The main code of IJAYA algorithm:

1. Add the path of object function file, “evaluate_normal_fitness.m”, and defined the array.

```
%%%%%%%%%% add the path %%%%%%%%%%%%%%%%%%%%%%%%
    addpath('Benchmark_Solar_Cell');
    fun = @evaluate_normal_fitness;

    FE_jilu=cell(1,3);
    x_jilu=cell(30,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2. Set the PV boundary value of X used the “PV_Xrange.m” file, run times and corresponding matrix.

```
%%%%%%%%%%%%%%%% set the parameter%%%%%%%%%%%%%%%%
    PV_Xrange;
    Number = 1;
    runNumber =30 ;
    FE_best=[];
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3. Initialize the population, calculate the fitness of object function and start the iteration cycle.

```
%%%%%%%%%%%%%%initialize the population%%%%%%%%%%%%%%%
    rand('seed', sum(100 * clock));
    tic
    fitcount=0;
    popsize = 30;
    X = repmat(Xmin, popsize, 1) + rand(popsize, D) .* (repmat(Xmax-Xmin,
        popsize, 1));
      for i=1:popsize
          val_X(i,:) = fun(X(i,:),func_flag);
          fitcount=fitcount+1;
          FE_best(Number,fitcount) =    val_X(i,:);
      end

    FES = popsize;
    maxFES = 50000;
    M_X=rand;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

4. Best and worst fitness value are obtained by ranking fitness value, judging and selecting update strategy.

```
%%%%%%%%%%%%%%ranking and judging%%%%%%%%%%%%%%%%%%
    while     FES < maxFES
              [~,r2] = sort(val_X);
              Best = X(r2(1),:);
              Worst= X(r2(end),:);
              if val_X(r2(end),:)==0
                    ww=1;

%%%%%%%%%%%%%%%%self-adaptive weight update%%%%%%%%%%%%%
              else
                    ww=abs(val_X(r2(1),:)/(val_X(r2(end),:)))^2;
              end

                 for i=1:popsize
                      if i~=r2(1)
                        if rand<rand
                              for j=1:D
                              Xi(j) = X(i,j) + rand*(Best(j) -abs(X(i,j)))-
                                ww*rand*(Worst(j) -abs(X(i,j)));
                          End

%%%%%%%%%%%experience-based learning update%%%%%%%%%%%%%%%%
                    else
                        nouse1(1)= randi(popsize);
                        while nouse1(1)==i
                        nouse1(1)= randi(popsize);
                        end
                        nouse1(2)= randi(popsize);
                        while nouse1(2)==i || nouse1(2)==nouse1(1)
                        nouse1(2)= randi(popsize);
                        end

                        if val_X(nouse1(1),:)<val_X(nouse1(2),:)
                             Xi = X(i,:) + rand(1,D).*(X(nouse1(1),:) -X(nouse1(2),:));
                        else
                             Xi = X(i,:) - rand(1,D).*(X(nouse1(1),:) -X(nouse1(2),:));
                        end

                    end

%%%%%%%%%%%%%%%chaotic learning method%%%%%%%%%%%%%%%%%
              else

                    M_X=4*M_X*(1-M_X);
                    for k=1:D
                          Xi(k)=Best(k)+(2*M_X-1)*rand;
```

```
                end
            end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5. Constraints on boundary values and calculating fitness value

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                Xi = boundConstraint_absorb(Xi, Xmin, Xmax);
                val_Xi = fun(Xi,func_flag);
                FES = FES+1;
               if val_Xi<val_X(i,:)
                   val_X(i,:) = val_Xi;
                   X(i,:) = Xi;
               end
           fitcount=fitcount+1;
           FE_best(Number,fitcount) =    min(val_X);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.Determine the optimal value and save it.

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [val_Best,index] = min(val_X);
        Best = X(index(1),:);
        jilu_besty(Number,func_flag)=val_Best;
        x_jilu{Number,func_flag}=Best;
        Number = Number + 1;
        FE_jilu{1,func_flag}=FE_best;
          save    jaya88_np30 jilu_besty x_jilu    FE_jilu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The value of jilu_besty is the best value to record the best RMSE for different run times (Number) and different problems (func_flag).

Paper: Yu K , Liang J J , Qu B Y , et al. Multiple learning backtracking search algorithm for estimating parameters of photovoltaic models[J]. Applied Energy, 2018, 226.

Yu K, Liang J J, Qu B Y, et al. Parameters identification of photovoltaic models using an improved JAYA optimization algorithm[J]. Energy Conversion and Management, 2017, 150: 742-753.