Supplementary Material

Step-by-Step Procedure, Pseudo-code, and Full Parameter Settings for Hybrid Swarm Optimization of Three ML Classifiers (FSDM4784)

Prepared by: **Narongdech Dungkratoke et al.**
Generated on: September 18, 2025.

Introduction

This supplementary file provides the complete, reproducible description of the optimization framework referenced in the paper.
It includes (1) a step-by-step procedure, (2) algorithmic pseudo-code for FA, GWO, and Hybrid FA–GWO, (3) full hyperparameter search spaces for SVM, Random Forest, and KNN, and (4) the evaluation protocol.
The objective being minimized during metaheuristic search is $1 - F1$ on a 70/30 stratified train/test split.

Reproducibility Notes

- Environment: Python ≥ 3.9; NumPy ≥ 1.23; scikit-learn ≥ 1.2.
- Determinism: Random seeds (random_state=42) are fixed where supported. Metaheuristics remain stochastic; convergence curves and final scores are reported.
- Scaling: Standardization is applied to features used by SVM and KNN; Random Forest uses raw features.
- Objective during search: Minimize $1 - F1$.
- Final evaluation: Re-train model on training split and evaluate on hold-out test split.

Step-by-Step Procedure

1. Data Preparation
   1.1 Split the dataset into train/test with stratification (test_size=0.30, random_state=42).
   1.2 Fit a StandardScaler on X_train and transform X_train/X_test for SVM and KNN.
2. Define Models & Bounds
   - SVM (RBF): optimize $C \in [0.1, 100]$, gamma $\in [1e\text{-}4, 1]$.

    - Random Forest: optimize n_estimators ∈ [50, 200], max_depth ∈ [2, 15].
    - KNN: optimize n_neighbors ∈ [2, 20].
3. Objective Function
   3.1 For candidate vector θ, map to model hyperparameters (cast to int where required).
   3.2 Train model on X_train; predict on X_test; compute F1.
   3.3 Return 1 − F1 to be minimized.
4. Initialize Population
   Draw *n_agents* × *dim* initial positions uniformly within bounds.
5. Run Optimizer
   Use FA, GWO, or Hybrid FA–GWO for *max_iter* iterations, recording the best score each iteration.
6. Select Best Hyperparameters
   Take the position with the minimum objective value as best parameters.
7. Final Evaluation & Reporting
   Refit the model on X_train with best parameters.
   Evaluate on X_test: Accuracy, Precision, Recall, F1-score, and ROC-AUC.
   Save best parameters and metrics for each *(Model × Optimizer)* pair.

Pseudo-code: Firefly Algorithm (FA) with Initialization

```
Input: objective f(·), bounds [L,U], init positions X (n×d),
      alpha, beta0, gamma, max_iter
Output: best position x*, best score f(x*), convergence curve
X ← init positions within [L,U]; I[i] ← f(X[i]) for all i
for t = 1..max_iter do
    for i = 1..n do
        for j = 1..n do
            if I[j] < I[i] then
                r ← ||X[i] − X[j]||
                β ← beta0 * exp(−gamma * r^2)
                step ← β (X[j] − X[i]) + alpha * (rand(d) − 0.5)
                X[i] ← clip(X[i] + step, L, U)
                I[i] ← f(X[i])
    record min(I) to curve
return argmin(I), min(I), curve
```

Pseudo-code: Grey Wolf Optimizer (GWO) with Initialization

```
Input: objective f(·), bounds [L,U], init positions X (n×d), max_iter
Output: best position α, best score f(α), convergence curve
Initialize α, β, δ as +∞; α_pos, β_pos, δ_pos as None
for t = 1..max_iter do
    for each wolf i:
        s ← f(X[i]); update α, β, δ and their positions
    a ← 2 − 2 * t / max_iter
    for each wolf i do
        for each dimension d do
            draw r1,r2; A ← 2*a*r1 − a; C ← 2*r2
            X1 ← α_pos[d] − A*|C*α_pos[d] − X[i,d]|
            repeat with β, δ → X2, X3
            X[i,d] ← clip((X1 + X2 + X3)/3, L[d], U[d])
    record α to curve
return α_pos, α, curve
```

Pseudo-code: Hybrid FA–GWO

```
Input: objective f(·), bounds [L,U], init positions X (n×d),
      alpha, beta0, gamma, max_iter
Output: best position α, best score f(α), convergence curve
Track α, β, δ wolves as in GWO
for t = 1..max_iter do
    Update α, β, δ by evaluating all wolves
    a ← 2 − 2 * t / max_iter
    for each agent i do
        compute GWO-guided position G[i] (average of X1,X2,X3)
        r ← ||X[i] − G[i]||; β ← beta0 * exp(−gamma * r^2)
        step ← β (G[i] − X[i]) + alpha * (rand(d) − 0.5)
        X[i] ← clip(X[i] + step, L, U)
    record α to curve
return α_pos, α, curve
```

Hyperparameter Search Spaces

Model	Hyperparameters to Optimize	Bounds / Domain	Type
SVM (RBF)	C, gamma	[0.1, 100], [1e-4, 1]	float,float
RandomForest	n_estimators, max_depth	[50, 200], [2, 15]	int,int
KNN	n_neighbors	[2, 20]	int

Evaluation Protocol

- Split: stratified train/test, test_size = 0.30, random_state = 42.
- Objective minimized during search: 1 − F1 on hold-out test split.
- After selecting best hyperparameters, refit on training set and evaluate on test set:
  Accuracy, Precision, Recall, F1-score, and ROC-AUC.
- Convergence curves record best score per iteration.

How to Run (Example Skeleton)

- Prepare X_train_scaled, X_test_scaled, y_train, y_test as described above.
- Set n_agents = 10, max_iter = 20 (modifiable).
- For each model in {SVM, RF, KNN}:
  - Define objective(params) → returns 1 − F1.
  - Initialize positions uniformly within bounds.
  - Call one of {firefly_with_init, gwo_with_init, hybrid_fa_gwo_init}.
  - Train final model with best parameters; compute metrics and ROC.