPSO significantly improves the ASR of various attack models while strictly controlling CA loss within preset thresholds. It demonstrates outstanding optimization performance and generalization capabilities across multiple datasets and different attack paradigms.

In the future, the field of backdoor attack enhancement holds vast exploration potential. Subsequent research will overcome the limitations of single-neuron parameter perturbations by exploring composite enhancement pathways that synergistically optimize model fine-tuning and poisoned sample injection, further strengthening the correlation between backdoor features and target labels.

References

[1] Y. Bai *et al*., "Backdoor attack and defense on deep learning: A survey," *IEEE Trans. Comput. Soc. Syst.*, vol. 12, no. 1, pp. 404-434, Feb. 2025.

[2] Y. Li, Y. Jiang, Z. Li, and S. -T. Xia, "Backdoor learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 5-22, Jan. 2024.

[3] B. Wu et al., "Backdoorbench: a comprehensive benchmark and analysis of backdoor learning," *Int. J. Comput. Vis.*, vol. 133, no. 8, pp. 5700–5787, Aug. 2025.

[4] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: defending against backdooring attacks on deep neural networks," in *Proc. Research in Attacks, Intrusions, and Defenses*, 2018, pp. 273–294.

[5] J. S. Borrero, C. Gillen, and O. A. Prokopyev, "Fractional 0–1 programming: applications and algorithms," *J. Glob. Optim.*, vol. 69, no. 1, pp. 255–282, 2017.

[6] Z. H. Zhan, J. Y. Li, S. Kwong, and J. Zhang, "Learning-aided evolution for optimization," *IEEE Trans. Evol. Comput.*, vol. 27, no. 6, pp. 1794–1808, Dec. 2023.

[7] Z. H. Zhan et al., "Matrix-based evolutionary computation," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 2, pp. 315–328, Apr. 2022.

[8] Z. H. Zhan et al., "Artificial intelligence-based methods for protein structure prediction: A survey," *Artif. Intell. Rev.*, vol. 58, Art. no. 328, 2025.

[9] Y. Jiang, X. X. Xu, M. Y. Zheng, and Z. H. Zhan, "Evolutionary computation for unmanned aerial vehicle path planning: A survey," *Artif. Intell. Rev.*, vol. 57, Art. no. 267, 2024.

[10] Z. G. Chen, Z. H. Zhan, S. Kwong, and J. Zhang, "Evolutionary computation for intelligent transportation in smart cities: A survey," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 83–102, May 2022.

[11] Q. T. Yang, X. X. Xu, Z. H. Zhan, J. Zhong, S. Kwong, and J. Zhang, "Evolutionary multitask optimization for multiform feature selection in classification," *IEEE Trans. Cybern.*, vol. 55, no. 4, pp. 1673–1686, Apr. 2025.

[12] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.

[13] Q. T. Yang, Z. H. Zhan, X. F. Liu, J. Y. Li, and J. Zhang, "Grid classification-based surrogate-assisted particle swarm optimization for expensive multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 28, no. 6, pp. 1867–1881, Dec. 2024.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[15] Q. T. Yang, J. Y. Li, Z. H. Zhan, Y. Jiang, Y. Jin, and J. Zhang, "A hierarchical and ensemble surrogate-assisted evolutionary algorithm with model reduction for expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 29, no. 6, pp. 2491–2505, Dec. 2025.

[16] A. Lambora, K. Gupta, and K. Chopra, "Genetic algorithm- a literature review," in *Proc. Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput.*, 2019, pp. 380–384.

[17] X. X. Xu et al., "A receding horizon control-based holistic ant colony system approach for multi-runway aircraft arrival sequencing and scheduling," *Memet. Comput.*, vol. 17, Art. no. 16, 2025.

[18] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[19] J. Y. Li, Z. H. Zhan, and J. Zhang, "Evolutionary computation for expensive optimization: A survey," *Mach. Intell. Res.*, vol. 19, no. 1, pp. 3–23, Feb. 2022.

[20] J. Y. Li et al., "Coevolutionary multi-objective discrete particle swarm optimization for gateway placement optimization problem," in *Proc. Int. Conf. Adv. Comput. Intell.*, 2025, pp. 265–272.

[21] H. Samma, C. P. Lim, and J. Mohamad Saleh, "A new reinforcement learning-based memetic particle swarm optimizer," *Appl. Soft Comput.*, vol. 43, pp. 276–297, Jun. 2016.

[22] J. Q. Yang, Z. H. Zhan, Y. Li, K. C. Tan, and J. Zhang, "Bi-directional feature fixation-based particle swarm optimization for large-scale feature selection," *IEEE Trans. Big Data*, vol. 9, no. 3, pp. 1004–1017, June 2023.

[23] J. Zhang et al., "Evolutionary computation meets machine learning: A survey," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 68–75, Nov. 2011.

[24] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. Int. Conf. Genetic Algorithms*, 1985, pp. 93–100.

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Tech. Rep.*, 2009.

[26] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks,"in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.