Science, Engineering and Technology Vol. 6, No. 2, pp. 68–94
www.setjournal.com https://doi.org/10.54327/set2026/v6.i2.372

Metaheuristic Hyperparameter Optimization for AdaBoost-Based Insider Threat Detection: A Comparative Study on CERT Logon Activity

Snezana Anetic[1], Svetlana Andjelic[1], Milos Grubjesic[1], Mahmoud Abdel-Salam[2], Vladimir Simic[3,4,5], Samson Omasirichukwu Offorjindu[1], Milos Antonijevic[1], Miodrag Zivkovic[1], Nebojsa Bacanin[1,6,7], and Komlen Lalovic[8]

[1]*Faculty of Informatics and Computing, Singidunum University, Danijelova 32, Belgrade 11000, Serbia.*
[2]*Faculty of Computers and Information Science, Mansoura University, Mansoura, Egypt.*
[3]*University of Belgrade, Faculty of Transport and Traffic Engineering, Vojvode Stepe 305, 11010 Belgrade, Serbia.*
[4]*Department of Industrial Engineering, Faculty of Engineering, Dogus University, 34775 Umraniye, Istanbul, Türkiye.*
[5]*Faculty of Engineering and Technology, Sunway University, No. 5, Jalan Universiti, Bandar Sunway, 47500 Selangor Darul Ehsan, Malaysia.*
[6]*Department of Mathematics, Saveetha School of Engineering, SIMATS, Thandalam, Chennai, Tamilnadu, 602105, India.*
[7]*Department of Pathological Physiology, Sechenov First Moscow State Medical University (Sechenov University), Trubetskaya St., 8, Moscow, 119048, Russia.*
[8]*Faculty of Information Technology and Engineering, Union Nikola Tesla University, Cara Dušana 62–64, 11158 Belgrade, Serbia.*

Abstract

Insider threats are hard to detect because malicious actions are rare and closely resemble ordinary activity. This study asks how much of the achievable detection performance comes from tuning a classifier's hyperparameters, and how much depends on which optimizer performs the tuning. AdaBoost is applied to logon activity from the publicly available synthetic Computer Emergency Response Team (CERT) Insider Threat Test Dataset. The majority class was down-sampled to a 9:1 ratio, giving 1,980 observations split 70:30, and nine metaheuristics were compared under an identical budget of 1,020 objective evaluations across 30 independent runs, with hyperparameters selected by three-fold cross-validation on the training partition only and the held-out partition used once for final evaluation. Tuning is decisive: AdaBoost with default parameters reaches a Cohen's kappa of 0.281 and recovers 14 of 59 insider events, whereas tuned configurations reach approximately 0.59. The choice of optimizer matters far less. Only 0.0151 of Cohen's kappa separates the strongest from the weakest of the nine, and although the differences are statistically detectable, the ranking is largely stable across budgets. A hybrid Crayfish Optimization Algorithm variant combining quasi-reflective learning with a firefly-inspired step is also evaluated; it improves on the original algorithm but does not reach the leading methods. Analysis of false negative rates and precision-recall behaviour shows that a model of this kind is best deployed as one risk signal within a wider access-control pipeline rather than as a standalone control.

Keywords: AdaBoost, hyperparameter optimization, insider threat detection, metaheuristic optimization, comparative evaluation, explainable artificial intelligence.

1. Introduction

Intrusion detection is now a central safeguard in complex information systems [1]. Insider-driven ransomware incidents and data breaches can cause severe financial and reputational damage, in some cases threatening the survival of an organization. Because the cybersecurity

Corresponding author: Nebojsa Bacanin (nbacanin@singidunum.ac.rs)
Received: 14 May 2026; Revised: 16 August 2026; Accepted: 27 August 2026; Published: 7 September 2026

landscape changes continuously, security teams and system administrators frequently struggle to respond to newly emerging attack methods and technologies [2].

Artificial intelligence (AI) offers one way to respond to the rapidly changing conditions of the digital security environment. AI-based methods learn recurring structures from observed data and can adapt their behaviour without requiring every decision rule to be explicitly programmed. As additional observations become available, these methods can update their learned patterns and respond to new operating conditions and threats [3].

Applying AI effectively in cybersecurity is nevertheless difficult. Real attack data are limited because organizations are often unwilling to disclose security incidents, which restricts access to representative datasets. Another difficulty is the choice of model parameters. Although learning algorithms are designed to perform reasonably across many tasks, task-specific performance often depends on careful tuning over a large configuration space. Searching this space can be computationally intensive and may constitute a nondeterministic polynomial-time hard (NP-hard) optimization problem. Metaheuristic methods are therefore frequently used to explore parameter combinations through stochastic, often nature-inspired search. They can produce high-quality solutions in practical time, but they cannot guarantee a global optimum.

Metaheuristic algorithms are among the methods most frequently selected for hyperparameter tuning [4], [5], [6]. They have produced competitive results on NP-hard configuration problems under realistic computational budgets in fields ranging from intrusion detection to medical signal analysis and renewable-energy forecasting [5], [6], [7], [8], [9]. The configuration they return is nevertheless a high-quality approximation rather than a guaranteed global optimum [10].

The objective of this study is to examine AdaBoost as a detector of insider threats in organizational settings. AdaBoost was selected because it performs well in binary classification and combines several weak learners into a stronger ensemble. During training, greater emphasis is placed on samples that are difficult to classify, enabling subsequent learners to focus on problematic observations. This property is valuable for insider-threat data, where malicious behaviour is rare and must be distinguished from a much larger volume of normal activity.

The experiments use a publicly accessible synthetic cybersecurity dataset and analyse login behaviour to identify malicious users. The study also develops a tailored modification of the recently proposed Crayfish Optimization Algorithm (COA). Preliminary small-scale comparisons with several optimizers indicated that COA was a promising baseline, which motivated its selection for further modification and detailed evaluation.

The study is organized around three questions. The first asks how much is gained by tuning the control parameters of AdaBoost at all, relative to default settings and to conventional classifiers applied to the same features. The second asks whether the choice of optimizer materially affects the outcome once every method is given the same evaluation budget and the same selection protocol. The third asks whether a purpose-built hybrid of the Crayfish Optimization Algorithm, combining quasi-reflective learning with a firefly-inspired attraction step, improves on the original algorithm and on established alternatives. The study is deliberately comparative rather than advocative: the aim is to establish where the achievable performance actually comes from, and the answer to the third question is reported as it was found.

The principal contributions of the study are summarized below:

- nine metaheuristic optimizers are compared for AdaBoost hyperparameter tuning on insider-threat data under an identical evaluation budget and an identical cross-validated selection protocol;
- the contribution of hyperparameter tuning itself is separated from the contribution of the particular optimizer used, by comparing every tuned model against conventional classifiers with default settings;
- a hybrid variant of the Crayfish Optimization Algorithm is introduced and evaluated, and its two mechanisms are separated by ablation;
- the sensitivity of the comparison to the search budget

is examined by repeating it at two budgets differing by a factor of more than six; and
- the resulting models are assessed with imbalance-aware, security-relevant measures and their operational role is discussed.

The contribution of this work is comparative rather than algorithmic. Studies that apply metaheuristic hyperparameter optimization to security classification problems usually introduce a single new variant and report that it outperforms a set of competitors, without separating the benefit of tuning from the benefit of the particular optimizer, and often without a selection protocol that keeps the evaluation partition genuinely unseen. The design adopted here addresses both points: an identical budget and an identical cross-validated selection procedure are applied to every method, conventional untuned classifiers are included as a reference point, and the experiment is repeated at two budgets to test whether the conclusions depend on that choice. The hybrid variant proposed in Section 3 is evaluated on the same terms as every other method rather than being positioned as the outcome the study is intended to support.

The remainder of this article is organized as follows. Section 2 reviews work on insider-threat detection, machine-learning-based intrusion detection, metaheuristic hyperparameter optimization and complementary trusted-access architectures, and states the research gap addressed here. Section 3 describes the original COA, introduces the proposed hybrid variant and presents its pseudocode. Section 4 presents the dataset, the experimental protocol and the results. Section 5 concludes the study, states its limitations and outlines directions for future work.

Nomenclature

N	population size of the optimizer (number of candidate solutions)
T	maximum number of optimization iterations
t	current iteration index
D	dimensionality of the search space
lb, ub	lower and upper bounds of the search space
X_i	position of candidate solution (agent) i
X_G	best position found so far during the run
X_L	best position in the current population
X_{shade}	position of the shelter in the COA exploration phase
X_{food}	position of the food source in the COA exploitation phase
Q	food-size factor of COA
$temp$	simulated temperature controlling the COA phase transition
C_1, C_2, C_3	COA control parameters
μ, σ	preferred temperature and its spread in the COA feeding function
ψ	activation threshold of the firefly-inspired step in HCOA
β_0, γ, α	firefly attraction, light-absorption and randomization parameters
$r_{i,j}$	Euclidean distance between agents i and j
ε_t	weighted error of AdaBoost weak learner t
α_t	ensemble coefficient of AdaBoost weak learner t
$\omega_{i,t}$	weight of training observation i at AdaBoost iteration t
$h_t(\cdot)$	prediction of AdaBoost weak learner t
y_i	true class label of observation i
κ	Cohen's kappa, the optimization objective
p_o, p_e	observed and chance-expected agreement in Cohen's kappa

Abbreviations

AB-	AdaBoost tuned by the optimizer named after the prefix
AI	artificial intelligence
AUC	area under the curve
CERT	Computer Emergency Response Team
COA	Crayfish Optimization Algorithm
COLSHADE	constrained L-SHADE with Lévy flights
CV	cross-validation
FA	Firefly Algorithm

FN	false negative
FNR	false negative rate
FP	false positive
FPR	false positive rate
GA	Genetic Algorithm
HCOA	Hybrid Crayfish Optimization Algorithm
IAM	identity and access management
IIoT	Industrial Internet of Things
IoT	Internet of Things
MCC	Matthews correlation coefficient
NAC	network access control
NP-hard	nondeterministic polynomial-time hard
PR-AUC	area under the precision–recall curve
PSO	Particle Swarm Optimization
QRL	quasi-reflective learning
ROC	receiver operating characteristic
RSA	Reptile Search Algorithm
SAMME	Stagewise Additive Modeling using a Multi-class Exponential loss function
SCA	Sine Cosine Algorithm
SHAP	SHapley Additive exPlanations
SIEM	security information and event management
SMOTE	synthetic minority oversampling technique
SOAR	security orchestration, automation and response
SVM	support vector machine
TN	true negative
TP	true positive
WOA	Whale Optimization Algorithm
ZTNA	zero trust network access

2. Related Works

Cybersecurity governance combines legal, institutional and organizational safeguards. Monitoring of cyber activity has to be reconciled with privacy and other fundamental rights [11], [12]; jurisdiction is complicated because data may be stored or processed in one state while users, victims and responsible actors are located in others [13], [14]; and cooperation between public agencies and private technology providers can blur established lines of accountability [15], [16]. National strategies such as Australia's 2023–2030 Cyber Security Strategy [17], and platform-level codes of conduct for harmful digital content [18], illustrate the resulting policy landscape. These arrangements define the environment in which insider-threat monitoring is deployed, but they are not the subject of the present study, which is a technical machine-learning contribution. The remainder of this section therefore concentrates on detection methods, on the optimization techniques used to configure them, and on the access-control mechanisms alongside which they operate.

Cybersecurity has traditionally relied on mechanisms such as firewalls [19] and block lists [20]. Although these controls remain useful, rapid technological change and the appearance of zero-day vulnerabilities [21] allow attackers to adapt faster than many administrators. More flexible detection approaches are therefore required.

Internet-of-Things (IoT) environments, and the industrial variant of them known as the Industrial Internet of Things (IIoT), are frequent targets of denial-of-service and distributed denial-of-service attacks [22], because large numbers of relatively simple devices can be used to interrupt services and expose information about users and their surroundings. Organizations must also consider malicious insiders, including employees motivated by resentment or perceived unfair treatment [23].

Insider-threat detection remains insufficiently explored compared with other cybersecurity tasks. This study investigates whether AI-based classification of user behaviour can help prevent harmful actions by insiders. The intended contribution is a more effective approach for protecting organizations against evolving threats that are hidden within otherwise normal activity. Complementing behavioural detection, Korkuc et al. [24] proposed BBNAC, a blockchain-based network access control architecture that integrates Hyperledger Fabric across information-technology and operational-technology environments to validate devices, enforce access policies, and maintain immutable access records. The architecture provides a tamper-evident and auditable access-control layer that

can operate alongside insider-threat detection models by translating behavioural risk signals into controlled network-access decisions.

Research on insider-threat detection has moved from rule-based policy monitoring towards data-driven behavioural modelling. Yuan and Wu [25] survey deep-learning approaches to the problem and identify extreme class imbalance, the scarcity of labelled malicious activity and the need for interpretable decisions as the dominant open challenges. User behaviour analytics builds a per-user or per-role baseline from activity logs and flags deviations from it; Le and Zincir-Heywood [26] show that unsupervised anomaly-detection ensembles applied to the CERT data can identify malicious insiders without labelled examples, at the cost of a high alert volume. Supervised behavioural models trained on the same data source report stronger precision, and Nasir et al. [27] apply recurrent architectures to sequential user activity derived from CERT features. A recurring observation across these studies is that a small number of temporal and access-related attributes, in particular the time and the day of a logon event, already carry a substantial part of the discriminative signal, which motivates the compact feature representation adopted in the present work.

Comparable trends are visible in the wider intrusion-detection literature, where ensemble and deep models trained on network or host telemetry have largely displaced signature matching [3], [5]. Because security analysts have to justify the actions an alert triggers, explainability has become a practical requirement rather than an optional addition. Capuano et al. [28] survey explainable artificial intelligence in cybersecurity and observe that post-hoc attribution methods, of which SHapley Additive exPlanations (SHAP) is the most widely adopted, are the dominant means of exposing the reasoning of an otherwise opaque detector. The present work follows that practice and reports a SHAP analysis of the selected model.

Behavioural detection is normally deployed alongside enterprise access-control mechanisms, and recent work has examined how those mechanisms can be made tamper-evident and auditable. In addition to the BBNAC architecture discussed above [24], Korkuc et al. [29] describe a blockchain-based network access control management solution and architecture, and subsequently extend the approach with an attestation layer intended to support zero trust network access, in which each access request is verified against attested device and identity state rather than against network location [30]. Related work on blockchain-based black-box design addresses tamper-evident recording of security-relevant events [31]. These architectures are complementary to the contribution made here: they govern how an access decision is enforced and recorded, whereas a behavioural classifier of the kind studied in this work supplies the risk signal on which such a decision can be conditioned.

Several modified variants have been proposed since the introduction of COA [32]. Wang et al. [33] combine multiple strategies, including an improved initialization scheme and an adjusted search-control mechanism, to improve performance on numerical benchmark functions, while Chaib et al. [34] introduce an improved COA for parameter estimation of photovoltaic models. These variants are evaluated primarily on benchmark suites or on continuous engineering problems, and they modify the algorithm throughout the entire run. The variant proposed here differs in two respects. The two mechanisms are activated in disjoint phases of the search, so that diversification and intensification are separated in time rather than superimposed, and the quasi-reflective step replaces only the weakest individual and consumes no additional objective-function evaluations, which is a material consideration when each evaluation requires training a complete ensemble classifier.

Table 1 summarizes representative prior work on insider-threat detection, recording for each study the data used, the method applied, the reported outcome and the principal limitation. Two gaps follow from this summary. First, studies that apply machine learning to CERT-derived behavioural data generally adopt default or manually chosen hyperparameters, so it remains unclear how much of the reported performance is attributable to the learning algorithm itself and how much to its configuration. Second, studies

Table 1. Comparative summary of representative prior work on insider-threat detection.

Study	Data used	Method	Reported outcome	Principal limitation
Kandias et al. [23]	Proprietary usage data combined with psychometric profiling	Rule- and profile-based insider prediction model	Demonstrates the feasibility of combining behavioural and psychological indicators	Requires psychometric input that is not present in ordinary system logs
Yuan and Wu [25]	Survey covering CERT and related corpora	Review of deep-learning insider-threat detectors	Consolidates the state of the art	Identifies class imbalance and limited interpretability as unresolved problems
Le and Zincir-Heywood [26]	CERT insider-threat test data	Unsupervised anomaly-detection ensembles	Identifies malicious insiders without labelled examples	Unsupervised setting produces a high alert volume; no hyperparameter optimization
Nasir et al. [27]	CERT-derived behavioural sequences	Recurrent deep learning on sequential user activity	Reports strong detection of anomalous user behaviour	Default hyperparameters; evaluation dominated by accuracy
Mladenovic et al. [6]	Insider-threat text corpus	Metaheuristic-optimized machine-learning classifiers	Tuned models outperform their untuned counterparts	Different modality (text); contribution of individual optimizer components not isolated
Dakic et al. [5]	IoT/IIoT intrusion data	Metaheuristic-optimized machine learning for intrusion detection	Tuned models outperform their untuned counterparts	Not insider-specific; no multiplicity-adjusted statistical comparison
This study	CERT logon and logoff activity	AdaBoost tuned by nine metaheuristics, including the proposed hybrid COA variant	Reported in Section 4	Simulated data, logon features only, restricted search budget

that do apply metaheuristic hyperparameter optimization rarely isolate the contribution of the individual mechanisms of the optimizer they propose, and rarely accompany their comparisons with multiplicity-adjusted significance testing or with imbalance-aware, security-relevant error measures. The present study is positioned against both gaps: it tunes AdaBoost with a purpose-built COA variant on logon-derived features, and reports the resulting models using measures appropriate to a rare-event security task.

AdaBoost [35] constructs an ensemble iteratively to approximate a strong decision rule from multiple weak classifiers. Training begins with weighted observations, and each new learner is influenced by the errors made in earlier rounds. The weighted error of weak learner t is calculated using Eq. (1):

$$\varepsilon_t = \frac{\sum_{i=1}^{N_{tr}} \omega_{i,t}\, \mathrm{I}(h_t(x_i) \neq y_i)}{\sum_{i=1}^{N_{tr}} \omega_{i,t}} \tag{1}$$

In Eq. (1), ε_t is the weighted error at iteration t, N_{tr} is the number of training observations, and $\omega_{i,t}$ is the weight assigned to observation i in that iteration. The expression $h_t(x_i)$ denotes the prediction of the weak learner for sample i, y_i is the corresponding true class, and $\mathrm{I}(\cdot)$ returns 1 when the stated condition is satisfied and 0 otherwise.

The updated observation weights are then used to train the next weak learner, and the procedure continues over successive rounds. The resulting ensemble combines the predictions of many component models through a weighted linear decision rule. Eq. (2) defines the contribution assigned to learner t:

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{2}$$

The coefficient α_t represents the influence of weak learner t in the final ensemble. Its value is derived from the learner's weighted error ε_t, so more accurate learners receive a larger contribution. Observation weights are subsequently updated according to Eq. (3):

$$\omega_{i,t+1} = \omega_{i,t} \exp(-\alpha_t\, y_i\, h_t(x_i)) \tag{3}$$

Here, $\omega_{i,t}$ is the current weight of sample i, α_t is the ensemble coefficient of learner t, y_i is the true label, and $h_t(x_i)$ is the learner's predicted label for that sample.

AdaBoost is particularly appropriate for two-class prediction, although extensions to multiple classes are generally more difficult to configure. Since the task considered in this study is binary, AdaBoost was selected as the classifier to be optimized.

Choosing suitable hyperparameters is often challenging because no universally accepted selection procedure exists. Exhaustive search may be computationally prohibitive, whereas manual trial and error is inefficient. When discrete and continuous parameters are optimized together, the resulting search can become a mixed NP-hard problem, motivating the use of more efficient optimization strategies.

Metaheuristic optimization provides a practical alternative for such search problems. Well-known examples include the Genetic Algorithm (GA) [36], Particle Swarm Optimization (PSO) [37], the Firefly Algorithm (FA) [38], the Sine Cosine Algorithm (SCA) [39], the Whale Optimization Algorithm (WOA) [40], the Reptile Search Algorithm (RSA) [41], and COLSHADE [42]. The diversity of available methods is consistent with the no-free-lunch theorem [10], which indicates that an optimizer that performs well for one problem may not be the best choice for another. Empirical comparison is therefore necessary for each application.

Researchers frequently hybridize established optimizers to compensate for weaknesses in their original search mechanisms. Metaheuristics have already been used in domains including finance [4], medicine [7], [8], computer security [5], [6], and renewable-energy systems [9], among many other applications [43].

3. Materials and Methods

This section first presents the methods that form the basis of the proposed approach. It then explains the introduced modifications and the expected improvements, followed by pseudocode for the complete algorithm.

3.1. Original Crayfish Optimization Algorithm

The Crayfish Optimization Algorithm (COA) is a recent population-based metaheuristic inspired by crayfish behaviour [32]. Crayfish belong to the Astacidea infraorder and are commonly found in freshwater habitats such as rivers and lakes, where they search the bottom for food as omnivorous organisms.

COA models several behavioural patterns. At high temperatures, crayfish search for cooler shelters, a process used to represent exploration. Competition for shelter and feeding under suitable temperatures are modelled as exploitation mechanisms.

As in other swarm-based methods, COA begins by generating a population P of candidate crayfish positions. A randomized temperature variable, defined by Eq. (4), controls the transition between exploration and exploitation behaviours:

$$temp = rand \times 15 + 20 \tag{4}$$

When the simulated temperature exceeds $30\,^{\circ}\mathrm{C}$, crayfish enter the summer-shelter phase and search for cooler locations. Feeding is associated with temperatures between $15\,^{\circ}\mathrm{C}$ and $30\,^{\circ}\mathrm{C}$, with approximately $25\,^{\circ}\mathrm{C}$ treated as favourable. Because reliable foraging is modelled mainly between $20\,^{\circ}\mathrm{C}$ and $30\,^{\circ}\mathrm{C}$, the algorithm samples temperatures from $20\,^{\circ}\mathrm{C}$ to $35\,^{\circ}\mathrm{C}$. Eq. (5) represents the temperature-dependent feeding function:

$$p = C_1 \cdot \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(temp-\mu)^2}{2\sigma^2}\right) \tag{5}$$

In Eq. (5), μ denotes the preferred temperature, whereas C_1 and σ regulate how food intake changes as temperature varies.

If $temp$ is greater than 30, the algorithm activates its exploration behaviour. The target shelter position is computed using Eq. (6):

$$X_{shade} = \frac{X_G + X_L}{2} \tag{6}$$

In this expression, X_L denotes the best position in the current population, while X_G represents the best position found during the optimization process.

A random value determines whether individuals compete for the shelter. When $rand$ is below 0.5, no competition occurs and the candidate moves directly toward the shelter according to Eqs. (7) and (8):

$$X_{i,j}^{t+1} = X_{i,j}^{t} + C_2 \times rand \times \left(X_{shade} - X_{i,j}^{t}\right) \quad (7)$$

$$C_2 = 2 - \frac{t}{T} \quad (8)$$

C_2 decreases over time, T is the maximum number of iterations, t denotes the current iteration, and $t+1$ refers to the following generation.

Under high-temperature conditions, the cave symbolizes the current best solution. Movement toward this shelter reduces the distance between candidate solutions and the optimum, increasing exploitation and accelerating convergence as the search progresses.

When $rand$ is at least 0.5, two crayfish compete for access to the shelter. This interaction initiates the competition behaviour represented by Eqs. (9) and (10):

$$X_{i,j}^{t+1} = X_{i,j}^{t} - X_{z,j}^{t} + X_{shade} \quad (9)$$

$$z = \text{round}(rand \times (N-1)) + 1 \quad (10)$$

The index z identifies a randomly selected crayfish from the population.

During competition, candidate X_i updates its location with respect to the randomly chosen individual X_z. Incorporating another population member enlarges the region considered by the search and therefore supports exploration.

At temperatures of $30\,^{\circ}\text{C}$ or lower, the algorithm models feeding and moves individuals toward the food source. The food position X_{food} and its size Q are defined by Eqs. (11) and (12):

$$X_{food} = X_G \quad (11)$$

$$Q = C_3 \times rand \times \frac{fitness_i}{fitness_{food}} \quad (12)$$

C_3 is the food-size factor and is fixed at 3. The term $fitness_i$ is the objective value of crayfish i, whereas $fitness_{food}$ is the objective value associated with the food location.

When the food is considered too large, that is, when $Q > (C_3+1)/2$, the tearing behaviour is modelled using Eq. (13):

$$X_{food} = \exp\left(-\frac{1}{Q}\right) \cdot X_{food} \quad (13)$$

The crayfish then shreds the rescaled food, a movement modelled by Eq. (14), in which θ is drawn uniformly from $[0, 2\pi]$:

$$X_{i,j}^{t+1} = X_{i,j}^{t} + \cos(\theta)\, X_{food}\, p - \sin(\theta)\, X_{food}\, p \quad (14)$$

For a sufficiently small food source, $Q \leq (C_3+1)/2$, the crayfish consumes it directly as described by Eq. (15):

$$X_{i,j}^{t+1} = \left(X_{i,j}^{t} - X_{food}\right) \cdot p + p \times rand \times X_{i,j}^{t} \quad (15)$$

The foraging update depends on Q, while X_{food} represents the best solution. A candidate approaches the food directly when its size is manageable. If Q is large, reflecting a substantial distance from the best solution, the food-related update is adjusted before the individual moves toward it.

3.2. Hybrid COA

Although the original COA has shown competitive performance, its recent introduction leaves opportunities for improving the search process. The proposed method therefore adds two mechanisms to the baseline algorithm.

The first modification applies quasi-reflective learning (QRL) [44] during the first half of the run, that is, during the first $T/2$ iterations. At the end of each such iteration the worst-performing individual in the population is replaced by a candidate generated through Eq. (16):

$$X_z^{qr} = rand\left(\frac{lb_z + ub_z}{2},\ X_z^{worst}\right) \tag{16}$$

In Eq. (16), X_z^{worst} is the z-th component of the worst-performing individual in the current population, lb_z and ub_z are the corresponding lower and upper search bounds, and $rand(a, b)$ returns a value drawn uniformly from the interval delimited by a and b. The quasi-reflective counterpart is derived from the current worst individual and substituted for it directly. The substitution is performed before the population-wide fitness assessment of that same iteration, so the replacement is scored in exactly the same way as every other agent and the per-iteration evaluation budget of the original method is preserved: the run still consumes N objective-function evaluations per iteration. Should the replacement prove inferior, it simply becomes the worst individual again and is itself eligible for substitution in the next round, so the mechanism can neither displace nor degrade the incumbent best solution.

QRL is included to preserve population diversity during the early search. Standard COA may converge too quickly around a limited region, reducing its ability to leave local optima. Reflection-based candidates distribute search effort toward less-explored areas while keeping the population size unchanged, because they replace weak individuals rather than being added as extra agents.

A successful metaheuristic must balance global exploration with local exploitation. To strengthen the exploitation component, the proposed method incorporates a mechanism inspired by the Firefly Algorithm (FA) [38]. FA was chosen because its attraction-based movement is simple and effective. In the model, agents with lower brightness move toward brighter individuals, and brightness is determined from the problem-specific objective function given in Eq. (17):

$$F_i = f(X_i) \tag{17}$$

The FA model also accounts for attenuation of light with distance and the properties of the transmission medium. Its basic movement equation is provided in Eq. (18):

$$X_i(t+1) = X_i(t) + \beta e^{-\gamma r_{i,j}^2}\left(X_j(t) - X_i(t)\right) + \alpha\varepsilon_i(t) \tag{18}$$

For computational efficiency, the exponential attractiveness term in Eq. (18) is often replaced by the simplified expression in Eq. (19), where β_0 is the attraction value at zero distance:

$$\beta(r) = \frac{\beta_0}{1 + \gamma \times r^2} \tag{19}$$

In these equations, $X_i(t)$ is the location of agent i at iteration t and $r_{i,j}$ represents the separation between agents i and j. The parameter β controls attraction, γ is the light-absorption coefficient, α determines random movement, and $\varepsilon_i(t)$ is a stochastic vector.

Because the FA-based movement can accelerate convergence, it is activated selectively to avoid excessive exploitation. The mechanism is available only during the second half of the run. At each eligible iteration, a random number in $[0, 1]$ is compared with threshold ψ. FA movement is applied when the random value exceeds ψ; otherwise, the standard COA update is retained. The threshold is selected empirically and is set to 0.6.

The resulting method is referred to as the Hybrid Crayfish Optimization Algorithm (HCOA). Its complete procedure is summarized in Algorithm 1.

As Algorithm 1 makes explicit, one run consumes N evaluations for the initial population and N evaluations in each of the T iterations, that is $N\,(T + 1)$ objective-function evaluations in total. The same accounting applies to every compared optimizer and is the basis of the budget figures reported in Section 4.

Algorithm 1 Pseudocode for the proposed HCOA algorithm.

Input: population size N, maximum number of iterations T, threshold ψ, bounds $[lb, ub]$
Output: best solution X_G and its objective value $f(X_G)$

Initialize the population $X = \{X_1, \dots, X_N\}$ at random within $[lb, ub]$
Evaluate every agent with the objective function f $\triangleright$ N evaluations
Set X_G to the best agent found and X_L to the best agent of the current population
$t \leftarrow 1$
while $t \leq T$ **do**
    Draw the simulated temperature $temp$ using Eq. (4)
    if $t \leq T/2$ **then**
        Update all agent positions with the COA mechanisms, Eqs. (5)–(15)
        Identify the worst agent X^{worst} in the current population
        Generate its quasi-reflective counterpart with Eq. (16)
        Replace X^{worst} by that candidate $\triangleright$ no extra evaluation
    else
        Draw a pseudo-random value $rnd \in [0, 1]$
        if $rnd > \psi$ **then**
            Update all agent positions with the firefly attraction mechanism, Eqs. (18) and (19)
        else
            Update all agent positions with the COA mechanisms, Eqs. (5)–(15)
        end if
    end if
    Repair any component that violates $[lb, ub]$
    Evaluate every agent with the objective function f $\triangleright$ N evaluations
    Update X_L and, if improved, the global best X_G
    $t \leftarrow t + 1$
end while
return X_G and $f(X_G)$

4. Experimental Results

To facilitate experimentation, this study uses the publicly available synthetic Insider Threat Test Dataset produced by the CERT (Computer Emergency Response Team) Division of the Carnegie Mellon University Software Engineering Institute [45], [46]. Release r4.2 was used, and the data were accessed on 25 January 2023. Although the dataset records several categories of activity for a number of simulated malicious insiders, the present analysis is restricted to logon and logoff events and their timing. The original extraction contained 854,661 observations of normal user activity and 198 observations of malicious insider activity, a ratio of approximately 4,316:1. To reduce this extreme imbalance while preserving every available minority-class observation, only the majority class was randomly down-sampled; no minority-class observation was discarded and no synthetic minority observation was generated. Down-sampling was carried out before any partitioning of the data and produced 1,782 normal and 198 insider observations, that is a 9:1 class ratio and 1,980 observations in total. The resulting dataset was then divided by a stratified 70:30 split, so that the class proportion was preserved in both partitions. The training partition contained 1,247 normal and 139 insider observations, and the held-out partition contained 535 normal and 59 insider observations.

The selection protocol was as follows, and it is stated here in full because it governs how every figure reported in this section should be read. Each candidate hyperparameter configuration proposed by an optimizer was scored by stratified three-fold cross-validation carried out *within the training partition alone*: the configuration was fitted on two folds and scored on the third, and the mean Cohen's kappa over the three folds was the value the optimizer maximized. No separate validation partition was therefore required, because the validation role is played by the rotating held-out fold of the inner cross-validation. The 30 % held-out partition took no part whatsoever in the search. It was consulted exactly once per run, after the search had terminated, in order to score the single configuration the optimizer had selected. The values reported in Tables 4 and 5 are therefore inner cross-validation scores obtained on the training partition, and the values reported in Tables 6, 11 and 13 are held-out results obtained on data that were never used for fitting or for selection. The ablation reported later in this section was carried out under exactly the same protocol, which

Table 2. Composition of the dataset across the preparation and evaluation stages.

Dataset stage	Normal	Insider	Total	Purpose
Original extraction	854,661	198	854,859	Full logon activity before class balancing
After majority-class down-sampling	1,782	198	1,980	Balanced experimental dataset with a 9:1 class ratio
Training partition (70 %)	1,247	139	1,386	Model fitting and hyperparameter selection
Inner stratified three-fold cross-validation, drawn from the training partition	1,247	139	1,386	Scoring of every candidate configuration during the search; the rotating validation fold replaces a separate validation partition
Held-out test partition (30 %)	535	59	594	Single final evaluation, after the search had terminated

is why its objective values are directly comparable with those in Table 4. Table 2 summarizes the composition of the dataset at each stage, and Figure 1 shows the structure of the resulting training and testing portions.

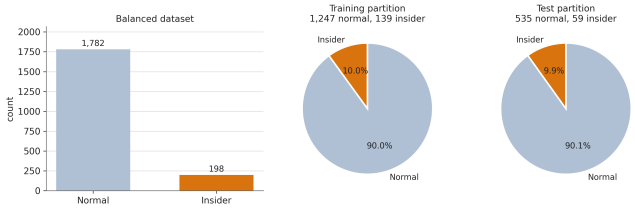

Figure 1. Dataset training and testing structure.

The 9:1 ratio was selected as a compromise between two competing requirements. A ratio closer to 1:1 would have discarded a very large part of the majority class and would have produced a training distribution far removed from operational conditions, in which insider events are rare. Retaining the original ratio, on the other hand, would have left fewer than one positive observation per four thousand negatives, at which point the objective function becomes almost insensitive to the minority class and the optimizer receives very little usable signal. A moderate residual imbalance of 9:1 preserves the rarity of the positive class while leaving enough positive observations in each partition for Cohen's kappa to discriminate between candidate configurations. Oversampling alternatives such as the synthetic minority oversampling technique (SMOTE) were deliberately not applied, because synthesizing additional insider observations from only 198 real ones risks generating interpolated points that correspond to no plausible user behaviour, and because those synthetic samples would then influence the hyperparameter search itself. A systematic sensitivity analysis over the down-sampling ratio, together with a comparison against oversampling and cost-sensitive alternatives, fell outside the computational budget of this study and is identified as future work.

The analysis uses temporal access attributes, specifically the day and time of each event. Login behaviour differs noticeably across these dimensions, as illustrated by the activity distributions in Figure 2.

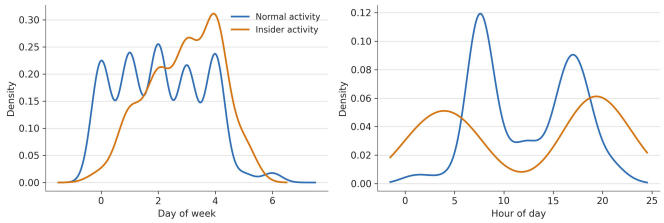

Figure 2. Distributions of normal user activity, shown in blue, and insider threat activity, shown in orange.

The proposed optimizer was compared with COA, GA, PSO, FA, SCA, WOA, RSA, and COLSHADE. This group includes both established and recently introduced optimization approaches. In what follows, a method name carrying the prefix AB- denotes AdaBoost tuned by the optimizer named after the prefix; AB-COLSHADE, for instance, is AdaBoost tuned by COLSHADE, and AB-HCOA is AdaBoost tuned by the proposed hybrid. All methods were implemented independently in Python using scikit-learn, with Pandas and NumPy supporting data manipulation and numerical processing. Parameter settings followed the recommendations in the corresponding original publications and are listed in full in Table 3.

Each optimizer searched for the AdaBoost control parameters: number of estimators in $[10, 50]$, tree depth in

$[1, 10]$, and learning rate in $[0.1, 2]$. The estimator range was kept moderate because repeated model evaluation is computationally expensive. In the main experiment all methods used a population of 20 candidates over 50 optimization iterations, which corresponds to $20 \times (50 + 1) = 1{,}020$ objective-function evaluations per run. That budget was identical for every optimizer, so no method could gain an advantage from being allowed more search. To account for stochastic variation, every experiment was repeated independently 30 times from the same ordered sequence of random seeds, so that all methods encountered the same set of initial populations. The entire comparison was additionally repeated at a reduced budget of a population of 10 candidates over 15 iterations, that is $10 \times (15 + 1) = 160$ evaluations per run, in order to test whether the conclusions depend on the amount of search allowed; the outcome of that comparison is reported at the end of this section.

Both the search space and the main search budget were deliberately modest. Each objective-function evaluation requires training and scoring a complete AdaBoost ensemble on three cross-validation folds, so the main budget of 1,020 evaluations corresponds to roughly three thousand ensemble fits per run and per optimizer, and the full comparison across all optimizers and all runs already represents a substantial computational load. Two consequences should be acknowledged. First, the reported values are not the best configurations of AdaBoost attainable on this dataset; they are the best configurations reachable within a fixed and equal budget, which is what makes a comparison between optimizers meaningful. Second, a narrow interval for the number of estimators limits the variance reduction the ensemble can achieve, so the absolute performance figures are likely to be conservative. Because every optimizer received an identical budget, an identical search space and an identical objective function, these restrictions bear equally on all compared methods.

4.1. Implementation and reproducibility

All experiments were implemented in Python. The classifier, the evaluation measures and the partitioning routines were taken from scikit-learn, with Pandas and NumPy used for data handling and numerical processing, Matplotlib for the figures, and the SHAP package for the interpretability analysis. The reported results were produced with scikit-learn 1.6.1, NumPy 2.4.6, pandas 2.2.3, SciPy 1.15.1, SHAP 0.52.0 and Matplotlib 3.10.0 under Python 3.13.2. The AdaBoost implementation used is the scikit-learn default for that version, namely SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function). All runs were executed on a workstation with an AMD Ryzen 9 9900X processor (12 cores, 24 threads), 32 GB of RAM and Windows 11. The workload is CPU-bound: the AdaBoost ensembles and the optimizers are fitted on the processor, and no GPU acceleration is used at any stage. The independent runs were distributed over separate single-threaded processes, so that the timings reported below are not distorted by contention between them. The train–test partition was generated once with a fixed random seed and stored, so that every optimizer and every run was fitted and evaluated on exactly the same data; the resulting partitions are distributed with the supplementary material. Each independent run of every optimizer was initialized from a distinct seed, and the same ordered sequence of seeds was reused across all optimizers, so that all methods encountered the same set of initial populations. The computational budget was identical for every method and is the one stated above: 1,020 objective-function evaluations per run in the main experiment and 160 per run in the reduced-budget experiment, with each evaluation requiring a complete AdaBoost ensemble to be trained and scored on each of the three inner cross-validation folds. Because this budget is fixed and equal across methods, the number of objective-function evaluations rather than wall-clock time is the quantity that makes the computational cost of the compared optimizers directly comparable. The mean execution time of a single run of the main experiment, averaged over three runs of each method on the workstation described above, was 29 s for COA, 33 s for HCOA, 39 s for RSA, 57 s for WOA, 59 s for PSO, 63 s for FA, 63 s for COLSHADE, 68 s for SCA and 70 s for GA. Every method performs the same 1,020 objective-function

evaluations, and each evaluation trains and scores three AdaBoost ensembles, so these differences do not reflect differing amounts of search. They reflect two other things: the overhead of the search operators themselves, which is largest for the methods that update the population through explicit per-individual loops, and the region of the search space each method tends to occupy, since the cost of one evaluation grows with the number of estimators in the candidate configuration. The proposed hybrid is among the cheaper methods, at roughly one seventh of a second per objective-function evaluation. Because the budget is fixed and equal across methods, the evaluation count rather than wall-clock time remains the quantity that makes the comparison fair.

The control parameters of all comparative algorithms were adopted from the default or recommended settings reported in their original publications [32], [36], [37], [38], [39], [40], [41], [42]. Table 3 consolidates the shared experimental settings and the method-specific control parameters of every optimizer in a single place, so that the experiments can be reconstructed from the article itself. The preprocessed dataset, the exact training and testing partitions, the implementation of all nine optimizers and the scripts that generate every table and figure in this section are available from the deposit identified in the data availability statement.

Cohen's kappa was used as the optimization objective because it is informative for classification with imbalanced classes. Eq. (20) defines the metric:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{20}$$

In Eq. (20), p_o is the observed proportion of agreement between the predicted and the true labels, and p_e is the proportion of agreement expected by chance given the marginal class frequencies. A value of 0 indicates agreement no better than chance and a value of 1 indicates perfect agreement.

Additional measures were recorded to provide a broader assessment of the classifiers. Accuracy, precision, recall, and F1-score are defined in Eqs. (21) through (24), respectively:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{21}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{22}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{23}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

In Eqs. (22) to (24), TP, FP and FN denote the numbers of true positive, false positive and false negative predictions respectively, with the insider class treated as positive. The error rate for each model was also calculated using Eq. (25):

$$\text{Error rate} = 1 - \text{Accuracy} \tag{25}$$

Error rate equals one minus accuracy and measures the fraction of observations assigned to the wrong class. It can therefore be interpreted as the number of misclassifications divided by the total number of evaluated samples.

4.2. Simulation outcomes

Tables 4 and 5 report the objective-function and indicator-function outcomes over the 30 runs, ordered by mean objective value. AB-COLSHADE attained the highest mean (0.587096) and AB-COA the lowest (0.571978). The interval separating the strongest from the weakest of the nine methods is therefore 0.0151 of Cohen's kappa, which is small relative to the run-to-run standard deviation of each method, itself between 0.0129 and 0.0208. Figure 3 shows the corresponding distributions and makes the degree of overlap visible: the nine methods occupy substantially the same range, and the ordering is determined by a modest shift in the centre of each distribution rather than by separation between them.

Convergence behaviour is shown in Figure 4. All nine

Table 3. Consolidated configuration of every optimizer. The shared settings apply identically to all nine methods. The method-specific control parameters are the values used in the experiments reported here; they follow the corresponding source publication except where the implementation is stated explicitly.

Item	Setting	Source
Shared settings, identical for every optimizer		
Population size N	20 (main experiment); 10 (reduced-budget experiment)	this study
Iterations T	50 (main experiment); 15 (reduced-budget experiment)	this study
Evaluation budget	$N\,(T+1)$ evaluations per run, comprising N evaluations of the initial population and N in each iteration: 1,020 (main); 160 (reduced)	this study
Independent runs	30 per method, seeded $1000, 1001, \ldots, 1029$; the same sequence is reused for every method	this study
Problem dimension D	3	this study
Search space	number of estimators $\in [10, 50]$; base tree depth $\in [1, 10]$; learning rate $\in [0.1, 2.0]$. The search is conducted in continuous space; the first two components are floored to integers when a candidate is decoded	this study
Objective function	mean Cohen's kappa over stratified three-fold cross-validation within the training partition, maximized. The held-out partition is never evaluated during the search	this study
Initialization	uniform random sampling within $[lb, ub]$	this study
Boundary handling	components violating $[lb, ub]$ are clipped to the nearest bound	this study
Base learner	decision tree of the searched depth; AdaBoost fitted with the searched number of estimators and learning rate	this study
Method-specific control parameters		
GA	binary tournament selection; blend (BLX-α) crossover with $\alpha = 0.5$ at probability $p_c = 0.9$; Gaussian mutation at rate $p_m = 1/D$ with standard deviation 0.1 of each variable's range; elitism 1	[36]
PSO	constriction-type settings, inertia weight $w = 0.7298$ and acceleration coefficients $c_1 = c_2 = 1.49618$; velocity clamped to 20 % of the range of each dimension; global-best topology	[37]
FA	attractiveness at zero distance $\beta_0 = 1.0$; light absorption $\gamma = 1.0$; randomization α annealed linearly from 0.2 to 0.02 over the run; the simplified attractiveness of Eq. (19) is used in place of the exponential form	[38]
SCA	$a = 2$, with r_1 decreasing linearly from a to 0	[39]
WOA	a decreasing linearly from 2 to 0; a_2 from -1 to -2; logarithmic spiral constant $b = 1$; encircling/spiral switch probability 0.5	[40]
RSA	sensitive parameters $\alpha = 0.1$ and $\beta = 0.005$; evolutionary sense $ES = 2\,r\,(1 - t/T)$ with r drawn uniformly from $\{-1, 0, 1\}$; the four behavioural phases switch at $T/4$, $T/2$ and $3T/4$	[41]
COLSHADE	historical memory size $H = 5$ with initial values $\mu_F = \mu_{CR} = 0.5$; p-best rate 0.3; Lévy mutation applied with probability 0.5 and index 1.5; F drawn from a Cauchy distribution of scale 0.1 and CR from a normal distribution of standard deviation 0.1; binomial crossover; greedy selection	[42]
COA	$C_1 = 0.2$; $C_3 = 3$; $\mu = 25$; $\sigma = 3$; $temp$ sampled uniformly from $[20, 35]$	[32]
HCOA (proposed)	all COA parameters as above; quasi-reflective learning applied to the worst agent once per iteration during iterations $1 \ldots T/2$; the firefly-inspired step available during iterations $T/2{+}1 \ldots T$ and taken when a uniform draw exceeds $\psi = 0.6$, with $\beta_0 = 1.0$ and $\gamma = 1.0$	this study

methods improve rapidly over the first ten iterations and then flatten, which is consistent with a three-dimensional search space that is largely explored well before the budget is exhausted. The proposed hybrid keeps pace with the leading methods over the first few iterations and separates from them thereafter, remaining below AB-COLSHADE for the rest of the run; devoting the first half of the run to diversification does not pay for itself on a search space of this size.

Table 6 reports the per-class metrics of the best model produced by each optimizer, and Table 7 the hyperparam-

Table 4. Objective function outcomes over 30 independent runs.

Method	Best	Worst	Mean	Median	Std	Var
AB-COLSHADE	0.617820	0.546936	0.587096	0.586447	0.017905	3.21E − 04
AB-PSO	0.617989	0.540965	0.586265	0.590125	0.020702	4.29E − 04
AB-GA	0.630581	0.554750	0.586208	0.587081	0.018600	3.46E − 04
AB-FA	0.617244	0.547715	0.585203	0.585070	0.015426	2.38E − 04
AB-WOA	0.618224	0.549540	0.580355	0.579498	0.019916	3.97E − 04
AB-HCOA	0.610504	0.540693	0.577024	0.576585	0.017836	3.18E − 04
AB-SCA	0.600583	0.546224	0.575799	0.576864	0.012942	1.68E − 04
AB-RSA	0.620195	0.531965	0.573621	0.571927	0.019496	3.80E − 04
AB-COA	0.609480	0.528939	0.571978	0.572531	0.020823	4.34E − 04

Table 5. Indicator function outcomes over 30 independent runs.

Method	Best	Worst	Mean	Median	Std	Var
AB-COLSHADE	0.060606	0.089226	0.071268	0.069865	0.005723	3.28E − 05
AB-PSO	0.062290	0.082492	0.070988	0.070707	0.004703	2.21E − 05
AB-GA	0.063973	0.087542	0.071605	0.070707	0.005211	2.72E − 05
AB-FA	0.060606	0.084175	0.070258	0.069865	0.005480	3.00E − 05
AB-WOA	0.062290	0.082492	0.070707	0.069024	0.004661	2.17E − 05
AB-HCOA	0.060606	0.084175	0.069978	0.069024	0.005282	2.79E − 05
AB-SCA	0.060606	0.079125	0.069136	0.069024	0.003687	1.36E − 05
AB-RSA	0.063973	0.080808	0.069641	0.069024	0.004203	1.77E − 05
AB-COA	0.058923	0.082492	0.070090	0.069024	0.004908	2.41E − 05

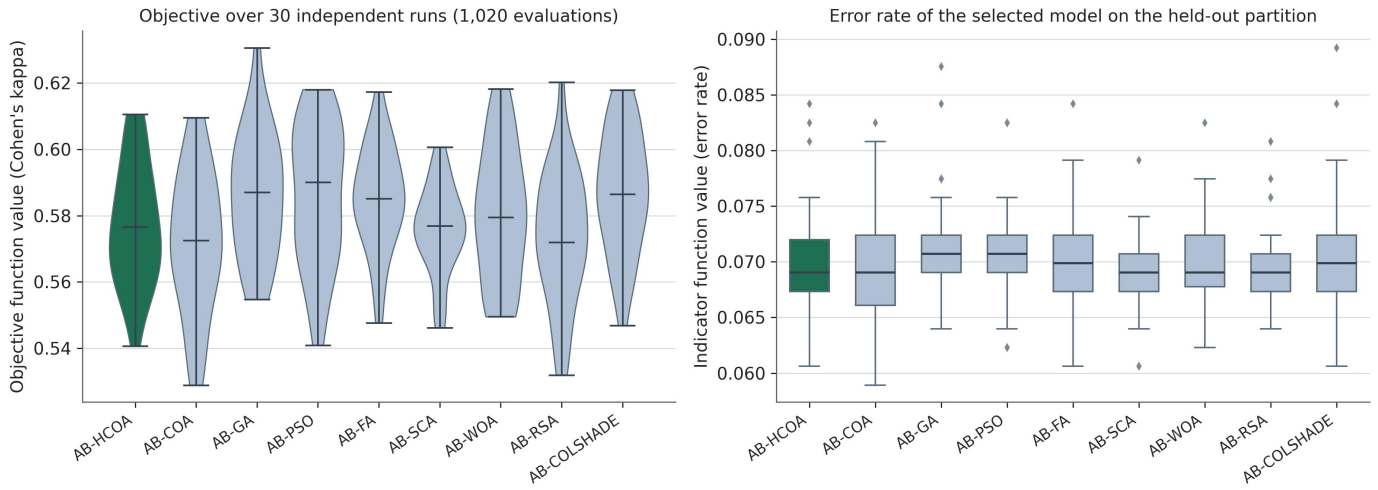

Figure 3. Objective and indicator function outcome distributions over 30 independent runs.

eter configurations selected. A point worth noting is that the ordering by held-out performance does not reproduce the ordering by objective value, and it does not do so to a striking degree. The highest held-out Cohen's kappa (0.621294) was obtained by AB-HCOA, which ranks only sixth of nine by mean objective; AB-COLSHADE, the strongest method by mean objective, reaches 0.565502 on the held-out partition; and AB-GA, third by mean objective, produces the weakest held-out model of all nine at 0.468356. This is possible precisely because the two quantities are measured on different data: the objective is an inner cross-validation score computed on the training partition, whereas the held-out figure is computed on a partition that took no part in the search. With 59 insider events in the held-out partition, a single model's held-out score is subject to considerable sampling noise, and the

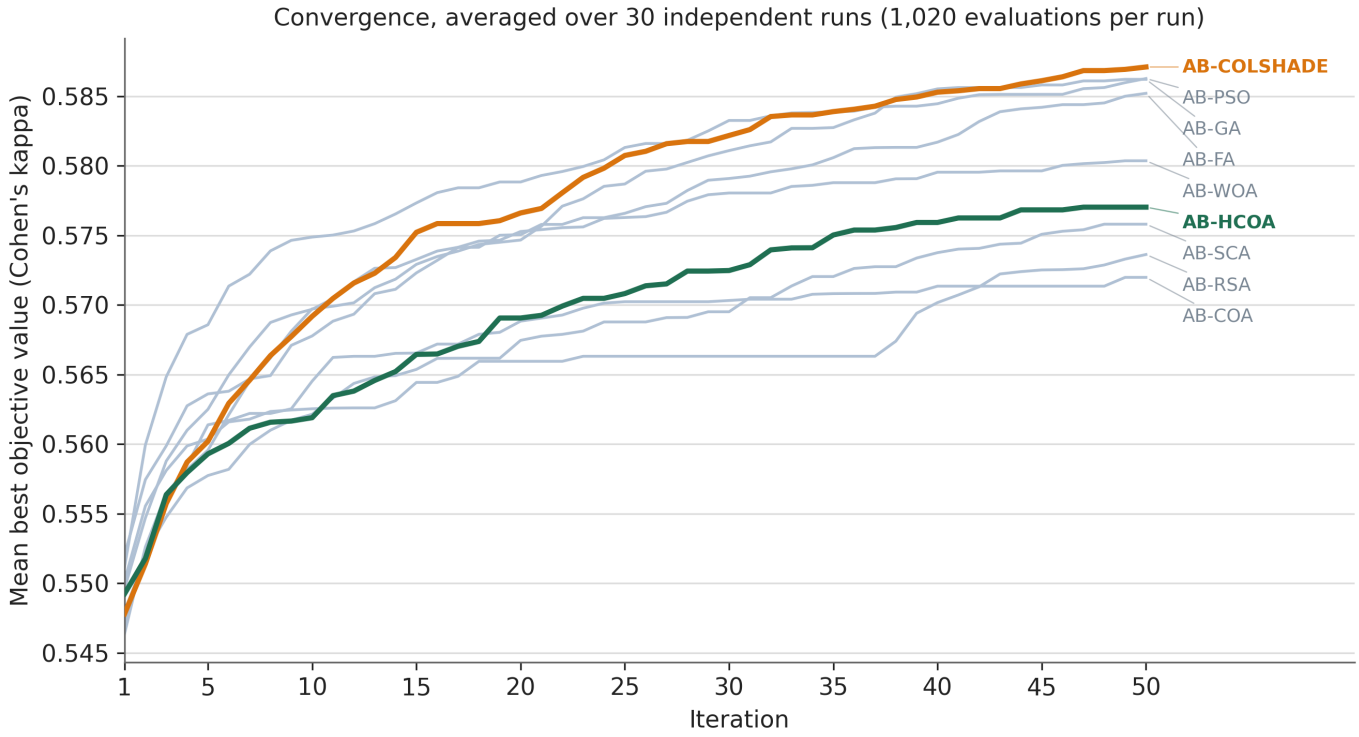

Figure 4. Convergence of the mean best objective value, averaged over 30 independent runs.

mean objective over 30 runs is the more reliable basis for comparing optimizers. This discrepancy is itself an argument against drawing strong conclusions from the single best model of a single run, a practice common in this literature.

Table 7 lists the hyperparameter configuration selected by each optimizer for its best-performing AdaBoost model.

4.3. Sensitivity to the search budget

Because the comparison above allocates a fixed budget, it is reasonable to ask whether the ranking is an artefact of that choice. The entire experiment was therefore repeated with a population of 10 over 15 iterations, that is 160 evaluations per run, approximately one sixth of the budget used above. Table 8 places the two sets of results side by side. Every method improves with the larger budget, as expected, and the interval separating the strongest from the weakest narrows from 0.0175 to 0.0151 of Cohen's kappa. The ordering is largely preserved: AB-PSO and AB-COLSHADE occupy the first two positions at both budgets, in one order or the other, AB-COA is the weakest at both, and no method moves by more than two positions. The conclusion that the choice of optimizer has a modest effect relative to the effect of tuning at all is therefore not an artefact of the budget, and the narrowing of the interval at the larger budget indicates that the differences between methods reflect how quickly each one explores a small search space rather than a difference in the quality of the optimum each is able to reach.

Graphical presentation of the outcomes for the best model in terms of confusion matrix and receiver operating characteristic (ROC) curve is provided in Figure 5. The ROC curve of the best-performing model indicates high discriminative capability with an area under the curve close to 1.0. Moreover, the corresponding confusion matrix demonstrates that the model achieves strong precision and recall for both normal and insider classes. Together, these plots indicate that the selected model separates the two classes well on the held-out data and raises few false alarms on normal activity, while the confusion matrix also shows that a non-negligible proportion of insider events is still

Table 6. Performance of the best model produced by each optimizer on the held-out partition. Kappa is Cohen's kappa; precision, recall, F1 and PR-AUC are computed for the insider class.

Method	Accuracy	Kappa	MCC	Precision	Recall	F1	ROC-AUC	PR-AUC
AB-COLSHADE	0.929293	0.565502	0.569911	0.680851	0.542373	0.603774	0.956487	0.620575
AB-PSO	0.930976	0.572302	0.577638	0.695652	0.542373	0.609524	0.962934	0.639729
AB-GA	0.924242	0.468356	0.494172	0.718750	0.389831	0.505495	0.960684	0.633164
AB-FA	0.932660	0.586192	0.590762	0.702128	0.559322	0.622642	0.962506	0.638077
AB-WOA	0.934343	0.599855	0.603712	0.708333	0.576271	0.635514	0.963045	0.633404
AB-HCOA	0.939394	0.621294	0.628144	0.755556	0.576271	0.653846	0.963250	0.630483
AB-SCA	0.929293	0.558177	0.564330	0.688889	0.525424	0.596154	0.964803	0.658516
AB-RSA	0.936027	0.613295	0.616495	0.714286	0.593220	0.648148	0.960177	0.615163
AB-COA	0.936027	0.619500	0.621498	0.705882	0.610169	0.654545	0.960985	0.632127

Table 7. Hyperparameter configuration selected by each optimizer.

Method	Estimators	Tree depth	Learning rate
AB-COLSHADE	12	6	0.412191
AB-PSO	46	4	0.182664
AB-GA	33	2	1.865548
AB-FA	31	4	0.305018
AB-WOA	48	4	0.182743
AB-HCOA	48	2	1.201142
AB-SCA	28	6	0.100000
AB-RSA	23	4	0.305879
AB-COA	32	2	1.520945

Table 8. Sensitivity of the comparison to the search budget, at 160 and 1,020 objective-function evaluations per run.

	Mean objective		Rank	
Method	160	1,020	160	1,020
AB-COLSHADE	0.570850	0.587096	2	1
AB-PSO	0.571270	0.586265	1	2
AB-GA	0.566567	0.586208	4	3
AB-FA	0.564605	0.585203	5	4
AB-WOA	0.570061	0.580355	3	5
AB-HCOA	0.557277	0.577024	8	6
AB-SCA	0.558475	0.575799	7	7
AB-RSA	0.559879	0.573621	6	8
AB-COA	0.553813	0.571978	9	9

missed. The operational consequences of that asymmetry are discussed below.

4.4. Statistical validation

The objective values recorded across the 30 runs were analysed with non-parametric procedures, the observations being treated as paired because every method was evaluated under identical conditions and on identical partitions. A Friedman omnibus test rejected the global null hypothesis, with $\chi^2(8) = 68.5581$ and $p < 0.000001$, but Kendall's coefficient of concordance was only $W = 0.2857$, indicating weak overall differentiation among the nine methods. Table 9 reports this result. In other words, the differences are real but small, which is precisely the combination that a large number of runs is able to detect and that a small number would not.

Pairwise Wilcoxon signed-rank tests were then carried out between the method with the highest mean objective, AB-COLSHADE, and each of the remaining eight, with the family-wise error rate controlled by the Holm step-down procedure and matched-pairs rank-biserial cor-

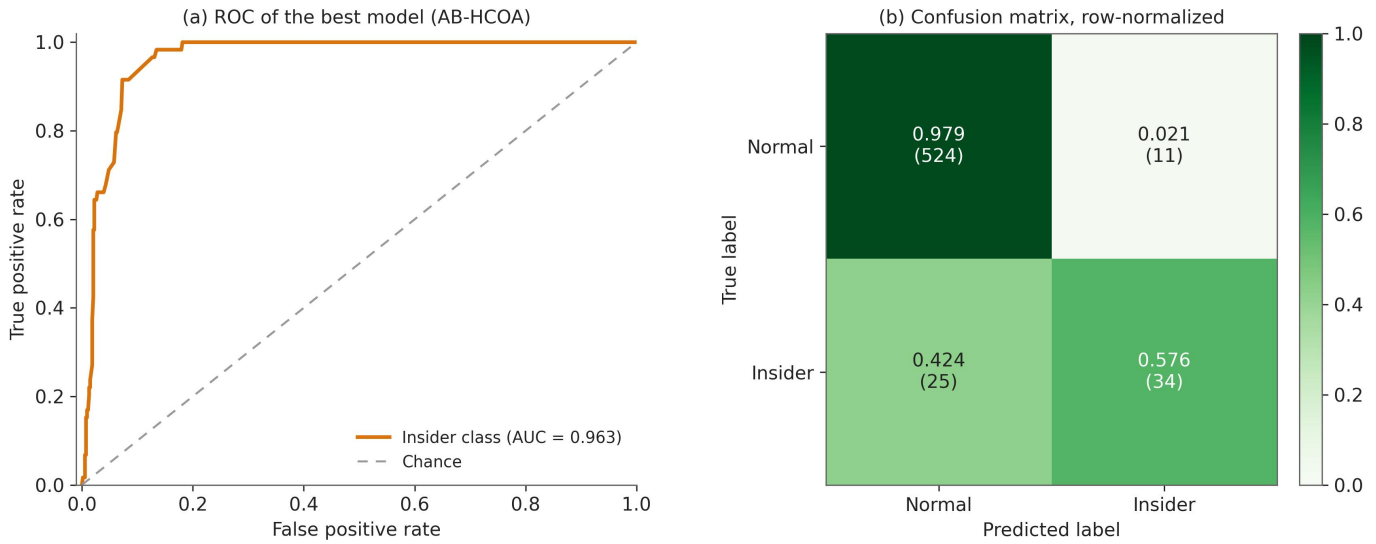

Figure 5. (a) Best performing model ROC; (b) confusion matrix.

Table 9. Omnibus statistical comparison of the optimization methods.

Statistical test	Methods	Runs	Test statistic	p-value	Effect size	Conclusion
Friedman test	9	30	$\chi^2(8) = 68.5581$	< 0.000001	Kendall's $W = 0.2857$	Significant

Table 10. Pairwise Wilcoxon signed-rank comparisons with Holm correction. Every comparison is against AB-COLSHADE, the method with the highest mean objective (0.587096).

Comparator	Comparator mean	Wilcoxon W	Raw p	Holm-adj. p	Rank-biserial r	Decision
AB-SCA	0.575799	37.0	0.00001	0.00008	+0.841	Significant
AB-RSA	0.573621	42.0	0.00002	0.00015	+0.819	Significant
AB-COA	0.571978	51.0	0.00006	0.00038	+0.781	Significant
AB-HCOA	0.577024	54.0	0.00009	0.00044	+0.768	Significant
AB-WOA	0.580355	113.0	0.01283	0.05133	+0.514	n.s.
AB-FA	0.585203	164.0	0.16418	0.49255	+0.295	n.s.
AB-PSO	0.586265	171.0	0.66541	1.00000	+0.095	n.s.
AB-GA	0.586208	198.0	0.67328	1.00000	+0.090	n.s.

relation reported as the effect size. Table 10 gives the complete results. AB-COLSHADE is significantly better than four of the eight comparators after correction, and is not distinguishable from AB-WOA, AB-FA, AB-PSO or AB-GA. The proposed hybrid falls in the significantly-worse group. These comparisons should be read alongside the magnitudes in Table 4: statistical detectability at 30 runs does not by itself imply that the difference is large enough to influence a practical choice of optimizer.

4.5. Feature attribution

A SHapley Additive exPlanations (SHAP) analysis [47] was conducted to determine the feature importance for the decision-making process of the best model. The outcomes are presented in Figure 6. The SHAP interpretation suggests that the time of the day and the day of the week play an important role in user activity being classified correctly. Additionally, the type of activity (logon or logoff) is considered. Whether an activity falls at a weekend, by contrast, contributes little to the classification, most probably because that information is already carried by the day-of-week feature. Activities occurring outside typical

working patterns significantly increase the probability of being identified as insider threats. This interpretability analysis enhances trust in the model and aligns with known behavioural characteristics of malicious insiders.

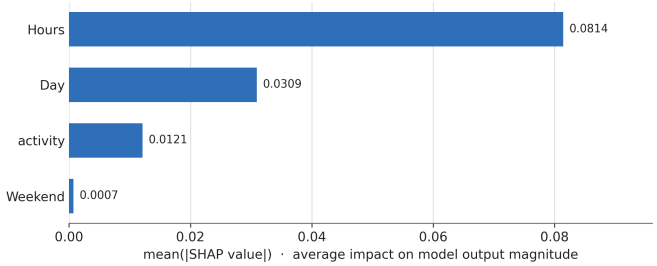

Figure 6. SHAP analysis outcomes for the best model.

4.6. Comparison with conventional classification methods

To establish whether the observed performance originates in the optimization procedure or simply in the suitability of AdaBoost for this feature set, the tuned model was compared against a range of conventional classifiers applied to the same problem. Every baseline was trained on the identical training partition, evaluated on the identical held-out test partition and scored with the same objective function, namely Cohen's kappa. All baselines were used with their default library settings, so that the comparison isolates the effect of hyperparameter selection rather than of implementation differences. The supervised baselines comprise AdaBoost with default parameters, a single decision tree, a random forest, gradient boosting, XGBoost, LightGBM, a support vector machine and logistic regression. Because insider-threat detection is frequently approached as an anomaly-detection problem, two unsupervised detectors were also included, an isolation forest and a one-class support vector machine, each fitted on the normal-class training data alone and evaluated by treating flagged outliers as insider predictions. Table 11 reports the outcome and Figure 7 presents it graphically.

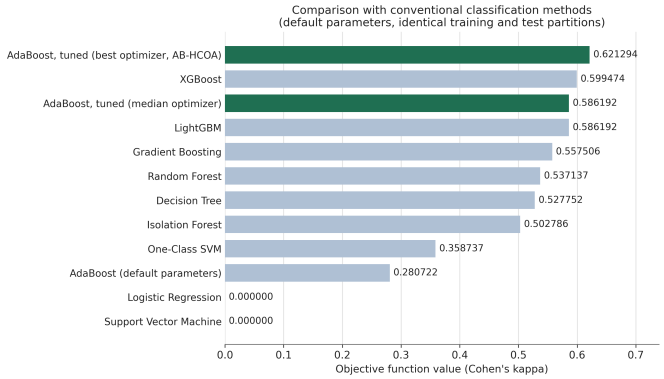

Figure 7. Objective function value attained by the conventional classification baselines and by the proposed approach.

Three observations follow. The first concerns the value of tuning. AdaBoost with default parameters attains a Cohen's kappa of 0.280722 and recovers only 14 of the 59 insider events, whereas the same learner tuned by the nine optimizers reaches between 0.468356 and 0.621294 on the held-out partition, with a median of 0.586192. Tuning therefore roughly doubles the achievable performance of this classifier, and it is the single largest effect observed anywhere in this study. The second observation qualifies the first. The tuned models land in the same band as modern gradient-boosted ensembles used straight out of the box: XGBoost attains 0.599474 and LightGBM 0.586192 with default settings, which places them above the median tuned AdaBoost and below the best of them. Tuning AdaBoost is thus worth a great deal relative to leaving AdaBoost untuned, but it does not by itself yield a model that clearly outperforms a well-chosen default ensemble, and an honest reading of Table 11 is that the choice of learner and the decision to tune matter more than the refinement of the tuning procedure. The third observation concerns the evaluation criterion: logistic regression and the support vector machine both assign every test observation to the majority class, producing a Cohen's kappa of exactly zero while still reporting an apparent accuracy of 90.07 %. That is a direct demonstration of why accuracy is an inadequate criterion on this dataset and why Cohen's kappa was adopted as the optimization objective.

Table 11. Comparison against conventional classification methods with default parameters, on the same training and test partitions.

Method	Cohen's kappa	Accuracy	Recall (insider)	Precision (insider)	MCC
AdaBoost, tuned (best optimizer, AB-HCOA)	0.621294	0.939394	0.576271	0.755556	0.628144
XGBoost	0.599474	0.932660	0.593220	0.686275	0.601407
AdaBoost, tuned (median optimizer)	0.586192	0.932660	0.559322	0.702128	0.590762
LightGBM	0.586192	0.932660	0.559322	0.702128	0.590762
Gradient Boosting	0.557506	0.930976	0.508475	0.714286	0.567133
Random Forest	0.537137	0.925926	0.508475	0.666667	0.543059
Decision Tree	0.527752	0.927609	0.474576	0.700000	0.539631
Isolation Forest	0.502786	0.892256	0.694915	0.471264	0.515121
One-Class SVM	0.358737	0.863636	0.525424	0.369048	0.365976
AdaBoost (default parameters)	0.280722	0.902357	0.237288	0.518519	0.305835
Logistic Regression	0.000000	0.900673	0.000000	0.000000	0.000000
Support Vector Machine	0.000000	0.900673	0.000000	0.000000	0.000000

Table 12. Ablation of the two mechanisms introduced into COA, over 30 independent runs under an identical search budget.

Configuration	Best	Worst	Mean	Median	Std
AB-HCOA (proposed)	0.610504	0.540693	0.577024	0.576585	0.017836
AB-COA with FA only	0.614439	0.527246	0.576788	0.578396	0.020956
AB-COA with QRL only	0.610504	0.534784	0.573964	0.571607	0.018763
AB-COA (original)	0.609480	0.528939	0.571978	0.572531	0.020823

4.7. Ablation of the introduced mechanisms

The hybrid adds two mechanisms to the original Crayfish Optimization Algorithm, and their individual contributions are separated by an ablation over the same 30 runs and the same budget. Four configurations were compared: the original algorithm, the algorithm with quasi-reflective learning alone, with the firefly-inspired step alone, and with both. Table 12 reports the outcome and Figure 8 the distributions.

The full hybrid attains the highest mean, and a Friedman test does detect a difference among the four configurations, returning $\chi^2(3) = 9.4839$ with $p = 0.0235$. The pairwise comparisons, however, locate that difference nowhere in particular: after Holm correction the full method is separated neither from quasi-reflective learning used alone (adjusted $p = 0.08342$, rank-biserial $+0.513$), nor from the unmodified algorithm ($p = 0.24827$, $+0.325$), nor from the firefly-inspired step used alone ($p = 0.73750$, -0.071). Taken together these results indicate that the firefly-inspired component accounts for what improvement there is, that quasi-reflective learning contributes little on this problem, and that combining the two is not demonstrably better than using the firefly component by itself: the effect size of that last comparison is in fact very slightly negative. This is reported as found; the design intention was that the two mechanisms would be complementary, and on this problem that expectation was not borne out.

4.8. Security-relevant performance and operational integration

Table 13 reports the raw confusion matrix, false positive and false negative rates, the Matthews correlation coefficient (MCC), ROC-AUC and insider-class PR-AUC for the best model produced by each optimizer, and Figure 9 compares the two threshold-independent measures. The figures are markedly less favourable than accuracy alone would suggest. For the model selected by AB-COLSHADE, 32 of the 59 insider events in the held-out partition are detected and 27 are missed, a false negative rate of 45.76 %, while 15 of the 535 normal events are incorrectly flagged, a false positive rate of 2.80 %. Across all nine methods the false negative rate lies between 38.98 % and 61.02 %. In other words, the best models available here miss between two fifths and three fifths of the insider events, at an accuracy of between 92.4 % and

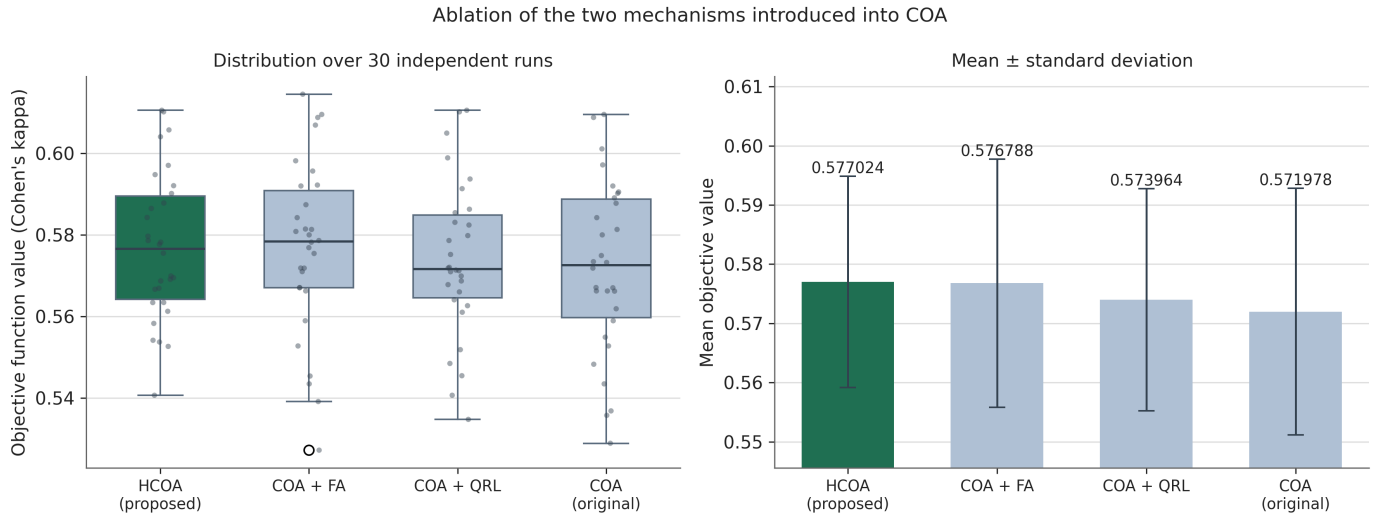

Figure 8. Ablation of the two introduced mechanisms. Left: distribution over 30 independent runs. Right: mean and standard deviation.

Table 13. Security-relevant performance measures on the held-out partition. The insider class is treated as positive.

Method	TP	FN	FP	TN	FPR	FNR	MCC	ROC-AUC	PR-AUC
AB-COLSHADE	32	27	15	520	0.0280	0.4576	0.5699	0.9565	0.6206
AB-PSO	32	27	14	521	0.0262	0.4576	0.5776	0.9629	0.6397
AB-GA	23	36	9	526	0.0168	0.6102	0.4942	0.9607	0.6332
AB-FA	33	26	14	521	0.0262	0.4407	0.5908	0.9625	0.6381
AB-WOA	34	25	14	521	0.0262	0.4237	0.6037	0.9630	0.6334
AB-HCOA	34	25	11	524	0.0206	0.4237	0.6281	0.9633	0.6305
AB-SCA	31	28	14	521	0.0262	0.4746	0.5643	0.9648	0.6585
AB-RSA	35	24	14	521	0.0262	0.4068	0.6165	0.9602	0.6152
AB-COA	36	23	15	520	0.0280	0.3898	0.6215	0.9610	0.6321

93.9 %.

The gap between ROC-AUC (0.9565) and insider-class PR-AUC (0.6206) makes the same point in a threshold-independent form: the ROC curve is dominated by the abundant negative class and looks uniformly strong across all nine methods, whereas the precision–recall curve, which conditions on the rare positive class, exposes how much harder that class is. A detector that never raised an alert would still report 90.07 % accuracy on this partition. Reporting accuracy or ROC-AUC alone would therefore substantially overstate the operational value of any of these models.

These figures set realistic expectations for deployment. A false negative rate of roughly one in two means the model cannot responsibly be operated as a standalone control, and even the lowest false positive rate observed here, 1.68 %, applied to an organizational logon stream of realistic volume would generate a substantial absolute number of alerts, because the true operating ratio of normal to insider activity is far more extreme than the 9:1 ratio used for training. The appropriate role for a model of this kind is as one contributing risk signal within a wider detection and access-control pipeline rather than as a decision procedure in its own right.

Several integration points follow directly from this. Within a security information and event management (SIEM) platform the per-event score can be emitted as a risk attribute and correlated with other indicators, so that an alert is raised on the conjunction of several weak signals rather than on this model alone; the SHAP attributions reported above can accompany each alert to give the analyst the reason for the score. Within a

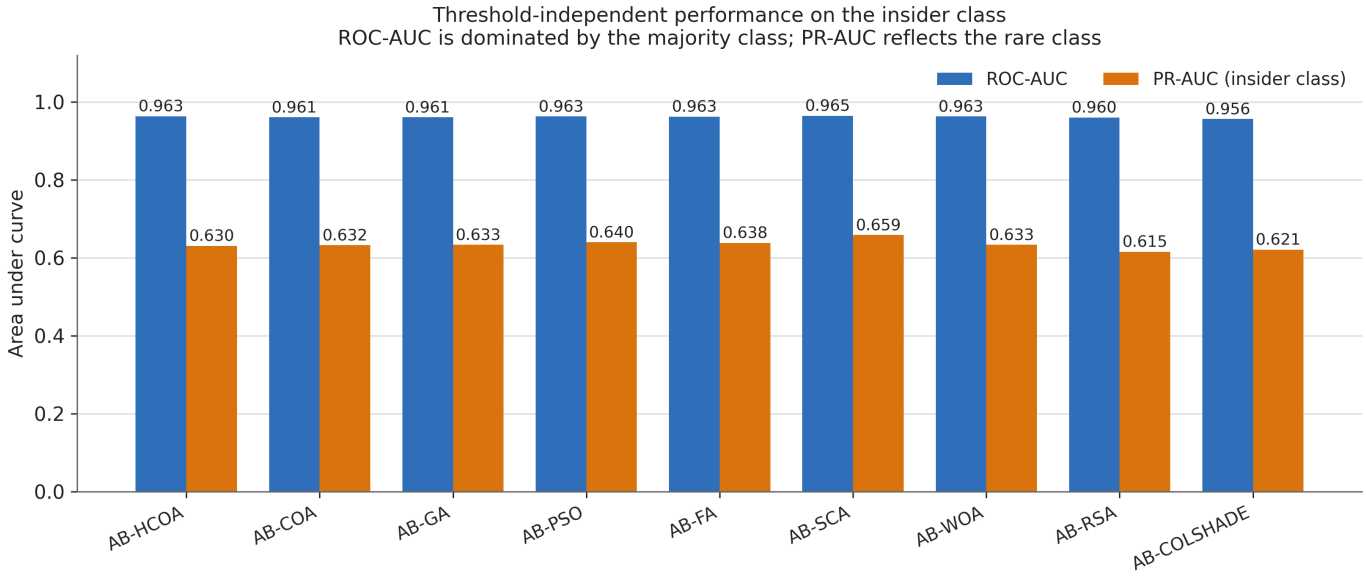

Figure 9. ROC-AUC and insider-class PR-AUC for the best model of each optimizer.

security orchestration, automation and response (SOAR) workflow the score is better suited to triggering low-cost, reversible actions, such as requiring re-authentication or opening an enrichment task, than irreversible ones such as account suspension. Within identity and access management (IAM) and network access control (NAC) the score can be consumed as an input to a step-up authentication policy. A behavioural score can only raise such a challenge, however, and cannot itself establish who is present, so the assurance the policy delivers is bounded by the strength of the authentication mechanism it invokes. Biometric modalities are the usual basis for that step, and the practical considerations are well covered in the literature: the design of fingerprint-based security information systems [48], the standardization of biometric security [49], the derivation of stable cryptographic keys from biometric templates [50], and verification procedures intended specifically to prevent identity substitution [51]. Within a zero trust network access (ZTNA) architecture, where every request is evaluated against current context rather than network location, a behavioural score of this kind is a natural additional term in the policy decision, and the blockchain-based attestation and tamper-evident logging architectures discussed in Section 2 provide the enforcement and audit layer through which such a decision can be applied and recorded. In each case the operating threshold should be set from the precision–recall curve according to the alert volume the receiving team can absorb, rather than from the default decision threshold of 0.5.

5. Conclusion

This study compared nine metaheuristic optimizers for tuning the control parameters of an AdaBoost classifier applied to insider-threat detection from CERT logon activity, under an identical evaluation budget and a selection protocol in which the held-out partition took no part in the search. Three findings emerge. First, hyperparameter tuning is decisive: AdaBoost with default settings attains a Cohen's kappa of 0.280722 and recovers 14 of 59 insider events, whereas tuned configurations attain approximately 0.59, and the strongest conventional alternative evaluated with default settings, XGBoost, reaches 0.599474. Second, the choice of optimizer matters considerably less than the decision to tune at all. Only 0.0151 of Cohen's kappa separates the strongest from the weakest of the nine methods, and although a Friedman test detects the differences at 30 runs, the concordance coefficient of 0.2857 indicates weak differentiation, and the ranking is

largely stable when the budget is varied by a factor of more than six. Third, the hybrid variant introduced here improves on the original Crayfish Optimization Algorithm but does not match the leading methods, and the ablation attributes what improvement there is to the firefly-inspired component rather than to quasi-reflective learning.

For practice, the implication is that effort is better spent on tuning and on the choice of evaluation criterion than on the selection of a particular metaheuristic. The security-relevant measures reinforce this. The best models available here miss roughly half of the insider events in the held-out partition while maintaining a false positive rate of a few per cent, which is a usable risk signal but not a standalone control, and the gap between ROC-AUC and insider-class PR-AUC shows that headline measures conceal that difficulty.

Several limitations should be taken into account when interpreting these results. The data are simulated rather than collected from a live organization, so behavioural patterns are generated by a scenario model and generalization to real operating environments has not been demonstrated. The majority class was down-sampled aggressively, from 854,661 to 1,782 observations, which makes the experimental distribution considerably denser in insider activity than any real deployment would be. Only logon and logoff activity and its timing were analysed, so the feature space is narrow by construction and excludes the file, email, device and web activity that the source dataset also records. The search budget was small, namely a population of 20 agents over 50 iterations with a restricted interval for the number of estimators, a choice forced by the cost of training a complete ensemble on three cross-validation folds at every objective-function evaluation. That budget also limits what the comparisons can establish: in the ablation of the two introduced mechanisms no pairwise difference survives correction for multiplicity, and although several of the pairwise comparisons between optimizers are statistically detectable after Holm correction, the magnitudes involved are small enough that they are unlikely to influence a practical choice of optimizer. The evaluation protocol keeps the held-out partition entirely outside the search, so the held-out figures are not selection-set results; they nevertheless describe a single stratified partition of a single dataset, and a repeated or nested resampling scheme would give a more stable estimate of generalization. Establishing that is the first item of future work.

Future work will address these limitations along several lines: validation on additional insider-threat datasets and, where access permits, on data captured from real organizational environments; extension of the feature set to the remaining categories of user activity recorded in the source data; comparison against non-metaheuristic tuning strategies such as grid search, random search and Bayesian optimization; a systematic analysis of computational cost across the compared methods; and a sensitivity analysis over the down-sampling ratio together with a comparison against oversampling and cost-sensitive alternatives. The modified optimizer may also prove useful for optimization tasks outside insider-threat detection.

Author Contributions

Snezana Anetic: conceptualization, methodology, investigation, writing — original draft. **Svetlana Andjelic:** methodology, validation, writing — review and editing. **Milos Grubjesic:** software, data curation, visualization. **Mahmoud Abdel-Salam:** formal analysis, software, validation. **Vladimir Simic:** formal analysis, writing — review and editing. **Miodrag Zivkovic:** methodology, software, supervision. **Nebojsa Bacanin:** conceptualization, supervision, project administration, writing — review and editing. **Samson Omasirichukwu Offorjindu:** investigation, validation, writing — review and editing. **Milos Antonijevic:** resources, supervision, writing — review and editing. **Komlen Lalovic:** supervision, writing — review and editing. All authors have read and agreed to the submitted version of the manuscript.

Competing Interests Statement

Nebojsa Bacanin is a member of the Editorial Team of *Science, Engineering and Technology*. He was not involved in the peer-review process or in any editorial

decision regarding this manuscript. The other authors declare no competing interests.

Data Availability Statement

The source dataset analysed in this study is publicly available. It is the Insider Threat Test Dataset, release r4.2, published by the CERT Division of the Carnegie Mellon University Software Engineering Institute and accessible at `https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247` [46]. The preprocessed logon-activity subset derived from it, the exact training and testing partitions, the implementation of all nine optimizers and the analysis scripts that produce every table and figure reported here have been deposited in the Zenodo repository and are available at `https://zenodo.org/records/21959722`.

Code Availability Statement

The complete implementation is openly available in the deposit referenced in the data availability statement. It comprises the nine optimizers compared here, including the proposed hybrid; a single driver script that reproduces the main experiment, the reduced-budget experiment and the ablation; and the scripts that regenerate every table and every figure in Section 4 from that output. Running the driver script on the deposited partitions reproduces the reported results exactly.

Preprint Statement

An earlier version of this work was posted as a preprint under a different title and with a different author list. That preprint was subsequently withdrawn from review, and the present manuscript is a substantially revised version of it. The preprint record is permanently available at `https://www.preprints.org/manuscript/202409.1500`.

Statement on the Ethical Use of AI Tools

Artificial intelligence tools were used during the preparation of this manuscript solely for language correction and readability improvement of text written by the authors. No AI tool was used to generate research data, to design or execute the experiments, to carry out the statistical analysis, or to formulate the scientific claims and conclusions reported here. All authors reviewed and verified the final text and accept full responsibility for its content.

Funding Statement

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Acknowledgements

The authors thank the CERT Division of the Carnegie Mellon University Software Engineering Institute for making the Insider Threat Test Dataset publicly available, and the anonymous reviewers and the Handling Editor for comments that materially improved this article.

References

[1] B. B. Gupta and S. Srinivasagopalan, *Handbook of Research on Intrusion Detection Systems*. Hershey, PA, USA: IGI Global, 2020.

[2] T. N. Nisha and D. Pramod, “Insider intrusion detection techniques: A state-of-the-art review,” *Journal of Computer Information Systems*, vol. 64, no. 1, pp. 106–123, 2024.

[3] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *Journal of Information Security and Applications*, vol. 50, p. 102 419, 2020. DOI: 10.1016/j.jisa.2019.102419

[4] N. Bacanin, M. Zivkovic, L. Jovanovic, M. Ivanovic, and T. A. Rashid, “Training a multilayer perception for modeling stock price index predictions using modified whale optimization algorithm,” in *Computational Vision and Bio-Inspired Computing: Proc. ICCVBIC 2021*,

ser. Advances in Intelligent Systems and Computing, vol. 1420, Singapore: Springer, 2022, pp. 415–430.

[5] P. Dakic et al., "Intrusion detection using metaheuristic optimization within IoT/IIoT systems and software of autonomous vehicles," *Scientific Reports*, vol. 14, no. 1, p. 22 884, 2024.

[6] D. Mladenovic et al., "Sentiment classification for insider threat identification using metaheuristic optimized machine learning classifiers," *Scientific Reports*, vol. 14, no. 1, p. 25 731, 2024.

[7] A. Minic et al., "Applying recurrent neural networks for anomaly detection in electrocardiogram sensor data," *Sensors*, vol. 23, no. 24, p. 9878, 2023. DOI: 10.3390/s23249878

[8] M. Zivkovic, L. Jovanovic, M. Ivanovic, A. Krdzic, N. Bacanin, and I. Strumberger, "Feature selection using modified sine cosine algorithm with COVID-19 dataset," in *Evolutionary Computing and Mobile Sustainable Networks: Proc. ICECMSN 2021*, ser. Lecture Notes on Data Engineering and Communications Technologies, vol. 116, Singapore: Springer, 2022, pp. 15–31.

[9] R. Damaševičius et al., "Decomposition aided attention-based recurrent neural networks for multistep ahead time-series forecasting of renewable power generation," *PeerJ Computer Science*, vol. 10, e1795, 2024.

[10] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997. DOI: 10.1109/4235.585893

[11] J. Kosseff, "Defining cybersecurity law," *Iowa Law Review*, vol. 103, no. 3, pp. 985–1031, 2018.

[12] A. Moallem, Ed., *HCI for Cybersecurity, Privacy and Trust*, vol. 14728, Lecture Notes in Computer Science, Cham, Switzerland: Springer, 2024.

[13] D. J. B. Svantesson and D. Kloza, *Trans-Atlantic Data Privacy Relations as a Challenge for Democracy* (European Integration and Democracy Series). Cambridge, U.K.: Intersentia, 2017, vol. 4.

[14] V. Papakonstantinou, "Cybersecurity as praxis and as a state: The EU law path towards acknowledgement of a new right to cybersecurity?" *Computer Law & Security Review*, vol. 44, p. 105 653, 2022. DOI: 10.1016/j.clsr.2022.105653

[15] M. Lukings and A. H. Lashkari, *Understanding Cybersecurity Law and Digital Privacy: A Common Law Perspective*. Cham, Switzerland: Springer, 2022.

[16] J. Pohl, "International data transfers and cybersecurity: Three regulatory approaches and their implications," in *Public and Private Governance of Cybersecurity: Challenges and Potential*, Cambridge, U.K.: Cambridge Univ. Press, 2023, pp. 134–160.

[17] Australian Government, Department of Home Affairs. "2023–2030 Australian cyber security strategy," Accessed: Aug. 18, 2026. [Online]. Available: https://www.homeaffairs.gov.au/about-us/our-portfolios/cyber-security/strategy/2023-2030-australian-cyber-security-strategy

[18] A. Savin and J. Trzaskowski, Eds., *Research Handbook on EU Internet Law*, 2nd ed. Cheltenham, U.K.: Edward Elgar Publishing, 2023.

[19] J. Ullrich, J. Cropper, P. Frühwirt, and E. Weippl, "The role and security of firewalls in cyber-physical cloud computing," *EURASIP Journal on Information Security*, vol. 2016, p. 18, 2016.

[20] S. Thapa and A. Mailewa, "The role of intrusion detection/prevention systems in modern computer networks: A review," in *Proc. 53rd Midwest Instruction and Computing Symp. (MICS)*, Milwaukee, WI, USA, 2020, pp. 1–14.

[21] L. Bilge and T. Dumitraș, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. 2012 ACM Conf. Computer and Communications Security (CCS)*, Raleigh, NC, USA, 2012, pp. 833–844. DOI: 10.1145/2382196.2382284

[22] E. Džaferović, A. Sokol, A. A. Almisreb, and S. M. Norzeli, "DoS and DDoS vulnerability of IoT: A review," *Sustainable Engineering and Innovation*, vol. 1, no. 1, pp. 43–48, 2019.

[23] M. Kandias, A. Mylonas, N. Virvilis, M. Theoharidou, and D. Gritzalis, "An insider threat prediction model," in *Proc. 7th Int. Conf. Trust, Privacy and Security in Digital Business (TrustBus)*, ser. Lecture Notes in Computer Science, vol. 6264, Bilbao, Spain, 2010, pp. 26–37.

[24] C. Korkuc, Y. Genc, T. Dogantuna, E. Afacan, and M. Dener, "BBNAC: Blockchain based network access control for IT/OT infrastructures," *Journal of Network and Systems Management*, vol. 34, no. 2, p. 60, 2026. DOI: 10.1007/s10922-026-10033-w

[25] S. Yuan and X. Wu, "Deep learning for insider threat detection: Review, challenges and opportunities," *Computers & Security*, vol. 104, p. 102 221, 2021. DOI: 10.1016/j.cose.2021.102221

[26] D. C. Le and N. Zincir-Heywood, "Anomaly detection for insider threats using unsupervised ensembles," *IEEE Transactions on Network and Service Management*,

vol. 18, no. 2, pp. 1152–1164, 2021. DOI: 10.1109/TNSM.2021.3071928

[27] R. Nasir, M. Afzal, R. Latif, and W. Iqbal, "Behavioral based insider threat detection using deep learning," *IEEE Access*, vol. 9, pp. 143 266–143 274, 2021. DOI: 10.1109/ACCESS.2021.3118297

[28] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, "Explainable artificial intelligence in cybersecurity: A survey," *IEEE Access*, vol. 10, pp. 93 575–93 600, 2022. DOI: 10.1109/ACCESS.2022.3204171

[29] C. Korkuc, T. Dogantuna, E. Afacan, and E. Bostanci, "Blockchain based network access control (NAC) management solution and architecture," in *Proc. 29th Signal Processing and Communications Applications Conf. (SIU)*, Istanbul, Turkey, 2021, pp. 1–4. DOI: 10.1109/SIU53274.2021.9477893

[30] C. Korkuc, T. Dogantuna, Y. Genc, E. Afacan, and M. Dener, "A blockchain attestation layer for zero trust network access," in *Proc. 8th Int. Congr. Human-Computer Interaction, Optimization and Robotic Applications (ICHORA)*, 2026, pp. 1–6.

[31] C. Korkuc, N. Aytas Korkmaz, Y. Genc, A. Akkoc, E. Afacan, and E. Yazgan, "BLOCKBOX: Blockchain based black box designing and modeling," *Concurrency and Computation: Practice and Experience*, vol. 36, no. 13, e8057, 2024. DOI: 10.1002/cpe.8057

[32] H. Jia, H. Rao, C. Wen, and S. Mirjalili, "Crayfish optimization algorithm," *Artificial Intelligence Review*, vol. 56, no. Suppl. 2, pp. 1919–1979, 2023. DOI: 10.1007/s10462-023-10567-4

[33] R. Wang, S. Zhang, and G. Zou, "An improved multi-strategy crayfish optimization algorithm for solving numerical optimization problems," *Biomimetics*, vol. 9, no. 6, p. 361, 2024. DOI: 10.3390/biomimetics9060361

[34] L. Chaib, M. Tadj, A. Choucha, F. Z. Khemili, and A. El-Fergany, "Improved crayfish optimization algorithm for parameters estimation of photovoltaic models," *Energy Conversion and Management*, vol. 313, p. 118 627, 2024. DOI: 10.1016/j.enconman.2024.118627

[35] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009. DOI: 10.4310/SII.2009.v2.n3.a8

[36] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks: Theory and Applications*, ser. Studies in Computational Intelligence, vol. 780, Cham, Switzerland: Springer, 2019, pp. 43–55.

[37] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks (ICNN)*, vol. 4, Perth, WA, Australia, 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968

[38] X.-S. Yang and X. He, "Firefly algorithm: Recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36–50, 2013. DOI: 10.1504/IJSI.2013.055801

[39] S. Mirjalili, "SCA: A sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016. DOI: 10.1016/j.knosys.2015.12.022

[40] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016. DOI: 10.1016/j.advengsoft.2016.01.008

[41] L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer," *Expert Systems with Applications*, vol. 191, p. 116 158, 2022. DOI: 10.1016/j.eswa.2021.116158

[42] J. Gurrola-Ramos, A. Hernàndez-Aguirre, and O. Dalmau-Cedeño, "COLSHADE for real-world single-objective constrained optimization problems," in *Proc. IEEE Congr. Evolutionary Computation (CEC)*, Glasgow, U.K., 2020, pp. 1–8.

[43] S. Mirjalili and A. H. Gandomi, Eds., *Comprehensive Metaheuristics: Algorithms and Applications*. Amsterdam, The Netherlands: Elsevier, 2023.

[44] W. Luo and X. Yu, "Quasi-reflection based multi-strategy cuckoo search for parameter estimation of photovoltaic solar modules," *Solar Energy*, vol. 243, pp. 264–278, 2022.

[45] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proc. IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2013, pp. 98–104. DOI: 10.1109/SPW.2013.37

[46] Carnegie Mellon University, Software Engineering Institute. "Insider Threat Test Dataset," Accessed: Jan. 25, 2023. [Online]. Available: https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247

[47] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, Long Beach, CA, USA, 2017, pp. 4765–4774.

[48] K. Lalović, I. Tot, A. Arsić, and M. Škarić, "Security information system, based on fingerprint biometrics," *Acta Polytechnica Hungarica*, vol. 16, no. 5, pp. 87–100, 2019. DOI: 10.12700/APH.16.5.2019.5.6

[49] M. Trikoš, I. Tot, J. Bajčetić, K. Lalović, B. Jovanović, and D. Bogićević, "Biometric security standardization," in *Proc. Zooming Innovation in Consumer Technologies Conf. (ZINC)*, Novi Sad, Serbia, 2019, pp. 17–20. DOI: 10.1109/ZINC.2019.8769419

[50] N. Maček, B. Đorđević, J. Gavrilović, and K. Lalović, "An approach to robust biometric key generation system design," *Acta Polytechnica Hungarica*, vol. 12, no. 8, pp. 43–60, 2015. DOI: 10.12700/APH.12.8.2015.8.3

[51] K. Lalović et al., "Biometric verification of maternity and identity switch prevention in maternity wards," *Acta Polytechnica Hungarica*, vol. 13, no. 5, pp. 65–81, 2016. DOI: 10.12700/APH.13.5.2016.5.4