Guidelines for WCCI(CEC)/GECCO 2024 Competition Evolutionary Computation in the Energy Domain: Optimal PV System Allocation

João Soares, Fernando Lezama, José Almeida, Bruno Canizes, Zita Vale
School of engineering (ISEP), Polytechnic of Porto, Porto, Portugal
flz@isep.ipp.pt, jan@isep.ipp.pt, jorga@isep.ipp.pt, bmc@isep.ipp.pt, zav@isep.ipp.pt

Wenlei Bai
Oracle America Inc., Austin, TX, USA
baiwenlei123@gmail.com
Kwang Y. Lee
Baylor University, Waco, TX, USA
Kwang_Y_Lee@baylor.edu

IEEE CIS Task Force on 'Computational Intelligence in the Energy Domain (ci4energy), part of IEEE CIS Intelligent Systems Applications TC (https://www.gecad.isep.ipp.pt/ci4energy/)

IEEE PES Intelligent Systems Subcommittee (ISS), part of IEEE PES Analytic Methods for Power Systems TC (http://sites.ieee.org/pes-iss/)

IEEE PES Working Group on Modern Heuristic Optimization (https://site.ieee.org/psace-mho/)

January 2024

Table of contents

1. Introduction 4

2. General description of the optimal PV allocation problem 5

3. Metaheuristic simulator framework 5

3.A) Encoding of the individual 7

3.B) Fitness function 7

3.C) Scenario overview 8

4. Guidelines for participants 9

4.A) ***main.m -*** Master function/script 9

A.0 and A.1 - # main.m - ***Loading the testbed and case study*** 10

A.2 - #DEparameters.m - Set parameters of the metaheuristic 10

A.3 - #setOtherParameters.m - ***Set other necessary parameters and struct*** 11

A.4 - #setVariablesBounds.m - Set bounds of variables 11

A.5 - #HyDE.m - Algorithm proposed by the competitor 11

A.6 - #Save_results.m - Benchmark results (text-files) 12

4.B) Fitness function evaluation 12

5. Evaluation guidelines 14

6. Material to be submitted to the organizers 14

Appendix: Mathematical formulation 15

Bibliography 17

1. Introduction

Following the success of the previous editions at PES GM (2017,2021), GECCO (since 2018 to 2023), WCCI and CEC (since 2017 to 2023)[1], we are launching a new edition of our algorithm competition at major conferences in the field of computational intelligence. This WCCI/GECCO 2024 competition proposes one track in the energy domain:

Track 1) Optimal PV systems allocation in an unbalanced distribution network. As photovoltaic (PV) penetration into distribution networks continues to grow, the transition from passive to active networks has brought about a new level of complexity in terms of planning and operation. The optimal PV allocation (sizing and location) is challenging because it is mixed-integer non-linear programming with three-phase non-linear unbalanced power flow equations. The objective is to find the optimal PV systems allocation that maximizes the PV penetration within a predefined planning horizon while satisfying operation constraints such as voltage and line limits. The IEEE 37-bus test feeder was used for a case study.

The proposed track (testbed) is developed under the same framework as the past competitions with few adjustments.

Important: The use of the software platform is prohibited for purposes other than competition without a prior warning to the organizing team.

Tip: The most important part for a quick participation in this competition is section 4 of this guideline, i.e., if you want to just implement your heuristic and treat the problem as pure black box item. Former sections (2-3) are introduction and explanation of the problem being optimized.

[1] Check former competitions in http://www.gecad.isep.ipp.pt/ERM-Competitions

2. General description of the optimal PV allocation problem

Figure 1 illustrates an overview of high PV system penetration into an unbalanced distribution network. It indicates that there are MW-scale, commercial, and residential types of PV systems, and the power flow becomes bidirectional after large PV penetration. Bidirectional power flow certainly increases the complexity for system operators, but what's more concerning is the voltage issue introduced by PV. During the heavy loading period, PV penetration normally helps improve the voltage profile because voltage under heavy load is near or even lower than low voltage limits. By penetrating PV, the voltage will be boosted to the acceptable range. Yet, during light loading and high PV penetration time, such as noon time, voltages are likely to be boosted above the high limit, which causes damage to the system. Therefore, the PV allocation problem is to find the optimal location and sizing of PV systems to optimize objectives such as maximizing PV power injection for all node voltages and subjected to certain equality and inequality constraints.

In this testbed, we assume that the load condition and PV system output are deterministic and known. Summarizing, the novelty of the proposed track can be described as follows:

- optimal PV systems allocation planning on an unbalanced three-phase distribution network to maximize the PV system penetration and yet satisfy security constraints.
- the incorporation of a power flow algorithm, fixed-point iterative method developed by EPRI's OpenDSS [13], to solve unbalanced three-phase continuous power flow in a period efficiently thanks to its fast convergence property.
- **participants will implement solution methods based on modern metaheuristic optimization to deal with the computational burden of the consideration of diverse possible scenarios of uncertain parameters and the large number of variables considered**.
- As a result, it is interesting to analyze the impact of various PV system models over a set of case studies using realistic data in distribution network.

Figure 1 illustrates an overview of high PV system penetration into an unbalanced distribution network. The reader can refer to the appendix section for more details of the formulation of optimal PV systems allocation.

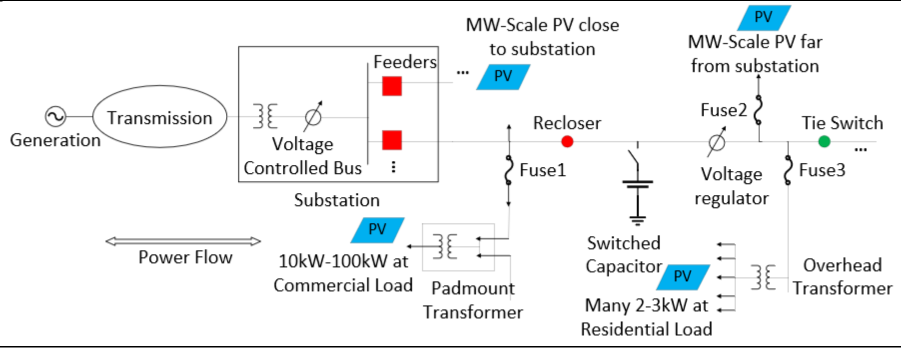


Figure 1 Distribution network with high PV penetration

3. Metaheuristic simulator framework

In this competition, the method of choice used by the participants to solve the problem must be a metaheuristic-based algorithm (e.g., Differential Evolution, Particle Swarm Optimization, Vortex Search, Hybrid approaches, etc.). The framework adopted in the competition is described in this document and follows the structure presented in **Figure 2.**

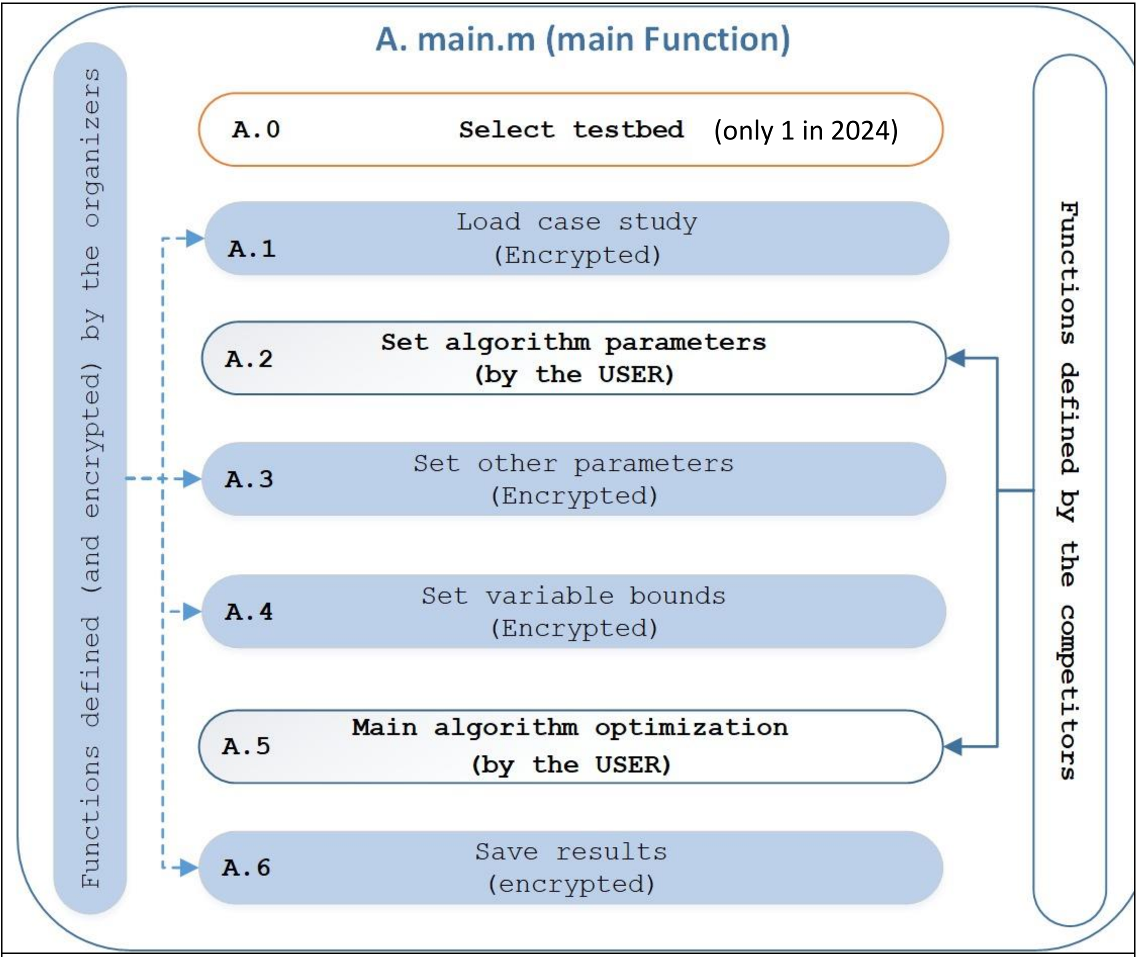

A. main.m (main Function)
Functions defined (and encrypted) by the organizers
A.0 Select testbed (only 1 in 2024)
A.1 Load case study (Encrypted)
A.2 Set algorithm parameters (by the USER)
A.3 Set other parameters (Encrypted)
A.4 Set variable bounds (Encrypted)
A.5 Main algorithm optimization (by the USER)
A.6 Save results (encrypted)
Functions defined by the competitors

Figure 2 General framework of the simulation platform

The simulation platform has been implemented in MATLAB© 2023 64-bit and consists of different scripts with specific targets in the simulation. As shown in **Figure 2**, some scripts correspond to encrypted files provided by the organizers (blue color in the figure). The user only needs to implement two scripts (see **Sect. 4.A.2** and **Sect. 4.A.6**), namely:

i. one script for setting the parameters required by their algorithm (A.2).
ii. a second script for the implementation of their proposed solution method (A.6).

Examples of how to implement these two script functions, and how the organizer's scripts work on the platform, are provided in **Sect. 4.**

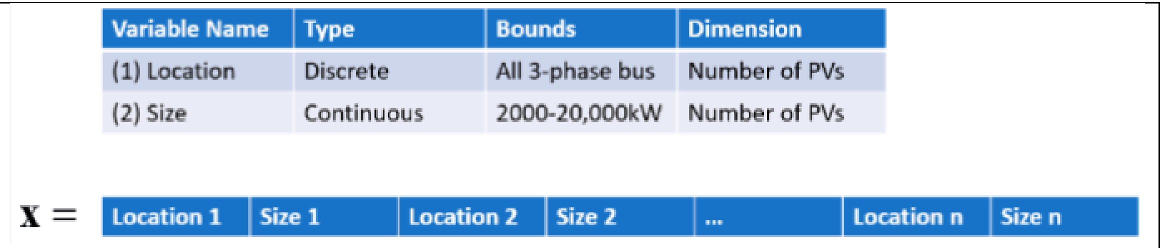
A maximum number of 5,000 function evaluations allowed in the competition. Take into account that one function evaluation corresponds to each time that one solution is evaluated in the fitness (this is not the same as algorithm iterations).

3.A) Encoding of the individual

One fundamental aspect of population-based algorithms is the encoding of the solutions. Depending on the problem, particles/vectors must contain all the information associated with a solution and should be evaluated in a fitness function in order to measure their performance.

The initial solutions should be initialized randomly between the upper and lower bounds of the variables. Heuristics and special tweaks for initial solutions are not accepted.

The solution structure (e.g., an individual in DE, a particle in PSO, or genotype in GA) is a fundamental part of the metaheuristics to represent a given solution. The solution representation for testbed 1 competition follows the vector representation showed in **Figure 3**.



Variable Name	Type	Bounds	Dimension
(1) Location	Discrete	All 3-phase bus	Number of PVs
(2) Size	Continuous	2000-20,000kW	Number of PVs

$\mathbf{x} =$

Location 1	Size 1	Location 2	Size 2	...	Location n	Size n

Figure 3 Solution representation in track 1

The control variables consist of location as discrete variable and size as continuous variable. For example, if there is only 1 PV system, then there are only 2 control variables (1 location + 1 size). Thus, the total control variables equal to the number of PV systems n x 2. Note that this is considered as a planning problem considering the 24-hour period, and since it's not an operational problem, therefore the location and size PV systems will not change for 24 hours operational time, which makes sense. In future work, a longer period of time such as 1 year should be considered in the planning.

3.B) Fitness function

A maximum number of 5,000 function evaluations are allowed in the competition. Notice that the participant can consider the problem as a black box, in which each solution evaluated in the fitness function has a single cost associated to it, as showed in **Figure**

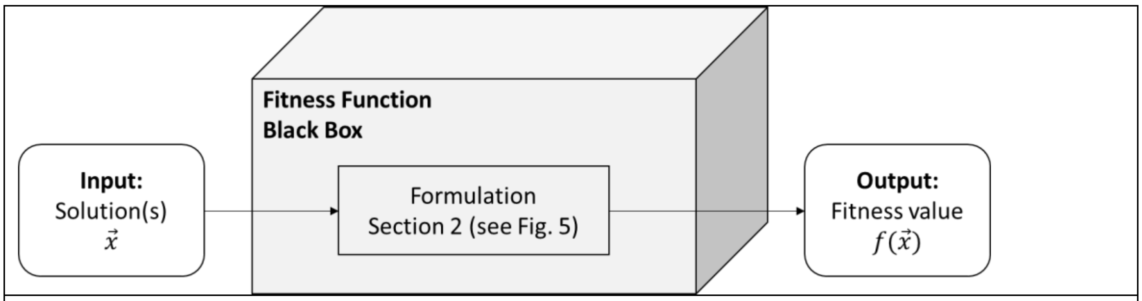


Figure 4 Fitness function as black box.

For participants that want more details on the design of the fitness function for testbed 1, please be referred to the appendix, and take the following explanation into account. Fitness values are obtained via evaluating solution vectors from the objective function (1). The process is straightforward. Initially, the load profile, PV forecast, and network data were loaded, and the objective is to maximize the PV system penetration (minimize the negative penetration).

$$f_1 = \sum_{t=1}^{24} \sum_{i=1}^{n_{PV}} - p_{PVi} \qquad (1)$$

where v_i and p_{PVi} are the bus voltage, and the real power injection at PV node i respectively; n_{pv} is the number of PV systems, n is the total number of nodes. Participants' algorithms will be tested on IEEE-37 bus distribution network system and the average fitness value will be calculated via certain scheme described in section 5.

3.C) Scenario overview

Participants' algorithms will be tested on an IEEE 37 bus system as shown in Fig. 5, the potential buses to interconnect three-phase PV systems are {701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 718, 720, 722, 724, 725, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 740, 741, 742, 744} because they are three-phase bus, and the possible size is from 2,000 – 20,000 kVA. The objective is to find the optimal location and sizing of one PV systems, while verifying voltage variations over one day. Note that there is a total of 117 node voltages for the IEEE-37 bus unbalanced three-phase distribution network, because each bus can have multiple nodes.

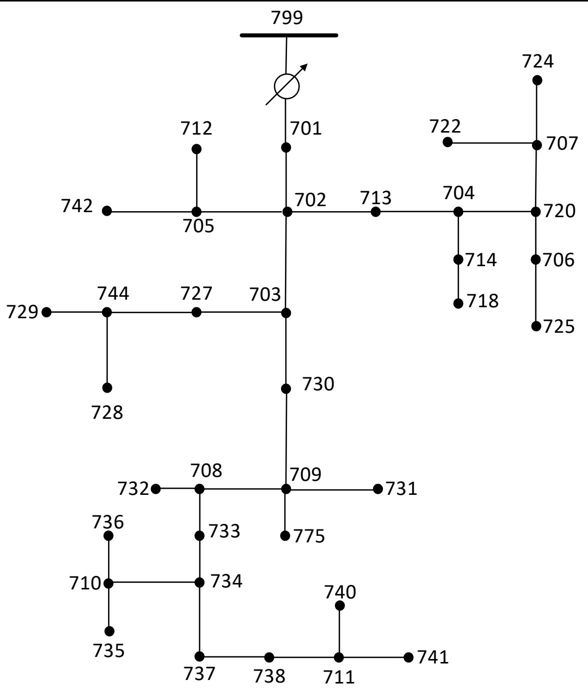
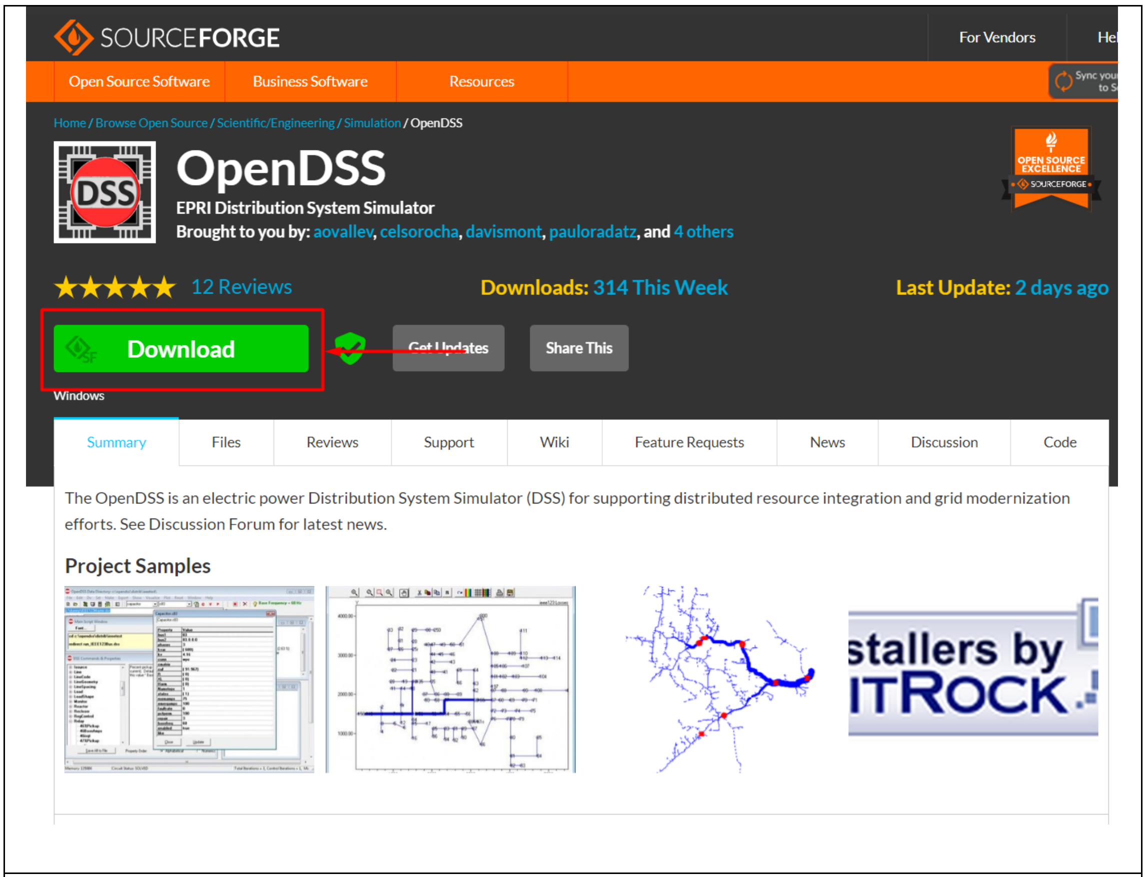


Figure 5 IEEE 37 bus test system.

NOTE: For this case study the installation of OpenDSS[1] is needed (download the file highlighted in Figure 6). The installation process is provided in the competition platform (see README.txt for the download and installation process).

[1]https://sourceforge.net/projects/electricdss/


SOURCEFORGE For Vendors He

Open Source Software Business Software Resources

Home / Browse Open Source / Scientific/Engineering / Simulation / OpenDSS

OpenDSS

EPRI Distribution System Simulator

Brought to you by: aovallev, celsorocha, davismont, pauloradatz, and 4 others

★★★★★ 12 Reviews Downloads: 314 This Week Last Update: 2 days ago

Download Get Updates Share This

Windows

Summary Files Reviews Support Wiki Feature Requests News Discussion Code

The OpenDSS is an electric power Distribution System Simulator (DSS) for supporting distributed resource integration and grid modernization efforts. See Discussion Forum for latest news.

Project Samples

Figure 6 Website for OpenDSS download

4. Guidelines for participants

These instructions include as example the metaheuristic hybrid-adaptive differential evolution (HyDE) [11] implemented and adapted to the present framework (It has been modified by GECAD).

It is important that the participants use the following recommendations and structure to avoid issues in using the supplied datasets and codes.

4.A) *main.m* - Master function/script

# main.m is the main file for the competition. The competitors can modify this main script as needed. Nevertheless, it is worth noting that this main script is ready to use. Participants should only include their functions to perform the optimization of the problem.

main.m

```
……
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% Select testbed
Select_testbed=1; %Only 1 track in 2024
%Testbed 1: Optimal PV system allocation

DB=Select_testbed;
% 1: Case study testbed 1

[caseStudyData, DB_name]=callDatabase(DB);

Select_Algorithm=1;
%1: HyDE algorithm (test algorithm)
%2: Your algorithm
algorithm='HyDE'; %'The participants should include their algorithm here'
DEparameters %Function defined by the participant
```

A.0

A.1

A.2

```
No_solutions=deParameters.I_NP; % %Notice that some algorithms are limited to one individual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%% Set other parameters
otherParameters =setOtherParameters(caseStudyData,No_solutions, Select_testbed);          A.3

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%% Set lower/upper bounds of variables
[lowerBounds,upperBounds] = setVariablesBounds(caseStudyData,otherParameters);            A.4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%% Call the MH for optimization
ResDB=struc([]);
  for iRuns=1:noRuns %Number of trails
    tOpt=tic;
    rand('state',sum(noRuns*100*clock))% ensure stochastic indpt trials

    otherParameters.iRuns=iRuns;                                                          A.5
      switch Select_Algorithm
        case 1
          [ResDB(iRuns).Fit_and_p, ...
          ResDB(iRuns).sol, ...
          ResDB(iRuns).fitVector]=                                                    ...
deopt_simple(deParameters,caseStudyData,otherParameters,lowerB,upperB);
....
   end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                                                                       A.6
    %% Save the results and stats
    Save_results


.....
```

As it can be seen, the main script follows the structure from Figure 2 (**Sect. 3**). Details in the implementation of each part of the code are given next.

A.0 and A.1 - # main.m - Loading the testbed and case study

# main– This is the main framework file where you can select the testbed, which will load the caseStudyData struct (callDatabase.p – encrypted) with all the relevant dataset information depending on the selected testbed. Participants do not need to worry about the content of the case study and loading the files.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
Select_testbed=1;
%Testbed 1: Optimal PV system allocation
%% Load Data base
noRuns=10; %this can be changed but final results should be based on 10 trials
DB=Select_testbed;
% 1: Case study testbed 1
[caseStudyData, DB_name]=callDatabase(DB);
```

A.2 - #DEparameters.m - Set parameters of the metaheuristic

# DEparameters.m file – This function file must be specific to the metaheuristic implemented by the participant. This is just an example using DE to show how participants should implement this function with all the parameters related to their algorithm.

```
deParameters.I_NP= 10; % Size of the population in DE
deParameters.F_weight= 0.3; %Mutation factor
deParameters.F_CR= 0.5; %Recombination constant
deParameters.I_itermax= 500; % number of max iterations/gen
deParameters.I_strategy   = 1; %DE strategy
deParameters.I_bnd_constr = 1; %Using bound constraints
% 1 repair to the lower or upper violated bound
% 2 rand value in the allowed range
% 3 bounce back
```

A.3 - *#setOtherParameters.m* - *Set other necessary parameters and struct*

***# setOtherParameters.m* (encrypted)** – This file is encrypted and should not be changed or modified by the user. It just sets parameters and data needed for the fitness function to work. Please take into account a third parameter is added to the previous framework to consider another track. It is a mandatory function that creates a struct "otherParameters" and should be run as illustrated in main function section:

```
%% Set other parameters
otherParameters =setOtherParameters(caseStudyData,No_solutions, Select_testbed);
```

Participants must pass the "otherParameters" struct as argument to the functions:

```
[lowerBounds,upperBounds]= ...        setVariablesBounds(caseStudyData,otherParameters, Select_testbed);
.....
[ResDB(iRuns).Fit_and_p, ...
            ResDB(iRuns).sol, ...
            ResDB(iRuns).fitVector]= ...              HyDE(deParameters,caseStudyData,otherParameters,lowerB,upperB);
```

A.4 - *#setVariablesBounds.m* - *Set bounds of variables*

***# setVariablesBounds.m* (encrypted)** – This file is encrypted and should not be changed or modified by the user. It just sets the bounds of the problem variables. Please take into account a third parameter is added to the previous framework to consider another track.

```
%% Set lower/upper bounds of variables
[lowerBounds,upperBounds]= ... setVariablesBounds(caseStudyData,otherParameters,Select_testbed);
```

The outputs of this function "[lowerBounds,upperBounds]" – should be used by your algorithm to generate the initial solutions and to validate if the bounds are being respected in each iteration.

The order of the variables in the implemented codes cannot be modify for the proper functioning of the fitness function. The structure of the solution is indicated in **Sect. 3.A** of this document.

A.5 - *#HyDE.m* - *Algorithm proposed by the competitor*

The participants should generate a scrip called ***#MHalgorithm.m*** or similar. This algorithm should replace ***#HyDE.m*** which is provided as example:

```
[ResDB(iRuns).Fit_and_p, ...
            ResDB(iRuns).sol, ...
            ResDB(iRuns).fitVector]= ...              HyDE(deParameters,caseStudyData,otherParameters,lowerB,upperB);
```

Your metaheuristic should receive as input parameters:

1. **deParameters:** struct with the parameters configuration for your algorithm to work (it is generated by the user)
2. **caseStudyData:** struct with the information of the case study
3. **otherParameters:** struct with additional information required by the fitness function
4. **lowerB/upperB:** lower and upper bounds of variables

Your metaheuristic code should return to the main script the following variables:

1. **ResDB(iRuns).fit_and_p:** array of size 1x2 with the best fitness and penalties values
2. **ResDB(iRuns).sol:** vector of size: 1 x noVariables with the best candidate solution found by your algorithm
3. **ResDB(iRuns).fitVector:** array of size: 2xnoIterations with the value of the fitness and penalties over the iterations.

The participants are encouraged to save the results of each trial/run in a struct "**ResDB**", as shown in the example. That will ease the evaluation process by the organizers.

A.6 - #Save_results.m - Benchmark results (text-files)

***#Save_results.m* (encrypted)** – The output is written to text-files using this script. The following tables should be produced:

Table 1. Table_Time: Computing time spent for all optimization trials (X_benchmark_Time_T1.txt)

	timeSpent (s)
Run1	
Run2	
Run3	
…	
Run10	

Table 2. Table_Fitness: Individual benchmark of the fitness in each iteration (X_benchmark_FitnessVector_T1.txt)

	Fitness
Run1	
Run2	
Run3	
…	
Run10	

Table 3. Table_Results: Individual benchmark of each of the trials (X_benchmark_Results_T1.txt)

	OF	Penalties	ConvergenceRate
Run1			
Run2			
Run3			
…			
Run10			

Table 5. Table_TrialStats: Summary statistics or the trials (X_benchmark_Summary_T1.txt)

Ranking Index	Standard deviation	Minimum	Maximum	Variance	Average time	Code
RankingIndex	PstdOF	PminOF	PmaxOF	PvarOF	AvgTime	validationCode

In addition, this function should automatically generate the file "X_Send2Organizers_TX.mat", which should include the best solutions found in each of the trials. That file will be used to double-check the reported results by validating all the solutions contained of the case study. For that reason, it is important that the participants put special care in returning the best solutions from their algorithms and stored in "ResDB.sol" (see Sect. 4.A.6).

To clarify, the "X_send2Organizers_TX.mat" file will include a matrix called *"solutions"* with the solutions stored in "ResDB.sol". The solutions there will be evaluated according to Sect. 5 in order to double check the ranking index of each participant. The lower the ranking index, the better the performance of a participant.

***A number 10 trials should be made.**
***5,000 evaluations per trial for testbed 1.**

4.B) Fitness function evaluation

***# fitnessFun_riskERM.m and #fitnessFun_transEP* (encrypted) –** this is the fitness function to be used by participants and should be called as below. The "fnc" parameter will be assigned automatically to load the corresponding fitness function according to the selected testbed **Sect. 4.A.0**.

```
[S_val, ~]=feval(fnc,FM_pop,caseStudyData,otherParameters);
```

The function receives as input:

1. **fnc**: string with the fitness function m file name: and "fitnessFun_riskERM.m".
2. **FM_pop:** matrix of size $N_{sol} \times D$, in which N_{sol} (rows) represents the number of individuals/solutions in an array, and D (columns) represents the dimension (i.e., number of variables) of the optimization problem. This variable should be encoded in the metaheuristic algorithm proposed by participants (e.g., ***#MHalgorithm.m*, Sect. 4.A.5**). Only 1 individual is also possible (one row).
3. **caseStudyData:** struct with data of the case study with the network data as loaded by callDatabase function (i.e., ***#callDatabase.m,*** **Sect. 4.A.1**).
4. **otherParameters:** Struct with additional information as loaded by ***#setOtherParameters.m*** **Sect. 4.A.3**).

The function returns as output:

1. **S_val**: Matrix of size N_{sol} represents the number of individuals. This matrix includes the fitness values including penalties of the solutions.

The ***#fitnessFun*** evaluates all the population (individuals) at once. A maximum number of 5,000 function evaluations is set for this competition. The table below helps the participant to have an idea of the maximum number of iterations and population It can set without surpassing the allowed number of evals. So, take it account when designing your algorithm:

Table 7. Algorithm population/iterations limits

Size of the population	Max. iterations Track 1
1	5,000
5	1,000
20	250
50	100
100	50
1000	5

5. Evaluation guidelines

A ranking index will be calculated using the 20 final solutions (one for each trial) provided by each participant. With these solutions, the organizers will calculate the ranking index (RI_{user}) for each participant a based on the average fitness of solutions in this competition. The values $RI_{user(a)\_T1}$ will be normalized and a final ranking will be produced.

$$RI_{user(a)\_T1} = \frac{1}{N_{trials}} \cdot \left[\sum_{i=1}^{N_{trials}} \left(Fit_a(\vec{X}_{i_{T1}}) \right) \right] \tag{2}$$

where $Fit_a(\vec{X}_i\_T1)$ is a function that returns the fitness value of the solution found in trial i (i.e., $\vec{X}_i$) by participant a (See **Sect. 3.B**).

Therefore, the winner of the competition will be the one that gets the minimum value of RI_{user} (minimization problems). The participants must consider this criterion while selecting the best search strategy in their algorithms.

6. Material to be submitted to the organizers

For the validation of the results, the 4 benchmark text files and the "send2Organizers_TX.mat" file produced by ***# Save_results.m*** (see **Sect. 4.A.6**) should be submitted to the organizers. The implementation codes of each algorithm entering the competition must also be submitted along with final results for full consideration in the evaluation. The submitted codes will be used for further tests, which are intended to crosscheck the submitted results (Note: this evaluation could consider the modification of the case study and number and the encoding of variables, so that algorithms should be designed generally enough to handle different case studies of the same problem). The submitted codes will be in the public domain and no intellectual property claims should be made.

Each participant is kindly requested to put the text files corresponding to final results, as well as the implementation files (codes), obtained by using a specific optimizer, into a zipped folder named:

WCCI_GECCO2024_*AlgorithmName*_*ParticipantName*.zip
(e.g., WCCI_GECCO2024_*DE*_*Lezama*.zip).

The zipped folder must be summited to jan@isep.ipp.pt; flz@isep.ipp.pt, jorga@isep.ipp.pt

by 21th June 2024 (anywhere on Earth)

Appendix: Mathematical formulation

The optimal PV system allocation's mathematical formulation is presented as follows [14].

$$\min \ f(\mathbf{u}) \tag{3}$$

$$g(\mathbf{u},\mathbf{x},\mathbf{y}) \le 0 \tag{4}$$

$$h(\mathbf{u},\mathbf{x},\mathbf{y}) = 0 \tag{5}$$

where **u** is the control variable including PV locations and size, **x** is the state variable/dependent variable including voltages and angles at each bus, **y** is the known network parameters such as network resistance, impendence, device rating, etc. $f(\cdot)$ is the objective function presented as follows.

$$f_1 = \sum_{t=1}^{24}\sum_{i=1}^{n_{PV}} - p_{PVi} \tag{6}$$

where v_i and p_{PVi} are the bus voltage, and the real power injection at PV node *i* respectively; n_{pv} is the number of PV systems, *n* is the total number of nodes, $h(\cdot)$ is the equality constraints which is the power balance equation at each node represented as:

$$\begin{gathered} P_i = V_i \sum_{j=1}^{N} V_j Y_{ij} \cos(\delta_i - \delta_j - \theta_{ij}) \\ Q_i = V_i \sum_{j=1}^{N} V_j Y_{ij} \sin(\delta_i - \delta_j - \theta_{ij}) \quad \forall i, \forall j \end{gathered} \tag{7}$$

where P_i, Q_i are the real and reactive power at each node *i*, note unlike transmission network, each bus in distribution network will have to include multiple nodes in the model to reflect the possible unbalanced power flow. In other words, the size of equations increases significantly. V_i, V_j, δ_i and δ_j, are voltage magnitude and angle at node *i*, *j*; Y_{ij} and ϑ_{ij} are the Y-bus admittance matrix elements between node *i* and *j*. Equation (7) is a list of highly nonlinear equations. $g(\cdot)$ is the inequality constraints which include line flow limit, voltage limit, PV active power injection limit, transformer tap limit, and generator output limit.

$$p_{Gi,\min} \le p_{Gi} \le p_{Gi,\max} \tag{8}$$

$$t_{i,\min} \le t_i \le t_{i,\max} \tag{9}$$

$$v_{i,\min} \le v_i \le v_{i,\max} \tag{10}$$

$$s_{Li} \le s_{Li,\max} \tag{11}$$

$$p_{pv,\min} \le p_{pvi} \le p_{pv,\max} \tag{12}$$

where (8) is the constraint for *i*th generator; (9) represents the *i*th transformer tapping limit; (10) is the voltage limit at the *i*th node; (11) is the complex power flow limit at the *i*th line, and (12) represents the PV system capacity limits, which are 2,000-20,000 kVA in this study.

Note that (12) is the control variable constraint, which is enforced within control variable feasible domain. The rest of the equations are related with dependent variables and only violations from (10) are penalized to objective functions if existing because (8), (9) and (11) are enforced when performing power flow calculation. The objective function then becomes (13):

$$f_{obj} = f + pen \times \sum_{i=1}^{n} \begin{cases} (v_i - v_{i,\max})^2 & v_i > v_{i,\max} \\ (v_{i,\min} - v_i)^2 & v_i < v_{i,\min} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The following describes the fixed-point iterative method for solving unbalanced distribution load flow (DLF) shown in Figure 7. The process is straightforward. We first use an initial guess of $\mathbf{v}_0$ (1.0 p.u.) to calculate $\mathbf{i}_{inj}$ and then calculate the voltage iteratively until the algorithm converges (the difference in current and previous **v** is within a predefined threshold). Figure 8 is the flow chart of the iterative method.

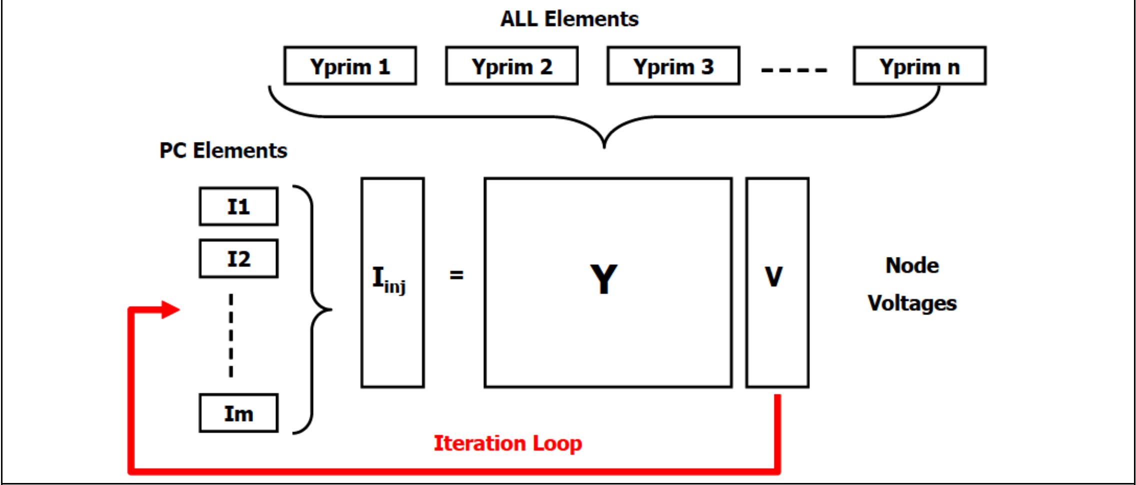


Figure 7 Fixed-point iterative method

$$\mathbf{v}_{n+1} = Y_{system}^{-1} \times \mathbf{i}_{inj}(\mathbf{v}_n), \quad n = 0,1,2,\ldots,n \qquad (14)$$

where $\mathbf{i}_{inj}(\mathbf{v})$ is the compensation or injection currents vector from power conversion elements (load, generator, Vsource, Isource, storage, etc.) in the circuit, which may be nonlinear, not constant, and node voltages dependent; **v** is the node voltage vector; Y_{system} is network admittance matrix composed of all elements' primitive matrices Y_{prim} as shown in Figure 9. *n* is the number of iterations. Note that most entries in ***I*** are zero, but for DERs and non-linear voltage-dependent loads (such as constant power and constant current loads), the corresponding entries in ***I*** are non-zero. The advantages of this DLF are: 1) Y_{system} remains constant if no network topology change. In other words, the matrix inversion only needs to be done once during the iteration, which saves a lot of computation time. 2) The DLF can take very unbalanced three-phase networks and converge successfully. 3) It's also friendly to parallel sources (mesh topology) networks as opposed to the radial network topology, which is required by the commonly used backward/forward sweep method [12]. 4) it provides the option to run a long time series continuous load flow efficiently thanks to the fast-solving feature.

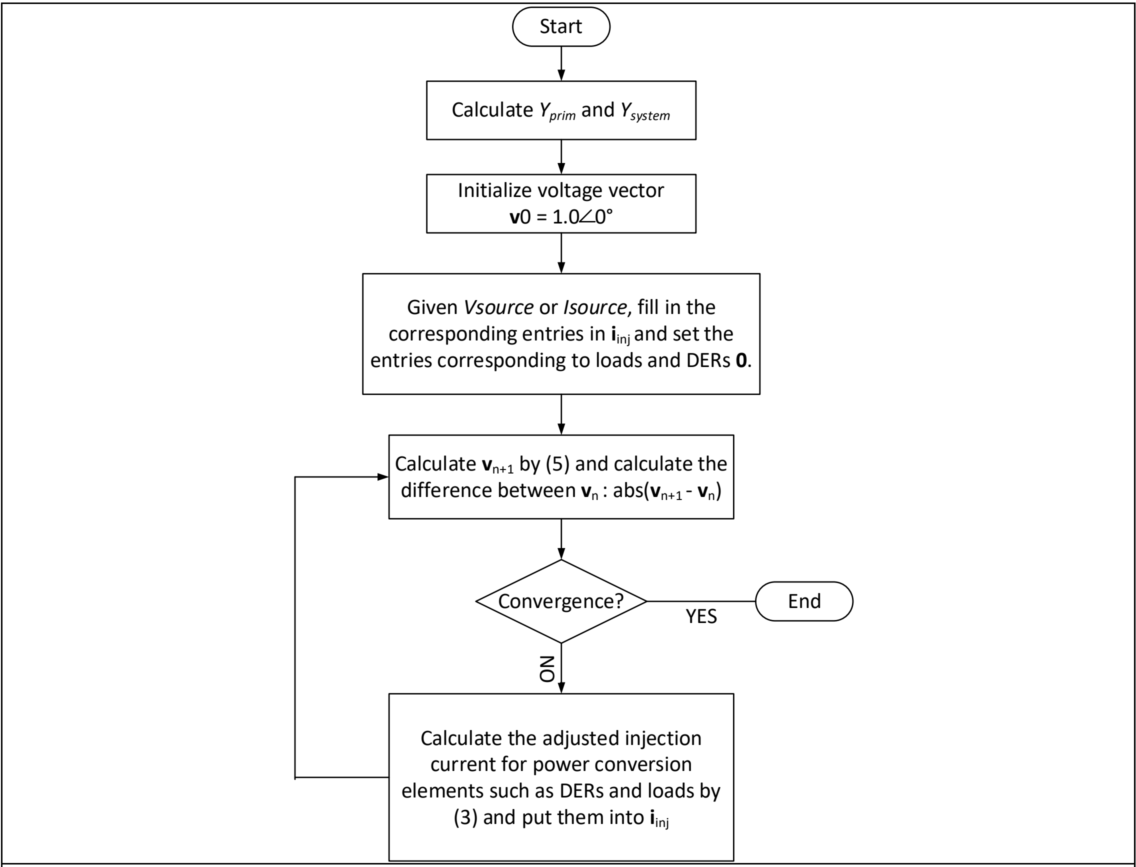

Start
Calculate Y_{prim} and Y_{system}
Initialize voltage vector $\mathbf{v}0 = 1.0\angle 0°$
Given *Vsource* or *Isource*, fill in the corresponding entries in $\mathbf{i}_{inj}$ and set the entries corresponding to loads and DERs $\mathbf{0}$.
Calculate $\mathbf{v}_{n+1}$ by (5) and calculate the difference between $\mathbf{v}_n$: abs($\mathbf{v}_{n+1}$ - $\mathbf{v}_n$)
Convergence?
YES
End
NO
Calculate the adjusted injection current for power conversion elements such as DERs and loads by (3) and put them into $\mathbf{i}_{inj}$

Figure 8 Flow chart of the iterative method

Bibliography

[1] G. Mavromatidis, K. Orehounig, and J. Carmeliet, "A review of uncertainty characterisation approaches for the optimal design of distributed energy systems," *Renew. Sustain. Energy Rev.*, vol. 88, pp. 258–277, May 2018, doi: 10.1016/j.rser.2018.02.021.

[2] M. Tavakoli, F. Shokridehaki, M. Funsho Akorede, M. Marzband, I. Vechiu, and E. Pouresmaeil, "CVaR-based energy management scheme for optimal resilience and operational cost in commercial building microgrids," *Int. J. Electr. Power Energy Syst.*, vol. 100, pp. 1–9, Sep. 2018, doi: 10.1016/j.ijepes.2018.02.022.

[3] F. Samadi Gazijahani and J. Salehi, "Optimal Bilevel Model for Stochastic Risk-Based Planning of Microgrids Under Uncertainty," *IEEE Trans. Ind. Inform.*, vol. 14, no. 7, pp. 3054–3064, Jul. 2018, doi: 10.1109/TII.2017.2769656.

[4] F. Lezama, J. Soares, P. Hernandez-Leal, M. Kaisers, T. Pinto, and Z. Vale, "Local Energy Markets: Paving the Path Toward Fully Transactive Energy Systems," *IEEE Trans. Power Syst.*, vol. 34, no. 5, pp. 4081–4088, Sep. 2018, doi: 10.1109/TPWRS.2018.2833959.

[5] J. Soares, B. Canizes, M. A. Fotouhi Gazvhini, Z. Vale, and G. K. Venayagamoorthy, "Two-stage Stochastic Model using Benders' Decomposition for Large-scale Energy Resources Management in Smart grids," *IEEE Trans. Ind. Appl.*, pp. 1–1, 2017, doi: 10.1109/TIA.2017.2723339.

[6] M. Rahmani, R. Romero, and M. J. Rider, "Strategies to Reduce the Number of Variables and the Combinatorial Search Space of the Multistage Transmission Expansion Planning Problem," *IEEE Trans. Power Syst.*, vol. 28, no. 3, pp. 2164–2173, Aug. 2013, doi: 10.1109/TPWRS.2012.2223241.

[7] J. Almeida, J. Soares, F. Lezama, and Z. Vale, "Robust Energy Resource Management incorporating Risk Analysis using Conditional Value-at-Risk," *IEEE Access*, pp. 1–1, 2022, doi: 10.1109/ACCESS.2022.3147501.

[8] M. Esmaeeli, A. Kazemi, H. Shayanfar, G. Chicco, and P. Siano, “Risk-based planning of the distribution network structure considering uncertainties in demand and cost of energy,” *Energy*, vol. 119, pp. 578–587, Jan. 2017, doi: 10.1016/j.energy.2016.11.021.

[9] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, “MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education,” *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011, doi: 10.1109/TPWRS.2010.2051168.

[10] B. Canizes, J. Soares, Z. Vale, and J. Corchado, “Optimal Distribution Grid Operation Using DLMP-Based Pricing for Electric Vehicle Charging Infrastructure in a Smart City,” *Energies*, vol. 12, no. 4, p. 686, Feb. 2019, doi: 10.3390/en12040686.

[11] F. Lezama, J. Soares, R. Faia, T. Pinto, and Z. Vale, “A New Hybrid-Adaptive Differential Evolution for a Smart Grid Application Under Uncertainty,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Jul. 2018, pp. 1–8. doi: 10.1109/CEC.2018.8477808.

[12] D. Gao, E. Muljadi, T. Tian, and M. Miller, "Software comparison for renewable energy deployment in a distribution network," *Technical Report, the National Renewable Energy Laboratory (NREL)*, 2017.

[13] R. C., Dugan, and T. E. McDermott, "An open-source platform for collaborating on smart grid research," *2011 IEEE PES General Meeting*, Detroit, Michigan, U.S, July 24–28, 2011. doi: 10.1109/PES.2011.6039829.

[14] W. Bai, W. Zhang, R. Allmendinger, I. Enyekwe, and K. Y. Lee, "A Comparative Study of Optimal PV Allocation in a Distribution Network Using Evolutionary Algorithms," *Energies*, vol. 17, no. 2, 2024, pp. 511 - 530, doi: https://doi.org/10.3390/en17020511.