Nutcracker optimizer: A novel nature inspired metaheuristic algorithm for global optimization and engineering design problems

Outline

- Inspiration
- Nutcracker in nature
- Foraging and storage strategy
- Cache-search and recovery strategy
- Reference memory
- Framework of NOA
- Exploration phase 1(foraging stage)
- Exploitation phase 1 (storage stage)
- Balance between exploration phase1 and exploitation phase1
- Exploration phase 2 (cache-search stage)
- Exploitation phase 2 (recovery stage)
- Balance between exploration phase2 and exploitation phase 2
- Pseudocode of the NOA
- Flowchart of NOA
- advantages of NOA
- Source code of NOA

Inspiration

- The Nutcracker optimization algorithm (NOA) is a nature-inspired algorithm that simulates the distinct behavior of a nutcracker bird that occurs in two separate periods.

- Nutcrackers are intelligent birds with a strong spatial memory.

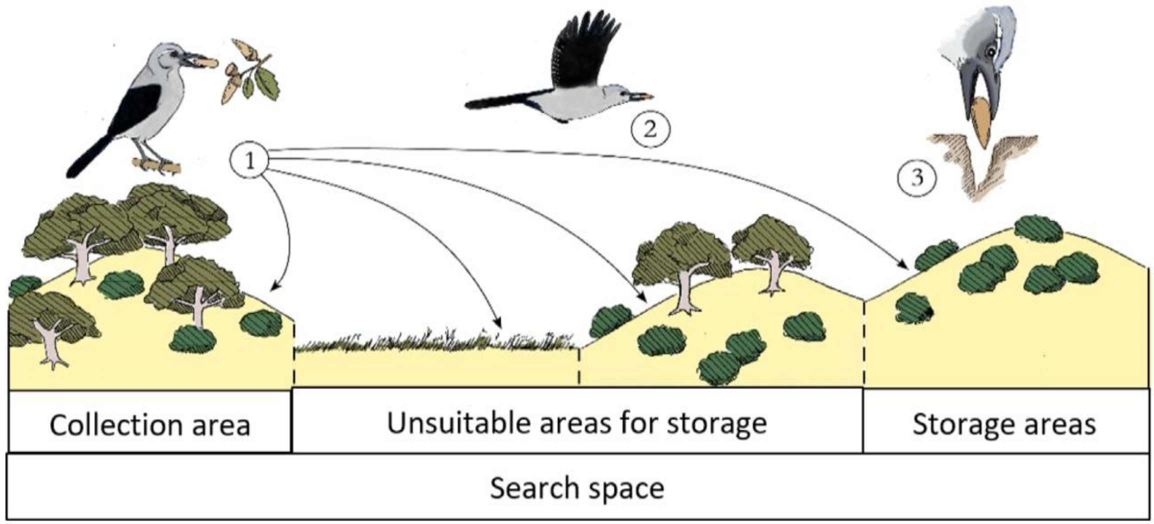
Nutcracker in nature

- Nutcrackers are medium-sized birds with long, sharp bills.
- Pine seeds represent the primary food source of these birds.
- In the summer and fall seasons, the nutcracker searches for seeds and stores them in appropriate caches.
- In the winter and spring seasons, the nutcracker uses its powerful memory to recover previously stored seeds.

Nutcracker bird

Foraging and storage strategy

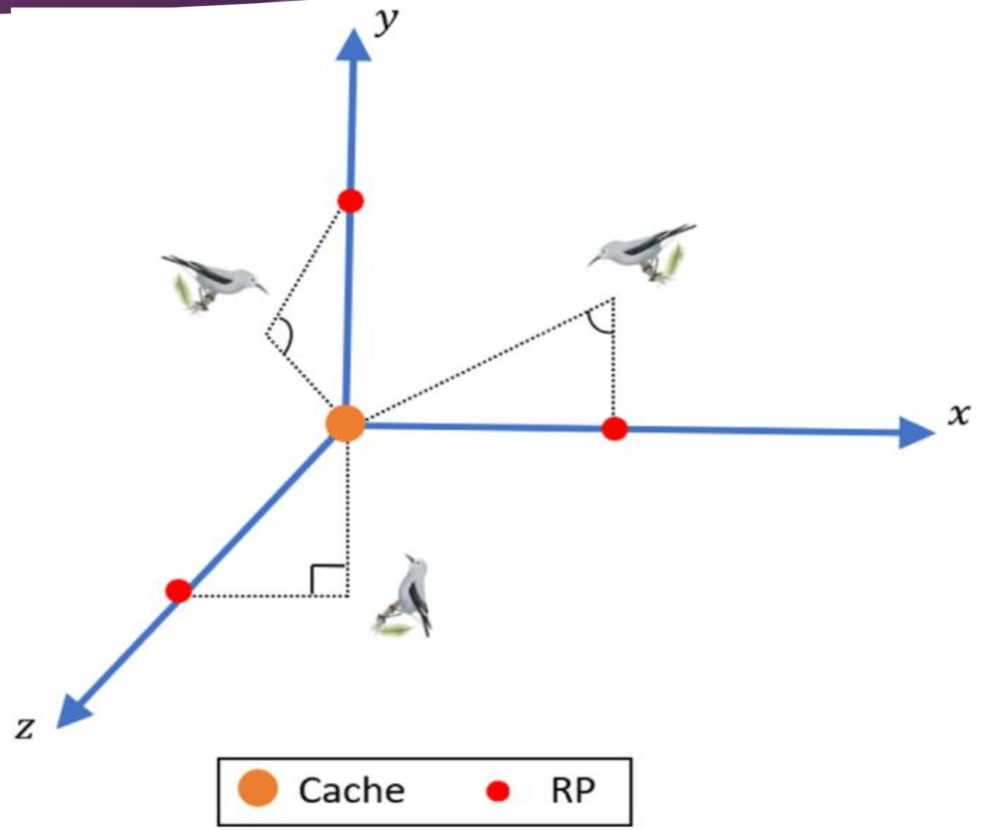
- Nutcracker employs the first strategy, represented by the foraging and storage.
- Nutcracker implements the first strategy in the summer and fall seasons.
- Nutcracker searches for good seeds and stores them, in suitable areas, away from the collection area.



Search space in NOA

Cache-search and recovery strategy

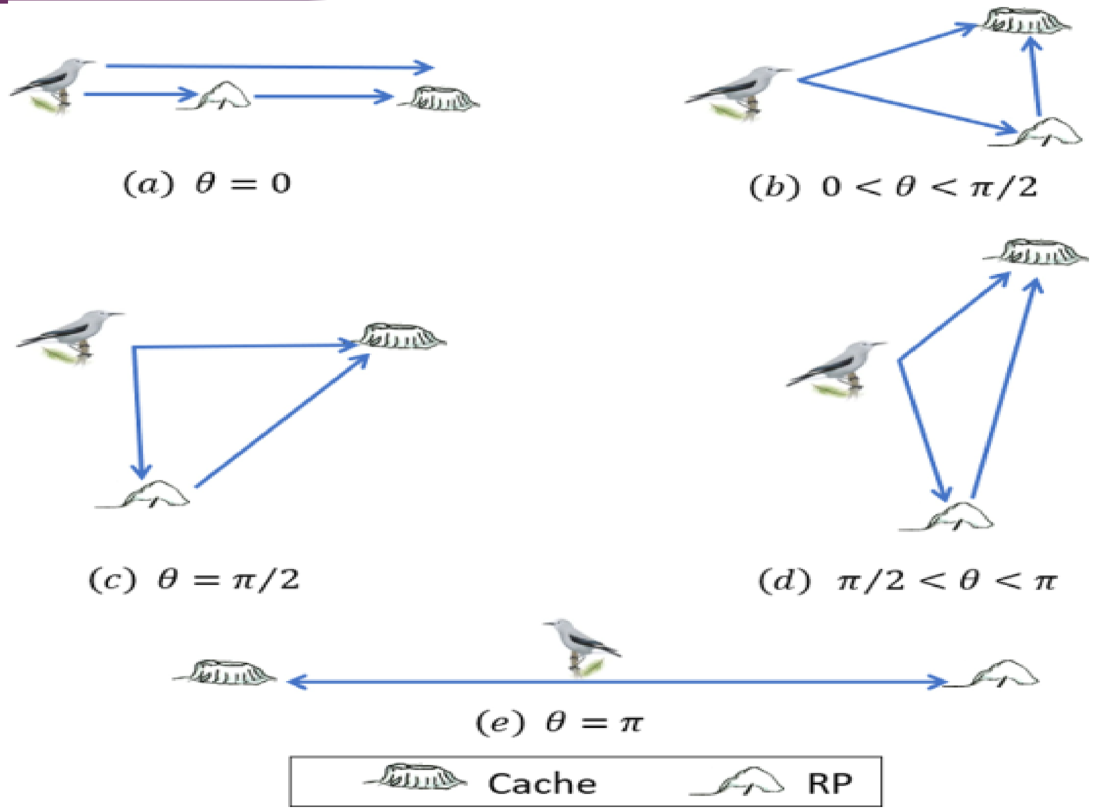
- Nutcracker employs the second strategy, represented by cache-search and recovery.
- Nutcracker implements the second strategy in the winter and spring seasons.
- Nutcracker uses more than one object or mark as reference points (RPs) that help it remember storage locations.



Three different locations for the nutcracker in 3-D space, with three different locations of RP for one cache.

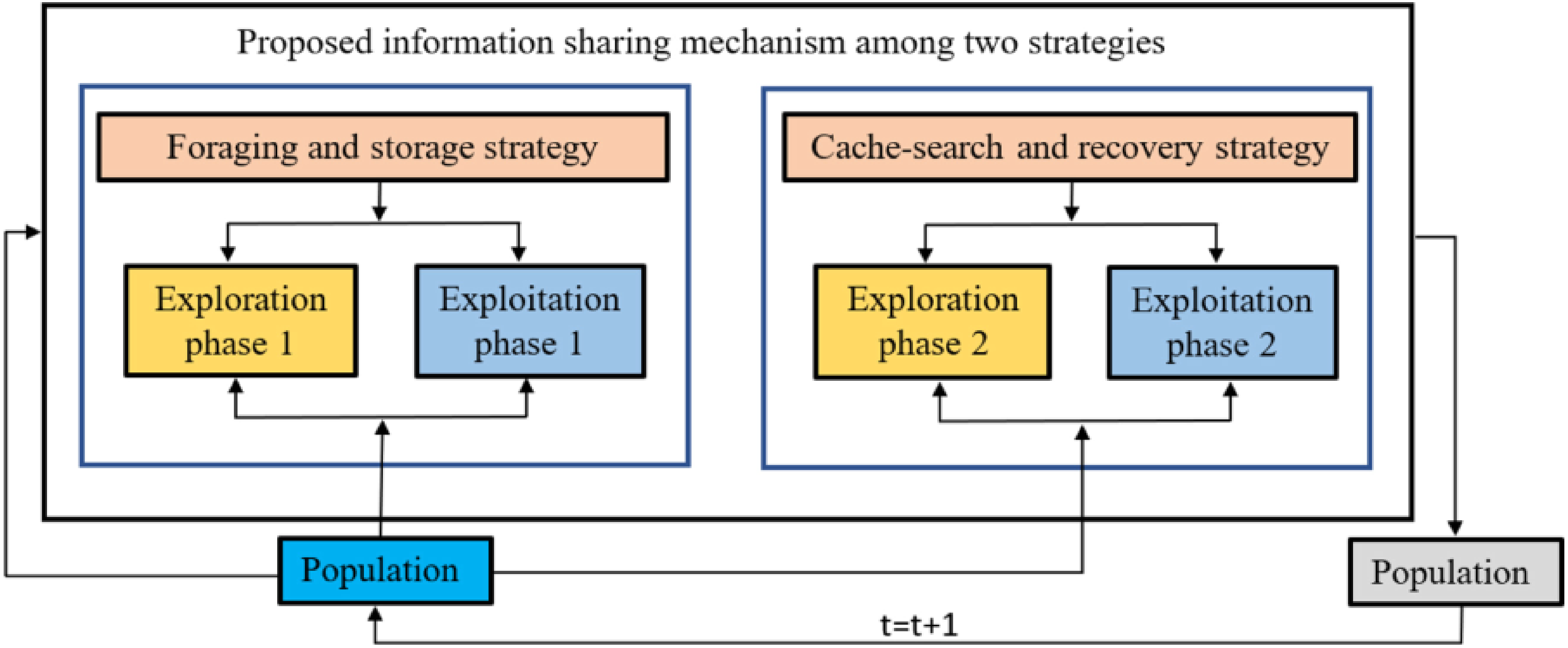
Reference memory

- Nutcracker uses the spatial memory strategy to search for the hidden caches marked at different angles using various RPs.



Different viewing angles of the nutcracker for both cache and RP.

Framework of NOA



Exploration phase 1 (foraging stage)

- At this stage, the nutcracker starts foraging for good seeds in the collection area (the search space). If the nutcracker cannot find good seeds, then it will seek another cone in another position within pine trees or other trees. This behavior can be mathematically modeled using the position update strategy as follows:

$$\vec{X}_i^{t+1} = \begin{cases} X_{i,j}^t & \text{if } \tau_1 < \tau_2 \\ \begin{cases} X_{m,j}^t + \gamma \cdot (X_{A,j}^t - X_{B,j}^t) \\ \quad + \mu . (r^2 . U_j - L_j) , & \text{if } t \le T_{max}/2.0 \\ X_{C,j}^t + \mu \cdot (X_{A,j}^t - X_{B,j}^t) \\ \quad + \mu . (r_1 < \delta) . (r^2 . U_j - L_j) , & Otherwise \end{cases} & otherwise \end{cases} \qquad (1)$$

- $\vec{X}_i^{t+1}$ is the new position of the ith nutcracker in the current generation t; $\vec{X}_{i,j}^t$ is the jth position of the ith nutcracker in the current generation; $\vec{X}_{m,j}^t$ is the mean of the jth dimensions of all solutions of the current population in the iteration t; U_j and L_j are vectors, including the upper and lower bound of the jth dimension in the optimization problem; γ is a random number generated according to the levy flight; τ_1, τ_2, r, and r_1 are random real numbers in the range of [0,1]; A, C, and B are three different indices randomly selected from the population; and μ is a number generated based on the normal distribution.


Exploitation phase 1 (storage stage)

- At this stage, the nutcrackers begin by transporting the food to a storage area, where they exploit pine seed crops and store them. Such behavior can be mathematically expressed as follows:

$$\vec{X}_i^{t+1(new)} = \begin{cases} \vec{X}_i^t + \mu \cdot \left(\vec{X}_{best}^t - \vec{X}_i^t\right) \cdot |\lambda| + r_1 \cdot \left(\vec{X}_A^t - \vec{X}_B^t\right) & if\ \tau_1 < \tau_2 \\ \vec{X}_{best}^t + \mu \cdot \left(\vec{X}_A^t - \vec{X}_B^t\right) & if\ \tau_1 < \tau_3 \\ \vec{X}_{best}^t \cdot l & Otherwise \end{cases} \quad (3)$$

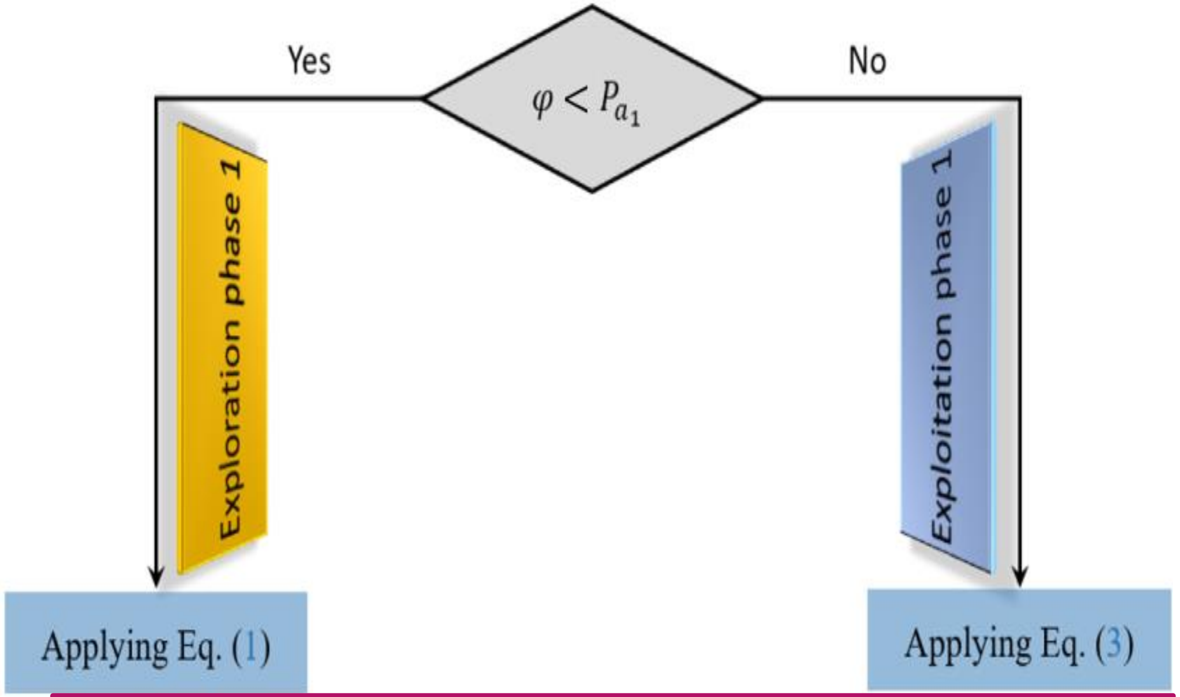
- $\vec{X}_i^{t+1(new)}$ is a new position in the storage area of the nutcrackers in current iteration t; $\vec{X}_{best}^{t}$ is the best position/cache in iteration t λ is a number generated according to the lévy flight, and τ_3 is a random number between 0 and 1; and l is a factor that linearly decreased from 1 to 0 to diversify in the exploitation behavior of NOA. This variety in the exploitation operator of NOA will help in accelerating its convergence speed, in addition to avoiding stuck into local minima that might occur when searching in one direction.

Balance between exploration phase1 and exploitation phase1

- In the NOA, in order to maintain the balance between exploration phase 1 and exploitation phase 1, the following formula is proposed:

$$\vec{X}_i^{t+1} = \begin{cases} \text{Eq. (1)}, & \text{if } \varphi < P_{a_1} \\ \text{Eq. (3)}, & \text{otherwise} \end{cases} \tag{4}$$

- φ is a random number between zero and one, and P_{a_1} represents a probability value that is linearly decreased from one to zero based on the current generation.



Flowchart of the exploration and exploitation process in the foraging and storage strategy

Exploration phase 2 (cache-search stage)

- At this stage, the Nutcrackers begin to search and explore their caches.
- Nutcrackers use a spatial memory strategy to locate their caches.
- Nutcrackers most likely use multiple objects as signals for a single cache.
- For simplicity, we will assume that each cache has only two objects. In NOA, two RPs of each cache/nutcracker of the population can be defined using the following matrix:

$$
\text{RPs} = \begin{bmatrix} \overrightarrow{RP}^{t}_{1,1} & \overrightarrow{RP}^{t}_{1,2} \\ \vdots & \vdots \\ \overrightarrow{RP}^{t}_{i,1} & \overrightarrow{RP}^{t}_{i,2} \\ \vdots & \vdots \\ \overrightarrow{RP}^{t}_{N,1} & \overrightarrow{RP}^{t}_{N,1} \\ \vdots & \vdots \end{bmatrix}
$$

- represent $RP^t_{i,1}$ and $RP^t_{i,2}$ (objects) of the cache position X^t_i of the ith nutcracker in the current generation t.

Exploration phase 2 (cache-search stage) (cont.)

- The first and second RPs are generated by updating the current position within the neighboring regions to find hidden caches around the nutcrackers. The mathematical formula for generating the first and second RPs are as follows:

$$\overrightarrow{RP}_{i,1}^{t} = \begin{cases} \vec{X}_i^t + \alpha \cdot \cos(\theta) \cdot \left(\left(\vec{X}_A^t - \vec{X}_B^t\right)\right) + \alpha \cdot RP, & \text{if } \theta = \pi/2 \\ \vec{X}_i^t + \alpha \cdot \cos(\theta) \cdot \left(\left(\vec{X}_A^t - \vec{X}_B^t\right)\right), & \text{otherwise} \end{cases} \quad (9)$$

$$\overrightarrow{RP}_{i,2}^{t} = \begin{cases} \vec{X}_i^t + \left(\alpha \cdot \cos(\theta) \cdot \left(\left(\vec{U} - \vec{L}\right) \cdot \tau_3 + \vec{L}\right) + \alpha \cdot RP\right) \cdot \vec{U}_2, & \text{if } \theta = \pi/2 \\ \vec{X}_i^t + \alpha \cdot \cos(\theta) \cdot \left(\left(\vec{U} - \vec{L}\right) \cdot \tau_3 + \vec{L}\right) \cdot \vec{U}_2, & \text{otherwise} \end{cases} \quad (10)$$

- $RP_{i,1}^{t}$ and $RP_{i,2}^{t}$ repersent the first and the second RP of the cache position $\vec{X}_i^t$ of the ith nutcracker in the current iteration t; $\vec{r_2}$ is a vector that includes values randomly generated in the range [0, 1]; $\vec{X}_A^t$, $\vec{X}_B^t$ are the cache positions of the *Ath* and *Bth* nutcrackers, respectively, in the iteration t; RP is a random position; θ is a random in the range [0, π]; and $\vec{U_2}$ is a random number equal to 1 if $\vec{r_2} < P_{rp}$, otherwise equal to zero, P_{rp} is a probability employed to determine the percentage of globally exploring other regions within the search space.

Exploration phase 2 (cache-search stage) (cont.)

- α ensures that the NOA converges on a regular basis, allowing the nutcracker to improve its RP selection in the next generations. α can be calculated according to the following equation:

$$\alpha = \begin{cases} \left(1 - \frac{t}{T_{max}}\right)^{2\frac{t}{T_{max}}}, & \text{if } r_1 > r_2 \\ \left(\frac{t}{T_{max}}\right)^{\frac{2}{t}}, & \text{otherwise} \end{cases} \quad (11)$$

- t and T_{max} indicate the current and maximum generations, respectively. The first state in Eq. (11) linearly decreases with the iteration to improve the convergence speed of the NOA. Meanwhile, the second state linearly increases to avoid being stuck into local minima, which might occur because of the first state.

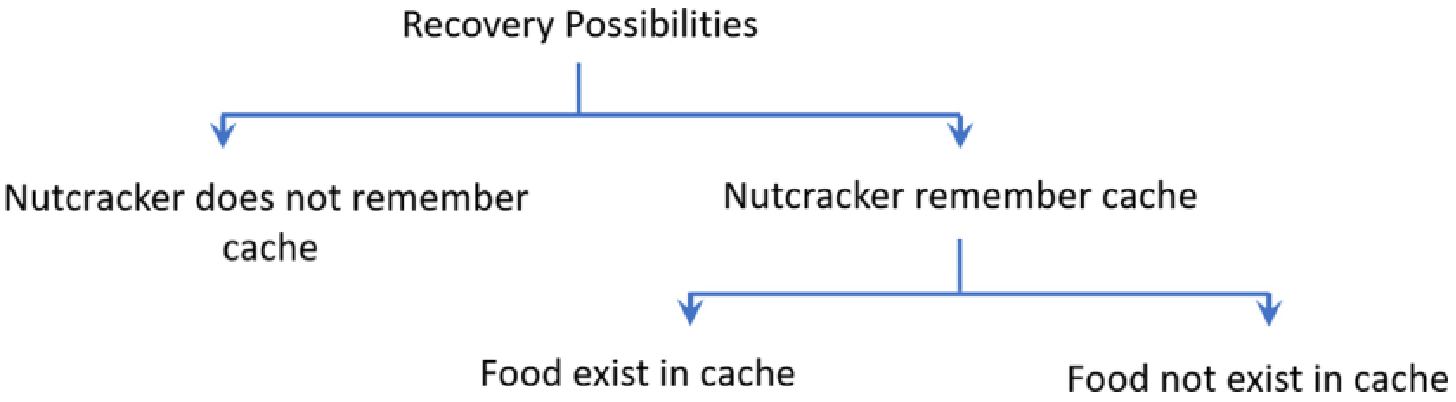
Exploration phase 2 (cache-search stage) (cont.)

- In NOA, all nutcrackers will apply the exploration mechanism to search for the most promising areas that might contain a near-optimal solution. With each generation passed, the algorithm will explore and exploit areas around caches with appropriate RPs to avoid getting stuck in local minima. The new position of a nutcracker can be updated using the following equation:

$$\vec{X}_i^{t+1} = \begin{cases} \vec{X}_i^t, & if\ f\left(\vec{X}_i^t\right) < if\ (\overrightarrow{RP}_{i,1}^t) \\ \overrightarrow{RP}_{i,1}^t, & otherwise \end{cases} \tag{12}$$

Exploitation phase 2 (recovery stage)

- At this stage, the Nutcracker tries to recover its cache. The following scheme (recovery scheme) depicts the possibilities that a Nutcracker might encounter when searching for its cache:



Probable options for the Nutcracker to recover its cache.

- The following equation simulates the first possibility (Nutcracker remember cache)

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t}, & if \;\; \tau_3 < \tau_4 \\ X_{i,j}^{t} + r_1 \cdot \left(X_{best,j}^{t} - X_{i,j}^{t}\right) + r_2 \cdot \left(\overrightarrow{RP}_{i,1}^{t} - X_{C,j}^{t}\right), & otherwise \end{cases} \tag{13}$$

Exploitation phase 2 (recovery stage) (cont.)

- The following equation simulates the second possibility (Nutcracker does not remember cache):

$$\vec{X}_i^{t+1} = \begin{cases} \vec{X}_i^t, & if\ f\left(\vec{X}_i^t\right) < if\ \left(\overrightarrow{RP}_{i,2}^t\right) \\ \overrightarrow{RP}_{i,2}^t, & otherwise \end{cases} \qquad (14)$$

- Eq. (14) offers an opportunity for the NOA to explore new regions around the second RP and exploit promising areas where a potential solution could be found.
- In NOA, a nutcracker is assumed to find its cache using the second RP, so Eq. (13) is updated based on the second RP using the following equation:

$$X_{ij}^{t+1} = \begin{cases} X_{ij}^t, & if\ \tau_5 < \tau_6 \\ X_{ij}^t + r_1 \cdot \left(X_{best,j}^t - X_{ij}^t\right) + r_2 \cdot \left(\overrightarrow{RP}_{i,2}^t - X_{Cj}^t\right), & otherwise \end{cases} \qquad (15)$$

- $r_1, r_2, \tau_4\ and \tau_5$ are a random number between 0 and 1.


Exploitation phase 2 (recovery stage) (cont.)

- In summary, the simulation of the recovery behavior (recovery scheme) can be summarized in the following

$$\vec{X}_i^{t+1} = \begin{cases} \text{Eq. } (13), & \text{if } \tau_7 < \tau_8 \\ \text{Eq. } (15), & \text{otherwise} \end{cases} \quad (16)$$

- τ_7 and τ_8 are a random number between 0 and 1.
- The following equation is proposed to achieve the trade-off between exploration behaviors about the first and second RPs:

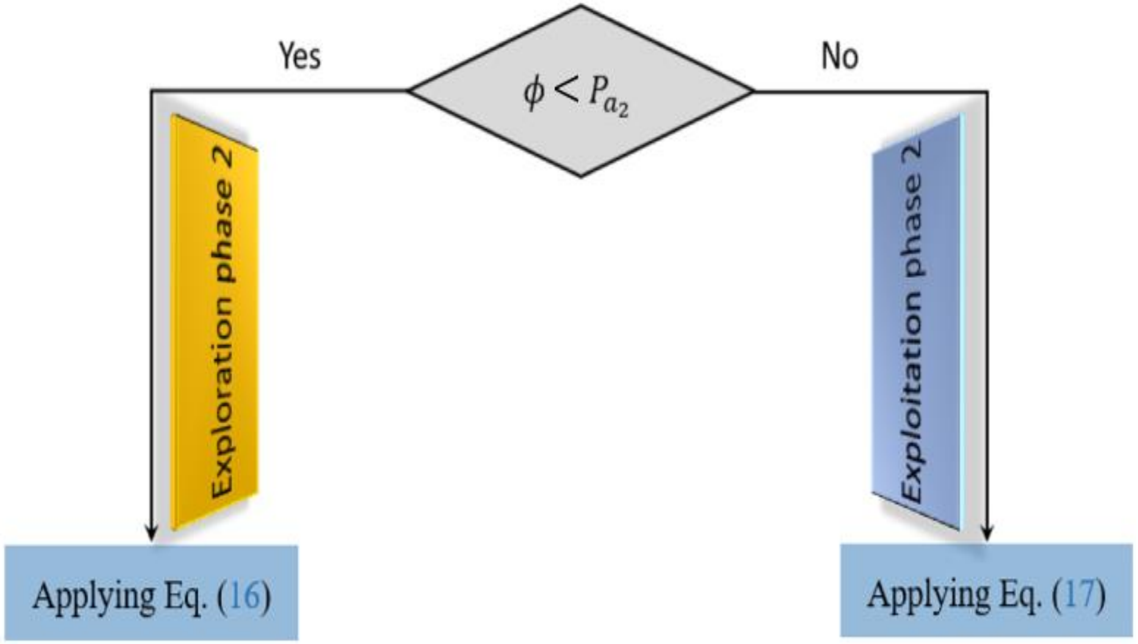
$$\vec{X}_i^{t+1} = \begin{cases} \text{Eq. } (12), & \text{if } f(\text{Eq. } (12)) < f(\text{Eq. } (14)) \\ \text{Eq. } (14), & \text{otherwise} \end{cases} \quad (17)$$

Balance between exploration phase2 and exploitation phase2

- In the NOA, in order to maintain the balance between exploration phase 2 and exploitation phase 2, the following formula is proposed:

$$\vec{X}_i^{t+1} = \begin{cases} \text{Eq. (16)}, & \text{if } \phi < P_{a_2} \\ \text{Eq. (17)}, & \text{otherwise} \end{cases} \quad (18)$$

- ϕ is a random number between zero and one, and P_{a_2} represents a probability value that is equal to 0.2.



Flowchart of the exploration and exploitation process in the foraging and storage strategy

Pseudocode of the NOA

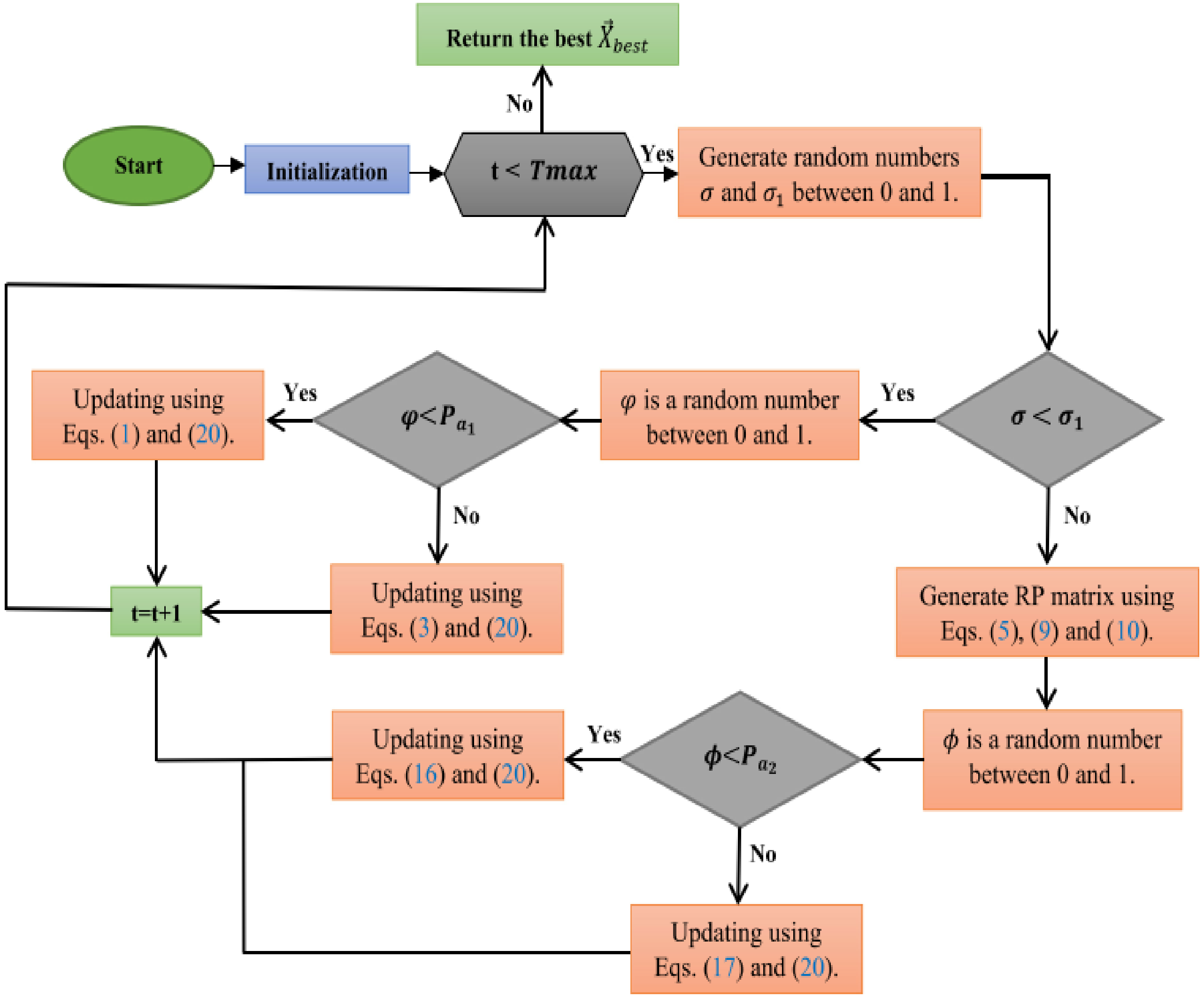
Algorithm 3 Pseudo-code of NOA

Input: population size N, the lower limits of variables $\vec{L}$, the upper limits of variables $\vec{U}$ the current number of iteration t=0, and the maximum number of iterations T_{max};

Output: the best solution found

Initialize N nutcracker/solution using Eq.(19);
Evaluate each solution and find the one with the best fitness in the population
$t = 1$; //the current function evaluation//
while $(t < T_{max})$
    Generate random numbers σ and σ_1 between 0 and 1.
    If $\sigma < \sigma_1$ //* ***Foraging and storage strategy****//
        φ is a random number between 0 and 1.
        for *i=1:N*
            for j= 1:*d*
                if $\varphi < P_{a_1}$ /***Exploration phase1***/
                    Updating $\vec{X}_i^{t+1}$ using Eq. (1) and Eq. (20)
                else /***Exploitation phase1***/
                    Updating $\vec{X}_i^{t+1}$ using Eq. (3) and Eq. (20)
                end if
            end for
            Update the current iteration t by $t = t + 1$
        end for
    else //* ***Cache-search and recovery strategy*** *//
        Generate RP matrix using Eq. (5), Eq.(9) and Eq.(10).
        Generate a random number ϕ between 0 and 1.
        for *i=1:N*
            if $\phi < P_{a_2}$ /***Exploration phase2***/
                Updating $\vec{X}_i^{t+1}$ using Eq. (16) and Eq. (20)
            else /***Exploitation phase2***/
                Updating $\vec{X}_i^{t+1}$ using Eq. (17) and Eq. (20)
            end if
            $t = t + 1$
        end for
end while

Flowchart of the NOA



Advantages of NOA

- Easy to implement.
- Able to avoid falling into local optima for several optimization problems with various characteristics.
- Having a high convergence speed.

Source code of NOA

The source code of NOA is publicly available at https://www.researchgate.net/publication/366921723_Nutcracker_optimizer_NOA_MATLAB_Code