YUKI algorithm

This algorithm suggests a principle of dynamic search space reduction. By creating a local search area around the absolute best solution found so far. The algorithm adapts the size of the search area dynamically throughout the progress of the search, using the distance between two points as a reference.

The first point is the absolute best point. It corresponds to the minimum fitness value. Furthermore, The second point is called MeanBest. We calculate this point as the center of the "best points" could. The best points here are the best solutions found so far by each member of the YA population.

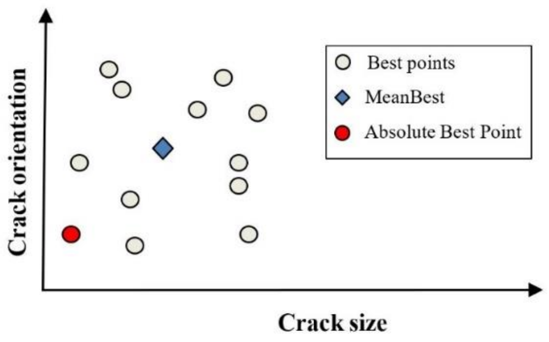

Figure 1. Illustration of the MeanBest point.

The following expression calculates the local boundaries:

$$D = |X_{best} - X_{MeanBest}| \tag{1}$$

$$\boldsymbol{LT} = X_{best} + D \tag{2}$$
$$\boldsymbol{LB} = X_{best} - D \tag{3}$$

Respectively $\boldsymbol{LT}$ and $\boldsymbol{LB}$ are the Local top and bottom boundaries of the local search space. Furthermore $\boldsymbol{X_{best}}$ Is the point corresponding to the absolute best fitness value at the current iteration. And the term $\boldsymbol{X_{MeanBest}}$ is the mean of the Best Points vector.

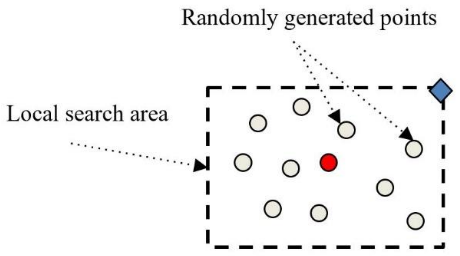

Figure 2. Illustration of the local search area. The red circle stands for the absolute best point, and the blue rhombus represents the MeanBest point.

The second concept is to create a random distribution of points $\boldsymbol{X_{loc}}$ Inside the local search area, split the search into two parts; One part of the population to explore outside the local search area. The other is assigned to focus on searching around the center of the search area. The size of the exploration population changes iteratively in this manner:

$$if\ rand < \ 1 - K\ /\ K_{max} \tag{4}$$

Eq. 4 condition compares the randomly generated value, between 0 and 1, to the output of the Eq. 4 expression, which is also between 0 and 1. $\boldsymbol{K}$ is the current iteration and $\boldsymbol{K_{max}}$ Represent the value of maximum iteration set initially as a stopping criterion. At early iterations, the output corresponds to a high possibility for the “if statement” to be True, and at late iterations, this possibility decreases linearly. This way, we assign most of the population toward exploring initially, then slowly shift toward exploitation.

Exploration population look in the direction away from the MeanBest expressed as follows:

$$E = X_{loc} - randi(10) \times X_{MeanBest} \tag{5}$$

Where $\boldsymbol{randi}(\mathbf{4})$ is a random integer between 1 and the value of the exploration control parameter chosen as an integer between 2 and 10, the following equation calculates the new solutions:

$$X_{new} = X_{loc} + rand \times E \tag{6}$$

“rand” here stands for a random value between 0 and 1. Note here that we use the same random value for all design variables.

The rest of the population, not assigned to exploration, is pushed to search around the local area center. The following equation governs this process:

$$\boldsymbol{F} = X_{loc} - X_{best} \tag{7}$$

$$X_{new} = X_{loc} - rand \times \boldsymbol{F} \tag{8}$$

Where $\boldsymbol{F}$ is the distance between the selected local point to the absolute best solution, similarly, *rand* here is the random value between 0 and 1. Note also here that we use the same random value for all design variables.

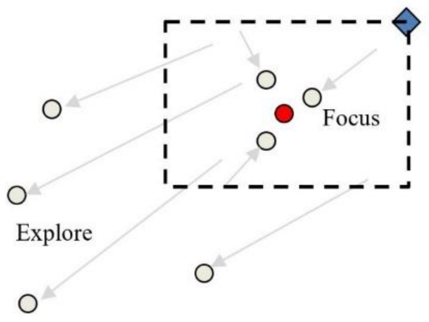

Figure 3. Illustration of the exploration and focus concepts.

We update the absolute best solution and the individual best solutions at the end of each iteration. For details, Fig.4 shows the YA pseudo-code.

In this study of crack identification, the fitness function is:

$$\begin{cases} f(\boldsymbol{P}) = \dfrac{\|\boldsymbol{u}(\boldsymbol{P}_0) - \boldsymbol{u}(\boldsymbol{P})\|^2}{\|\boldsymbol{u}(\boldsymbol{P}_0)\|^2} \\ f(\boldsymbol{P}_{optimal}) = min\,[f(\boldsymbol{P})] \end{cases} \tag{9}$$

Where the response vectors $\mathbf{u}(\mathbf{P})$ of the suggested cracks is compared to the reference response $\mathbf{u}(\mathbf{P}_0)$ That was caused by the unknown crack.

The following points discuss the theory of the suggested algorithm.

- By exporting new areas situated away from the MeanBest point, we focus the exploration effort in one direction, which increases the possibility of finding new solutions compared to randomly exploring the global area.
- By multiplying the MeanBest value to a random integer, we create a pulsating effect, essential for suggesting different solutions in every iteration. It also helps extend the reach of the search.
- As long as there is good coverage of the global search space, the distance between Best Points remains safely large. Thus the local search area avoids collapsing on a local optimum.
- The search area sizes are independent across search dimensions. Because this algorithm can create variations for which the local search area is significant in one dimension and very small in another dimension. Thus the search focus is reacting to the sensitivity of each design variable.
- The local search area's size can increase dynamically if the algorithm finds a new best solution far from the current local search area.
- As solutions converge to the optimum, the local search area gets smaller,

```
Yuki algorithm pseudo-code
Load the search parameters
Initiate the population $X$
Evaluate the fitness
Calculate the $X_{MeanBest}$ and $X_{best}$
**for** $K = 1$ to $K_{max}$
   Calculate the local boundaries
   Generate the local population randomly
   **if** $rand < 1 - K / K_{max}$
      Calculate the exploration solutions
   **else**
      Calculate focus solutions
   **end**
   update the $X_{MeanBest}$
   update the $X_{best}$ if better solutions are found
**end**
return $X_{best}$
```

Figure 4. YUKI Algorithm.