2 Artificial Bee Colony (ABC)

Similar to Particle Swarm Optimization, Artificial Bee Colony or ABC is a meta-heuristic algorithm that mimics the intelligence swarm behavior of bees. Worker bees are grouped into three classes, employer, onlooker and scout. ABC algorithm is based on the coordination between these worker bees trying to retrieve the highest amount of nectar. With this intelligence swarm of worker bees, ABC algorithm could be used to solve optimization problem including multi-objective optimization problem, MOP.

In the next subsections, single objective and multi-objectives ABC algorithm will be explained. Then, the implementation of multi-objectives ABC, MOABC will be explained. Lastly, the result after testing the algorithm with two MOP will be discussed.

2.1 Algorithm Explanation: ABC

From the analogy above, each nectar or food source lies in a coordinate grid with the same dimension as the problem. A food source represents a solution, and the quantity of nectar or food is representing the fitness of the corresponding solution. The task for worker bees is to find the food source with highest quantity of food.

A good optimization algorithm fundamentally needs two elements, exploration and exploitation. ABC algorithm aligns with this rule. The algorithm consists of three phase, employer phase, onlooker phase and scout phase. Employer phase and scout phase are responsible for exploration element while onlooker phase is responsible for exploitation element of the algorithm. The algorithm's flow chart is shown in Figure 5 below.

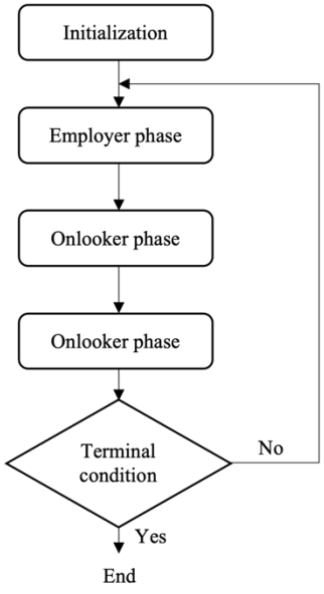

Figure 5 ABC algorithm flow chart

First, the algorithm starts with randomly initialize food sources with the same amount as of the employer or onlooker, let's define it as *n*. The food source is a vector with *N* dimensions which is the same as the dimension of the problem.

In employer phase, each employer starts at different initialized food sources. Then, each employer moves toward each other randomly to find new food sources. The movement can be formulated as follow,

$$\boldsymbol{v}_i = \boldsymbol{x}_i + \phi_i(\boldsymbol{x}_j - \boldsymbol{x}_i) \tag{7}$$

$\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is the food source employed by i^{th} and j^{th} employer. ϕ_i is a random number within [-1,1] and $\boldsymbol{v}_i$ is the new food source found by i^{th} employer. The fitness for each of the new food source is calculated, if there isn't an improvement for fitness, employer will return to the original food source. Then each employer will return to the hive and gives onlooker bee an information about fitness of the food source which then the onlooker phase begins.

In onlooker phase, each onlooker starts at different food sources found by employers. Then, instead of randomly choosing each other, onlookers will prioritize the direction of food source with high fitness. The probability of choosing the i^{th} food source as a moving direction is shown below in equation (8)

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{8}$$

,where f_i is the fitness of the i^{th} food source. The movement is the same as in the employer phase.

Scout phase is when the food source is decided to be abandoned. Scouts keep track of which food source has been selected as a moving direction either by employers or scouts. Food sources that had not been selected for a limited amount of time denoted by parameter *L*, abandonment limit counter, will be re-initialized.

The algorithm terminates after a terminal condition is satisfied; maximum number of iterations reached for instance. The food source with highest fitness is the solution to the optimization problem found by ABC algorithm.

2.2 Algorithm Explanation: MOABC

For multi-objective optimization problem, MOP, the algorithm needs some modification to effectively solve the problem. Unlike single-objective optimization, worker bees can't focus on moving toward one best goal because in MOP, the goal is the Pareto front which composed of many nondominanted solutions. Just like the sigma method used in MOPSO, Wenping Zou's study used External Archive together with Comprehensive Learning to cope with this problem. These two techniques will be explained in the following subsections.

2.2.1 External Archive

In case of sigma method, a certain amount of nondominated particles is selected to be guiding particle based on their sigma and fitness value. Each sigma bin will contain only one guiding particle and every particle in the same sigma bin will move toward this guiding particle.

Similarly, External Archive, EA, is a set of nondominated solutions among the solution set or food sources in case of ABC. During onlooker phase, instead of selecting food sources with high fitness, onlooker will randomly select one of the food sources in EA instead. Note that this random selection is similar to the improved sigma method I used in MOPSO in section 1.2.2.

EA is always a set of nondominated solutions so every time employer or onlooker found a new food source that dominates the original one, it must be added to EA. At the same time, food sources originally in EA that get dominated by newly added food source must be removed. However, if employer or onlooker found a new food source that does not dominate nor get dominated by the original food source, according to Wenping's study, this newly found food source will either be added to EA or ignored with probability of 0.5. Wenping's study referred to this method as greedy selection.

Obviously, time took to update EA is proportional to the size of EA. In case of sigma method, the size of guiding particles is limited by the amount of sigma bins. For EA, the size is limited by an adjustable parameter. When the size of EA exceeds the limit, some of the solutions will be discarded from EA which determined using crowding distance calculation. Crowding distance of the boundary solutions for every objective function is set to infinity while crowding distance of non-boundary solutions x_i denoted by $CD(x_i)$ is shown in the following Equation (9) and (10).

$$CD(x_i) = \frac{1}{N}\sum_{k=0}^{N} dist(x_i, k) \tag{9}$$

$$dist(x_i, k) = (x'_{s+1,k} - x'_{s,k}) + (x'_{s,k} - x'_{s-1,k}) \tag{10}$$

dist function of solution x_i and dimension k is calculated by sorting the solutions in EA by their k^{th} objective value and calculate the summation of distances between x_i and its two adjacent solutions. In other word, if sorted solutions are in $x'_{0\ldots N}$ order, if $x'_s = x_i$ and it isn't boundary solution (s is not *0* or *N*) then the *dist* function can be calculated with Equation (10).

The method of elimination isn't clear in the study, so my implementation was to eliminate the solution with smaller crowding distance first. It might not be optimal because it doesn't guarantee to maximize the aggregated crowding distances of solution in EA. Other way that is possible is elimination by probability based on crowding distance. However, it's quite complicated and inefficient if a certain amount of solution needed to be eliminated.

2.2.2 Comprehensive Learning

Wenping suggested in the study that despite of EA, the MOABC algorithm still needs more diversity in solutions exploration. In the study, Comprehensive Learning, CL, is being used to

mitigate this problem. During onlooker phase, instead of using only one food source as a moving direction, random *m* dimensions of the food source vector are moving toward one nondominated solution in EA and other dimensions are moving toward other random nondominated solutions in EA. The movement equation for Comprehensive Learning becomes,

$$v_{i,f(m)} = x_{i,f(m)} + \phi(m)(EA_{k,f(m)} - x_{i,f(m)}) \quad (11)$$

$$v_{i,j} = x_{i,j} + \phi_{i,j}\,(EA_{i,j} - x_{i,j}) \quad (12)$$

f(m) is the *m* dimensions that moves toward the same nondominated solution in EA represented by EA_k. Other dimensions of the i^{th} food source move according to the Equation 12.

1.3 Implementation

Many parts of the algorithm proposed in Wenping's study is unclear, such as the omitted employer phase in algorithm 3 pseudo code, so many part of the algorithm is implemented as in the following subsection using Python together with NumPy and Random library. The two problems, Multi-model MOP and Biased Pareto-optimal front problem, are the same as the previous MOPSO section.

2.3.1 MOABC implementation

Pseudo code for my MOABC implementation is as below in Figure 6. The implementation of MOABC is done in a class called **ABC**. The initialization (1-3) is done in *init_ABC* method. The main loop started in *train* method which takes one argument, maximum iteration, as a termination condition. As shown in the pseudo code, employer phase and onlooker phase have a very similar operation. Beside target selection, the code for the rest of the operation (6-8 and 10-12) is implemented under a method called *beesRoutine*. After finished training, maximum iteration reached, the *train* method will return the objectives of the food source in the last iteration. The food source in EA can also be accessed through object's attribute.

Repositioning function is the same as the MOPSO algorithm where the worker bees will moved to the closest problem's constraint boundary if they went outside.

The current implementation works fine but quite slow to converge during the tests so I added accelerator parameter to the movement equation. The movement equation becomes,

$$\boldsymbol{v}_i = \boldsymbol{x}_i + \phi_i a(\boldsymbol{x}_j - \boldsymbol{x}_i) \quad (13)$$

The algorithm was tested using the two MOPs and the visualization of the last iteration is presented in the following section.

1. Initialize food source position
2. Calculate objective functions for each food source
3. Find nondominated food source and store in EA
4. Repeat the following until termination condition is satisfied

Employer phase

5. Choose random food source position as a target for each employer
6. Calculate new food source positions
7. Use greedy selection and update EA
8. Eliminate solutions in EA if the size exceeds the limit

Onlooker phase

9. Choose one food source from EA and random N-m food source from EA for Comprehensive learning for each onlooker
10. Calculate new food source positions
11. Use greedy selection and update EA
12. Eliminate solutions in EA if the size exceeds the limit

Scout phase

13. Find and replace food sources that have abandonment counter reached the limit.
14. Update EA
15. Eliminate solutions in EA if the size exceeds the limit

Figure 6 MOABC pseudo code

1.4 Result and Discussion

The results of the MOABC algorithm for both Multi-model MOP and Biased Pareto-optimal front problem are shown in the following subsection. The result will be discussed in each subsection. The parameter for MOABC algorithm is set as follow for every tests.

Table 3 MOABC Parameter settings

Parameters	Value
Colony_size	10000
Abandonment counter limit's *L*	100
Comprehensive Learning's *m*	1
EA_size_limit	100
acceleration	1.5
max iteration	100

2.4.1 Multi-model MOP

The result of MOABC for Multi-model MOP is shown below in Figure 7. Although some of the worker bees weren't converged to Pareto front within 100 iterations, most of them did and equally distributed across the frontier.

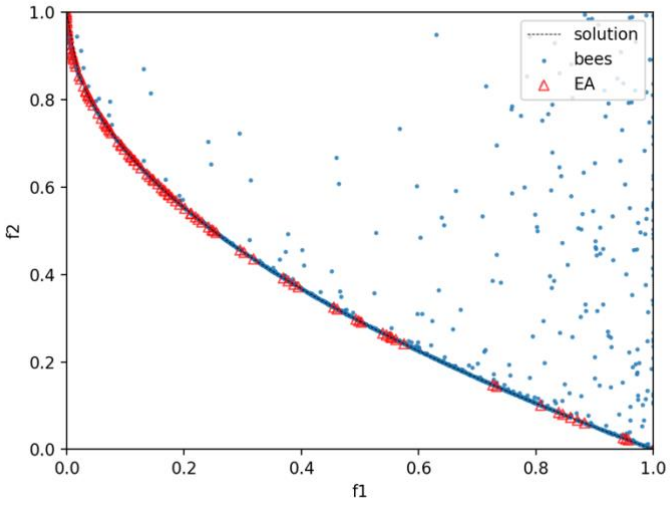

Figure 7 Plots of Multi-Model MOP objective values of food sources of worker bees and External Archive

2.4.2 Biased Pareto-optimal front problem

Same as the previous section in MOPSO, the Biased Pareto-optimal front problem was tested with its two variation, concave down (alpha=2) and concave up (alpha=0.5). The result is shown below.

Concave down (alpha=2)

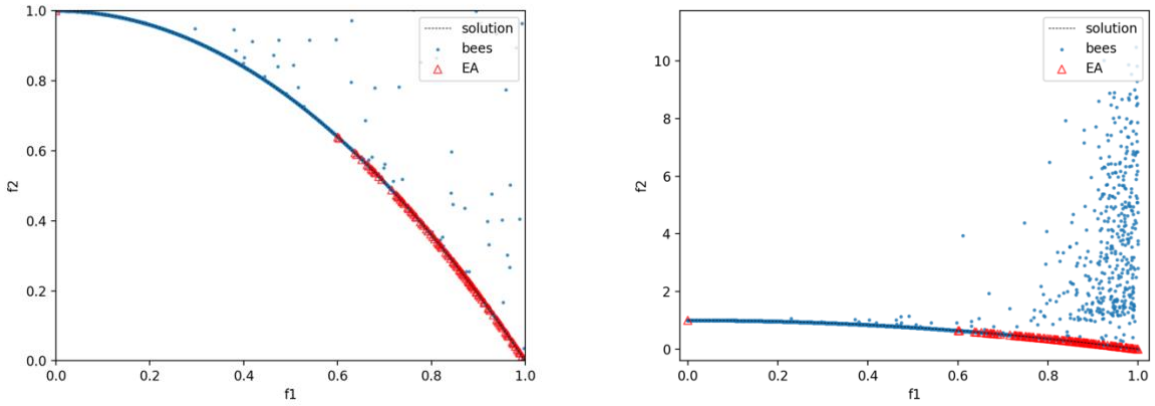

Figure 8 Plots of Biased Pareto-optimal front (concave down) objective values of food sources of worker bees and External Archive. Extended view shown on the right.

Figure 8 shows the result of testing MOABC with Biased Pareto-optimal front problem with concave down variation. Solutions found by the algorithm lies on the Pareto front and distributed across the frontier. However, the solution in EA is quite biased toward one side. On the right plot of Figure 8 is the extended view of the solutions. Interestingly, most of the worker bees that's yet to converge also moving toward one side of the plot. This might have some positive

feedback looped between these phenomena that resulted in Figure 8. Further investigation such as plots of each iteration's solutions is needed.

Concave up (alpha=0.5)

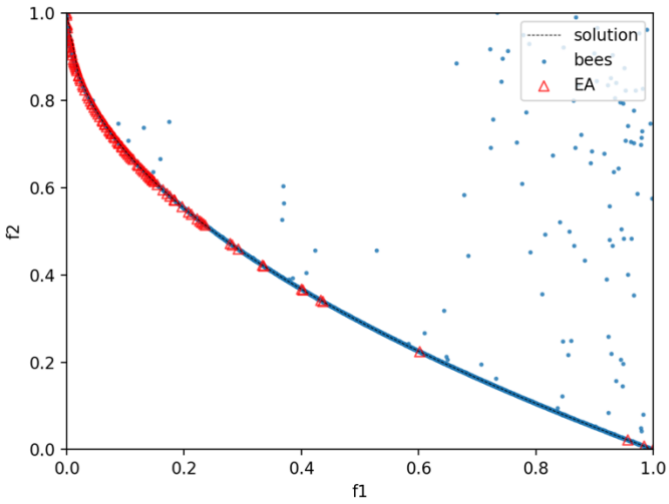

Figure 9 Plots of Biased Pareto-optimal front (concave up) objective values of food sources of worker bees and External Archive

Similar to the concave down variation, most of the solutions converged on the Pareto front and distributed across the frontier while the solutions in EA is biased toward one side.

Course comment

- The contents of the course are very interesting and doesn't focus only one field but rather diversified. I think it was very good this way because these contents broaden my view on the subject.
- Personally I really like AI implemented with meta-heuristic.
- I hope the course slides are distributed.