Optimal capacitor placement in 34-bus system using Archimedes Optimization Algorithm (AOA) for power loss reduction and voltage profile improvement

Supervisor Dr.Mohamed Mouwafi

المشاركون في المشروع

رقم	الرقم الأكاديمي	الاسم	السكشن
1	161017	محمد ربيع عبدالحميد محمد الشيخ	6
2	170598	محمد رجب عبدالمعطي خفاجة	6
3	170562	محمد الدسوقي احمد حليمه	6
4	161016	محمد خالد احمد سلطان	6
5	170691	محمد فكري محمد عبدالوارث زهران	7
6	170860	مصطفى محمد عبدالغنى عبدالله شاهين	9
7	170666	محمد على السيد عبد الجليل	7
8	160677	محمد عماد الدين عبدالنبى عبدالله	7
9	170861	مصطفى محمد عبدالغني ندا	9
10	160805	مصطفي أحمد صبري احمد العشماوي	8
11	140183	اسلام جمال حسين الاغا	3

Table of contents

1. Introduction

1.1. Distribution systems optimization

Shunt capacitors and DGs installation on radial distribution feeders is commonly used to improve power flow control, system stability enhancement and power factor correction, managing voltage profile, and reducing active power and energy loss in distribution network.

1.1.1. Complexity of Distribution systems optimization problems

Distribution systems optimizations problems are too complex that it is computationally impossible.

To get a sense of the complexity of this type of problems, consider a 34-bus distribution system like that in figure 1 and we are required to place 3 capacitors on those systems and the capacitors sizes must be within the boundaries 500: 1200 KVAR. If we considered bus 1 as the slack bus then for the first capacitor there is 33 possible location and the second there are 32 possible location and 31 possible location for the third. So in total we need to try $33 \times 32 \times 31 = 32736$ possible locations.

If we considered only the integer values of the capacitors sizes (500 , 501,502,……,1198,1199,1200), then for each capacitor there are 700 possible value of its size. So there are $(700 \times 700 \times 700 = 343000000 = 343 \times 10\text{^}6$. So the search space with these assumptions would have $32736 \times 343000000 = 1.1228 \times 10^{13}$ candidate solutions.

We can get the mean time required to run the BFS algorithm in years and multiply it with the number of the candidate solutions to find the number of years of computations required to find the best solution (under the mentioned assumptions). The result is shockingly above 200 years.

If we considered values to 1 decimal number instead of integer sizes of the capacitors (500.1, 500.2,500.3,……, 1199.8,1199.9,1200). Then the result will be above 2000 years and in general the result would be $\mathit{number\ of\ years\ for\ one\ increment} \times \mathit{the\ number\ of\ increments\ between\ each\ two\ integers}$.

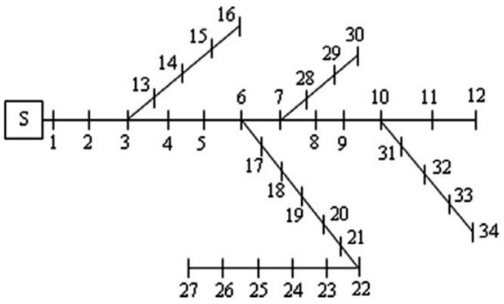


Figure 1

1.2. What is metaheuristics?

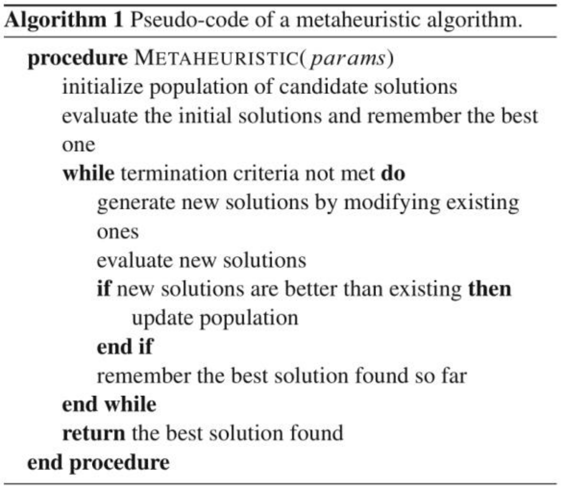
Metaheuristics is a subfield of **stochastic optimization**. They're algorithms used to find solutions to problems which brute-force search algorithms (random search) is computationally impossible to use and you must be able to test these solutions and assess how good it is . Metaheuristics are independent of the problem which means that can implemented to solve different types of problems.

Stochastic optimization is the general class of algorithms and techniques which employ some degree of randomness to find optimal solutions to hard problems. Metaheuristics are the most general kind of these algorithms and applied to a wide range of problems.

Brute-force search means trying each possible solution and choosing the best of them

In our case, we are trying to optimally place DGs and capacitors in a 34-bus network to improve power flow and voltage regulation. It's impossible computational to solve this problem using brute-force search as the space we search is too enormous. We can always check the candidate solution and assess that it improves the power flow and voltage regulation of our network. A metaheuristic algorithm will be suitable for our problem.

1.2.1. Pseudo code of metaheuristics



Algorithm 1 Pseudo-code of a metaheuristic algorithm.

```
procedure METAHEURISTIC(params)
    initialize population of candidate solutions
    evaluate the initial solutions and remember the best one
    while termination criteria not met do
        generate new solutions by modifying existing ones
        evaluate new solutions
        if new solutions are better than existing then
            update population
        end if
        remember the best solution found so far
    end while
    return the best solution found
end procedure
```

1.3. Archimedes Optimization Algorithm (AOA)

Archimedes optimization algorithm (AOA) generally emulates what happens when objects of different weights and volumes are immersed in a fluid.

1.3.1. Archimedes' principle

Archimedes' principle 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."

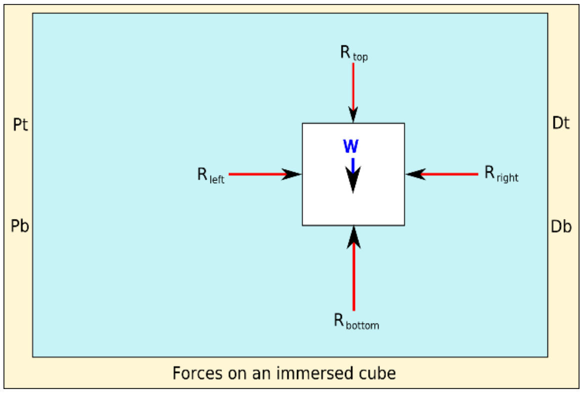


Figure 2

1.3.2. Theory

Objects in a fluid tries to reach equilibrium state. For an object to be in equilibrium state, the buoyant force F_b must equal the object's weight W_0:

$$F_b = W_0$$

$$p_b v_b a_b = p_0 v_0 a_0$$

Where p is the density, v is the volume, and a is the acceleration. The subscripts b and 0 are referring to the fluid and immersed object, respectively.

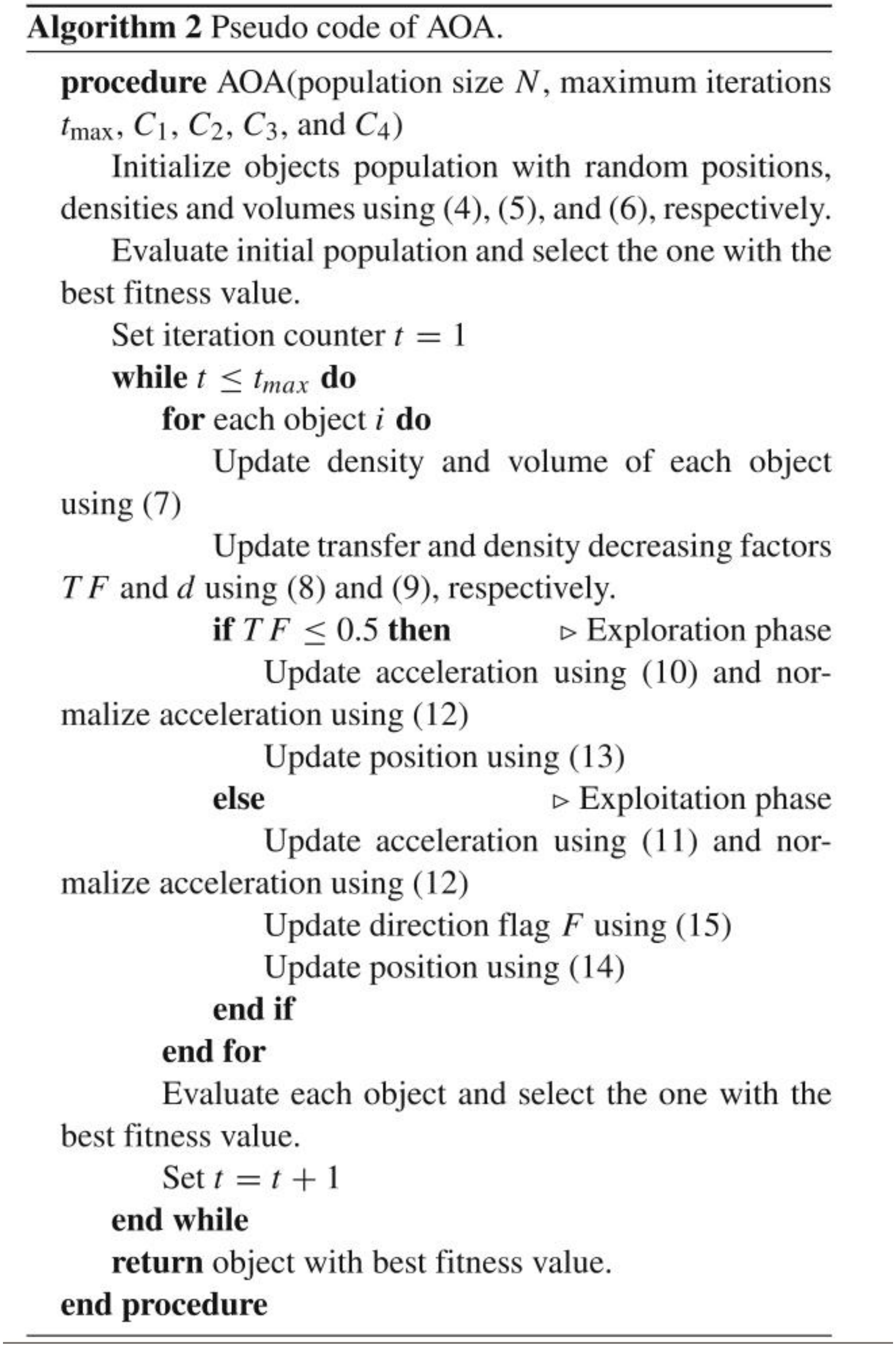
If there is a collision between to objects (r, o) the equilibrium state is reached when

$$p_b v_b a_b = p_r v_r a_r + p_0 v_0 a_0$$

1.3.3. Algorithmic steps

AOA is a population-based algorithm. In the proposed approach, the population individuals are the immersed objects. Like other population-based metaheuristic algorithms, AOA also commences search process with initial population of objects(candidate solutions)with random volumes, densities, and accelerations. At this stage, each object is also initialized with its random position in fluid. After evaluating the fitness of initial population, AOA works in iterations until termination condition meets. In every iteration, AOA updates the density and volume of every object. The acceleration of object is updated based on condition of its collision with any other neighboring object. The updated density, volume, acceleration determines the new position of an object.

1.3.4. Pseudo code of AOA

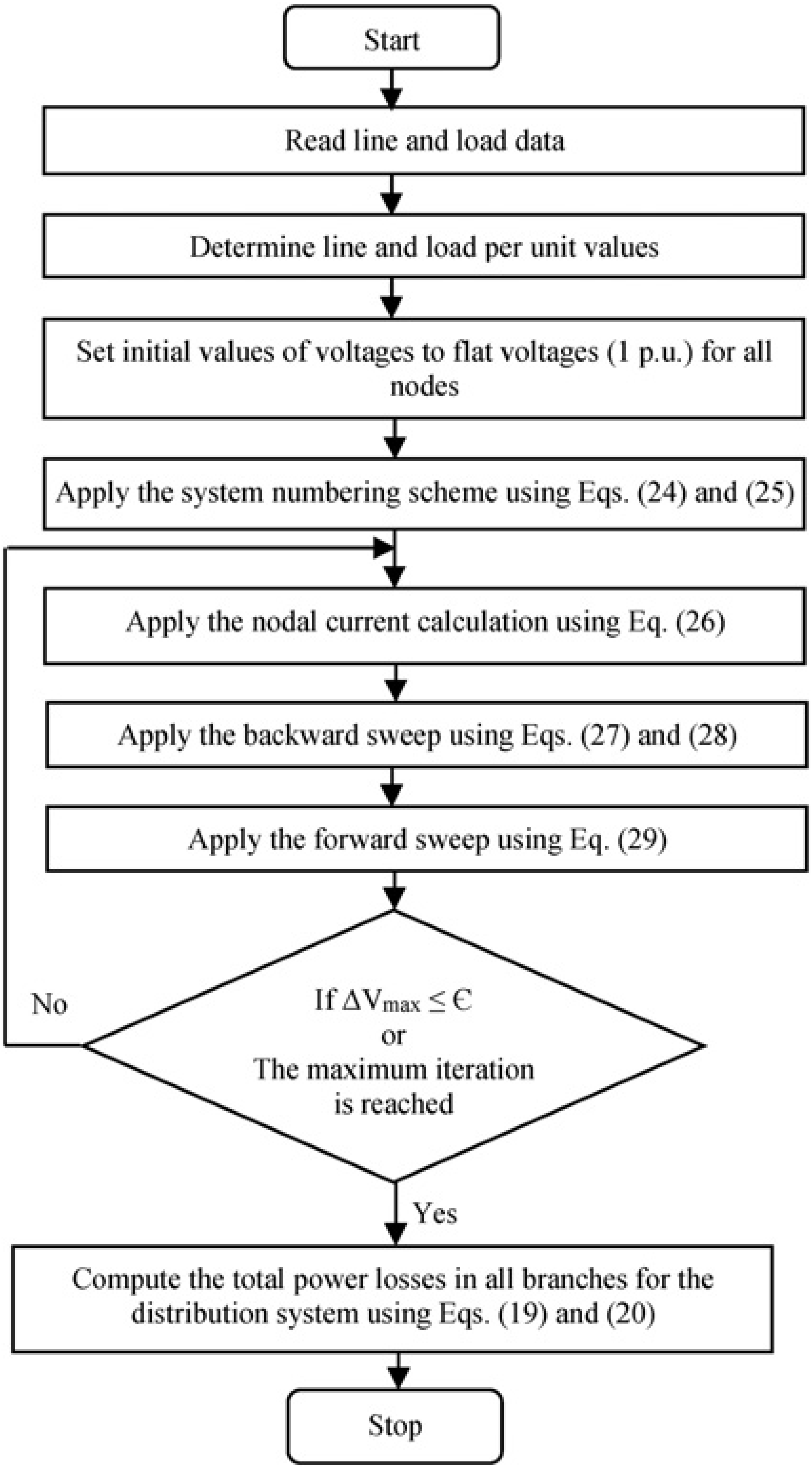


Algorithm 2 Pseudo code of AOA.

```
procedure AOA(population size $N$, maximum iterations $t_{\max}$, $C_1$, $C_2$, $C_3$, and $C_4$)
    Initialize objects population with random positions, densities and volumes using (4), (5), and (6), respectively.
    Evaluate initial population and select the one with the best fitness value.
    Set iteration counter $t = 1$
    while $t \leq t_{max}$ do
        for each object $i$ do
            Update density and volume of each object using (7)
            Update transfer and density decreasing factors $TF$ and $d$ using (8) and (9), respectively.
            if $TF \leq 0.5$ then                    ▷ Exploration phase
                Update acceleration using (10) and normalize acceleration using (12)
                Update position using (13)
            else                                     ▷ Exploitation phase
                Update acceleration using (11) and normalize acceleration using (12)
                Update direction flag $F$ using (15)
                Update position using (14)
            end if
        end for
        Evaluate each object and select the one with the best fitness value.
        Set $t = t + 1$
    end while
    return object with best fitness value.
end procedure
```

2. Problem formulation for power system networks

2.1. Backward/forward sweep (BFS) algorithm


Start
Read line and load data
Determine line and load per unit values
Set initial values of voltages to flat voltages (1 p.u.) for all nodes
Apply the system numbering scheme using Eqs. (24) and (25)
Apply the nodal current calculation using Eq. (26)
Apply the backward sweep using Eqs. (27) and (28)
Apply the forward sweep using Eq. (29)
If $\Delta V_{max} \leq \epsilon$ or The maximum iteration is reached
No
Yes
Compute the total power losses in all branches for the distribution system using Eqs. (19) and (20)
Stop

3. Results

4. conclusion

Type equation here. (1)

Type equation here. (2)