Supplementary File 1

An efficient hybrid filter-wrapper method based on improved Harris Hawks optimization for feature selection

Jamshid Pirgazi*, Mohammad Mehdi Pourhashem Kallehbasti, Ali Ghanbari Sorkhi, Ali Kermani

Department of Electrical and Computer Engineering, University of Science and Technology of Mazandaran, Behshahr, Iran

Algorithm S1. HHO algorithm

```
Input:  N (population size),
        T (maximum number of iteration)
Output: The best position of prey
Randomly generating N hawks $X_i$ (i=1, 2, …, N)
t = 1
while $t \leq T$ do
    Calculate the fitness values of hawks
    Set $X_{prey}$ as the best position of the prey
    for each hawk ($X_i$) do
        Update the initial energy $E_0$ and jump strength J
        Update the energy (E) using Eq. 2.
        if $|E| \geq 1$ then
            Update the position vector using Eq. 1.
        end if
        if $|E| < 1$ then
            if $r \geq 0.5$ and $|E| \geq 0.5$ then
                Update the position vector using Eq. 4.
            else if $r \geq 0.5$ and $|E| < 0.5$ then
                Update the position vector using Eq. 6.
            else if $r < 0.5$ and $|E| \geq 0.5$ then
                Update the position vector using Eq. 10.
            else if $r < 0.5$ and $|E| < 0.5$ then
                Update the position vector using Eq. 11.
            end if
        end if
    end for
    t = t + 1
end while
Return $X_{prey}$
```

Algorithm S2. Pseudo code of the proposed algorithm

```
Input: D: Dataset; FS: filter measure; RF: classifier;
Output: S: The selected features
// FILTERING STAGE
For i = 1 to N // N = the count of features in D
    Score[i] = FS($D_i$) // The Score is calculated based on F-Score method
For i = 1 to N
    ProbSelect[i] = Score[i] / $\sum_{j=1}^{N}$ Score(j)
// WRAPPER STAGE - HHO Initialization
Nvar = 40, Npop = 10
For i = 1 to Npop
    Hawk[i].Position = Select a feature from D without replacement by using ProbSel[];
    Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
    Hawk[i].Fitness = Accuracy(C, T, [i].Position, D, RF);
// HHO Main Loop
While (the stopping condition is not met) do
    Hawks = Sort(Hawk.Fitness) // Sort Ascending by fitness of Hawk.
    Rabbit.Position = Hawks[1].position // The position of best Hawk
    For i = 1 to Npop do
        E0[i] = 2rand() - 1 // Update the initial energy E0
        J[i] = 2(1 - rand()) // Update the initial jump strength J
        E[i] = 2 E0[i](1 - $\frac{t}{T}$) // Update the E
        if (|E[i]| ≥ 1) then // Exploration phase
            if q ≥ 0.5
                Hawk[i].Position = RandomHawk.Position - r1 | RandomHawk.Position - 2r2 Hawk[i].Position |
                Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
            if q < 0.5
                Hawk[i].Position = (Rabbit.Position - AvrageHawks.position) - r3(LB + r4(UB - LB))
                Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
        if (|E[i]| < 1) then // Exploitation phase
            if (r ≥ 0.5 and |E[i]| ≥ 0.5) then // Soft besiege
                Hawk[i].Position = ΔHawk[i].Position - E | J Rabbit.Position - Hawk[i].Position |
                Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
            else if (r ≥ 0.5 and |E[i]| < 0.5) then // Hard besiege
                Hawk[i].Position = Rabbit.Position - E | ΔHawk[i].Position |
                Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
            else if (r < 0.5 and |E[i]| ≥ 0.5) then // Soft besiege with progressive rapid dives
                if F(Y) < F(X(t)) then
                    Hawk[i].Position = Y.position
                    Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
                if F(Z) < F(X(t)) then
                    Hawk[i].Position = Z.position
                    Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
            else if (r < 0.5 and |E[i]| < 0.5) then // Hard besiege with progressive rapid dives
                if F(Y) < F(X(t)) then
                    Hawk[i].Position = Y.position
                    Hawk[i].Position = Grasp(Hawk[i].Position, D, RF);
                if F(Z) < F(X(t)) then
                    Hawk[i].Position = Z.position
                    Hawk[i].Position=Grasp(Hawk [i].Position, D, RF);
    Pr1, Pr2 = Select_best(Hawks) // select 2 Hawks with the best fitness
    Child1, Child2 = crossover_mutation(Pr1, Pr2)
    substitute 2 children instead of 2 samples with the worst fitness
S = Hawks[1].position // final subset
```