Algorithm 4. Pseudo-Code of BSA

Input: $ObjFun$, N, D, $maxcycle$, $mixrate$, $low_{1:D}$, $up_{1:D}$
Output: $globalminimum$, $globalminimizer$
// $rnd \sim U(0,1), rndn \sim N(0,1), w = rndint(\cdot), rndint(\cdot) \sim U(1,\cdot) \,|\, w \in \{1,2,3,\ldots,\cdot\}$
$function\ bsa(ObjFun, N, D, maxcycle, low, up)$
// INITIALIZATION
$globalminimum = inf$
for i **from** 1 **to** N **do**
  for j **from** 1 **to** D **do**
    $P_{i,j} = rnd \cdot (up_j - low_j) + low_j$ // Initialization of population, P.
    $oldP_{i,j} = rnd \cdot (up_j - low_j) + low_j$ // Initialization of oldP.
  end
  $fitnessP_i = ObjFun(P_i)$ // Initial-fitness values of P
end
for $iteration$ **from** 1 **to** $maxcycle$ **do**
  // SELECTION-I
  if $(a < b | a, b \sim U(0,1))$ **then** $oldP := P$ **end**
  $oldP := permuting(oldP)$ // 'permuting' arbitrary changes in positions of two individuals in oldP.
  Generation of Trial-Population
    // MUTATION
    $mutant = P + 3 \cdot rndn \cdot (oldP - P)$
    // CROSSOVER
    $map_{1:N,1:D} = 1$ // Initial-map is an N-by-D matrix of ones.
    if $(c < d | c, d \sim U(0,1))$ **then**
      for i **from** 1 **to** N **do**
        $map_{i,u_{(1:\lceil mixrate \cdot rnd \cdot D \rceil)}} = 0 \;\mid\; u = permuting(\langle 1,2,3,\ldots,D \rangle)$
      end
    else
      for i **from** 1 **to** N **do**, $map_{i,randi(D)} = 0$, **end**
    end
    // Generation of Trial Population, T
    $T := mutant$
    for i **from** 1 **to** N **do**
      for j **from** 1 **to** D **do**
        if $map_{i,j} = 1$ **then** $T_{i,j} := P_{i,j}$
      end
    end
    // Boundary Control Mechanism
    for i **from** 1 **to** N **do**
      for j **from** 1 **to** D **do**
        if $(T_{i,j} < low_j) or (T_{i,j} > up_j)$ **then**
          $T_{i,j} = rnd \cdot (up_j - low_j) + low_j$
        end
      end
    end
  end
  // SELECTION-II
  $fitnessT = ObjFnc(T)$
  for i **from** 1 **to** N **do**
    if $fitnessT_i < fitnessP_i$ **then**
      $fitnessP_i := fitnessT_i$
      $P_i := T_i$
    end
  end
  $fitnessP_{best} = min(fitnessP) \mid best \in \{1,2,3,\ldots,N\}$
  if $fitnessP_{best} < globalminimum$ **then**
    $globalminimum := fitnessP_{best}$
    $globalminimizer := P_{best}$
    // Export globalminimum and globalminimizer
  end
end