# Archimedes Optimization Algorithm (AOA)

## Modern Optimization Methods

The modern optimization methods (also called nontraditional optimization methods) have emerged as powerful and popular methods for solving complex engineering optimization problems in recent years. These methods are simulated annealing, evolutionary programming (EP), Tabu search (TS), Neural-network based methods, genetic algorithm (GA), differential evolution (DE) algorithm, particle swarm optimization (PSO) technique, seeker optimization algorithm (SOA), ant colony optimization (ACO) [5] algorithm, Bat algorithm (BA) [4][6] and Archimedes Optimization Algorithm (AOA). Most of these methods are labeled on certain characteristics and behavior of biological, molecular, swarm of insects and neurobiological systems. These new meta-heuristic tools have been combined among themselves and with knowledge elements, as well as with more traditional approaches such as statistical analysis to solve extremely challenging problems.

## Archimedes Optimization Algorithm

### Introduction

Archimedes Optimization Algorithm (AOA) [7] is based on Archimedes’ principle which states that “Any object, totally or partially immersed in a fluid or liquid, is buoyed up by a force equal to the weight of the fluid displaced by the object.”. AOA emulates the behavior of many objects, which have different densities and volumes, immersed in the same fluid and each one tries to reach equilibrium state.

### AOA Theory

If we assume that many object are immersed in the same fluid and each one of them tries to reach equilibrium state. The object will be in the equilibrium state if the buoyant force equal to the object’s weight :

|  |  |  |
| --- | --- | --- |
|  |  | ‑ |

Where is the density, is the volume, and is the gravity or acceleration, subscripts and are for fluid and immersed object, respectively. This equation can be rearranged as:

|  |  |  |
| --- | --- | --- |
|  |  | ‑ |

If there is another force influenced on the object like collision with another neighboring object (), the equilibrium state will be:

|  |  |  |
| --- | --- | --- |
|  |  | ‑ |

### Algorithmic steps

**5.2.3 BBA Algorithm**

The process of BBA for solving the optimization problems can be summarized as follows:

***Step 1: Initialization***

Insert the follows:

* Insert the control variables or bat positions that represent the locations and sizes od DGs and capacitors between the minimum and maximum limits as follows:

 (5.9)

* Insert the BBA parameters, number of bats (*Nbats*), *α*, *β*, *γ*, *ε*, λmin and λmax.
* Insert the initial values of the loudness (*Li*) and the initial pulse emission rate (*ri0*). In addition, initialize the bats velocity and frequency.
* Insert the candidate buses.
* Create a search space with dimensions (*Nbats* x *Ncan*), where *Ncan* is the number of candidate buses.

***Step 2: Provide bat first position***

Each bat is positioned randomly within the reasonable range of each control variable in a search space with one bat at each position in the length of randomly distributed values.

***Step 3: Initial evaluation***

For each bat, calculate the initial value of objective function with the candidate buses in the search space as:

 (5.10)

***Step 4: Check the system constraints***

For each bat, check the system constraints in Eqs. (3.2)-(3.12) based on the values of control variables to accept the values of objective functions that correspond to the bats satisfying the system constraints.

***Step 5: Initial global best solution***

For accepted solutions after checking constraints, the initial global best solution (*gbestinit*) can be determined according to the objective function.

***Step 6: Probabilistic transition rule***

For each bat, calculate the velocity vector from Eq. (5.1). Then, each bat changes its position in the range of control variables according to the probabilistic transition rule in Eq. (5.3). The bat positions or control variables are modified as the following steps:

1. Calculate the probability using Eq. (5.3) for each bat.
2. If *Pik+1* < *rand*, the bat position (*xik+1*) is modified to be the complement of the bat position (*xik*). Otherwise, the bat position (*xik+1*) is kept as the position (*xik*).
3. Repeat the previous steps for *bat*-times.
4. Get the final form of the modified control variables.

***Step 7: Search space updating***

For each bat, the control variables can be updated using Eq. (5.5).

***Step 8: Fitness function***

After updating the search space, the objective function can be determined for each bat at iteration *k* as:

 (5.11)

***Step 8: Check the system constraints***

For each bat, check the system constraints in Eqs. (3.2)-(3.12) based on the updated control variables to accept the values of objective functions that correspond to the bats satisfying the system constraints. Therefore, the global best solution (*gbest*) at the current iteration can be obtained.

***Step 9: Global best solution***

For accepted solutions after checking constraints, the global best solution (*gbest*) at iteration *k+*1 can be determined as:

 (5.12)

***Step 10: Control variables, loudness and pulse emission rate updating***

After the optimal solution is obtained at the current iteration, the control variables, loudness and pulse emission rate are updated using Eqs. (5.6)-(5.7).

***Step 11: Check stopping criterion***

The program will be terminated when the maximum iteration is reached or the best solution is obtained.

The flow chart of the BBA to find the optimal solution is shown in Fig. 5.1

**Fig. 5.1 Flow chart of BBA**