Data: Set the control parameters of the ABC algorithm
SN: Number of Foods
limit: Maximum number of trial for abandoning a source
MFE: Maximum number of fitness evaluations
begin

//Initialization:
num_eval ← 0;
for s = 1 to SN do
    X(s) ← random solution by Eq. 1 [3];
    f_s ← f(X(s));
    trial(s) ← 0;
    num_eval + +;
end

//Employed Bees Phase:
for s = 1 to SN do
    x' ← a new solution produced by Eq. 2 [3];
    f(x') ← evaluate new solution;
    num_eval + +;
    if f(x') < f_s then
        X(s) ← x'; f_s ← f(x'); trial(s) ← 0;
    else
        trial(s) ← trial(s) + 1;
    end
end
if num_eval == MFE then
    Memorize the best solution achieved so far and exit main repeat;
end

Calculate the probability values p_t for the solutions using fitness values by Eqs. 3 and 4 [3]:

//Onlooker Bee phase:
s ← 1; t ← 1;
repeat
    r ← rand(0, 1);
    if r < p(s) then
        t ← t + 1;
        x' ← a new solution produced by Eq. 2 [3];
        f(x') ← evaluate new solution;
        num_eval + +;
        if f(x') < f_s then
            X(s) ← x'; f_s ← f(x'); trial(s) ← 0;
        else
            trial(s) ← trial(s) + 1;
        end
    end
    if num_eval == MFE then
        Memorize the best solution achieved so far and exit main repeat;
    end
end
s ← (s mod SN) + 1;

//Scout Bee Phase:
mi ← {s : trial(s) = max(trial)};
if trial(mi) >= limit then
    X(mi) ← random solution by Eq. 1 [3];
    f_mi ← f(X(mi));
    num_eval + +;
    trial(mi) ← 0;
    if num_eval == MFE then
        Memorize the best solution achieved so far and exit main repeat;
    end
end
Memorize the best solution achieved so far:
until num_eval = MFE;
end

Algorithm 1: The pseudo-code of \( ABC_{imp1}(FEs) \)
Data: Set the control parameters of the ABC algorithm
SN: Number of Foods
limit: Maximum number of trial for abandoning a source
MCN: Maximum number of cycles

begin

// Initialization:
num_eval ← 0;
for s = 1 to SN do
  X(s) ← random solution by Eq. 1 [3];
  f_s ← f(X(s));
  trial(s) ← 0;
  num_eval + + ;
end

cycle ← 1;
while cycle < MCN do

  // Employed Bees Phase:
  mi ← { s : trial(s) = max(trial) };
  for s = 1 to SN do
    if (trial(s) < limit or s = mi) then
      x' ← a new solution produced by Eq. 2 [3];
      f(x') ← evaluate new solution;
      num_eval + + ;
      if f(x') < f_s then
        X(s) ← x'; f_s ← f(x'); trial(s) ← 0;
      else
        trial(s) ← trial(s) + 1;
    end
  end

Memorize the best solution achieved so far:

// Scout Bee Phase:
  if (trial(mi) ≥ limit) then
    X(mi) ← random solution by Eq. 1 [3];
    f_mi ← f(X(mi));
    num_eval + + ;
    trial(mi) ← 0;
  end

Calculate the probability values p_t for the solutions using fitness values by Eqs. 3 and 4 [3]:

// Unlooker Bees Phase:
  s ← 1; t ← 1;
  while t ≤ SN do
    r ← rand(0, 1);
    if r < p(s) then
      t ← t + 1;
      x' ← a new solution produced by Eq. 2 [3];
      f(x') ← evaluate new solution;
      num_eval + + ;
      if f(x') < f_s then
        X(s) ← x'; f_s ← f(x'); trial(s) ← 0;
      else
        trial(s) ← trial(s) + 1;
    end
  end
  s ← (s mod SN) + 1;
end
Memorize the best solution achieved so far:
cycle + + ;
end

Algorithm 2: The pseudo-code of the $ABC_{imp2}(FEs)$