Guidelines for WCCI(CEC)/GECCO 2022 Competition Evolutionary Computation in the Energy Domain: Risk-based Energy Scheduling

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

IEEE CIS Task Force on 'Computational Intelligence in the Energy Domain (ci4energy), part of IEEE CIS Intelligent Systems Applications TC (http://ci4energy.uni-paderborn.de/committee/)

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 2022

Table of contents

1. Introduction

Following the success of the previous editions at PES GM (2017,2021), GECCO (since 2018 to 2021), WCCI and CEC (since 2017 to 2021)[1], we are launching a new edition of our algorithm competition at major conferences in the field of computational intelligence. This WCCI(CEC)/GECCO 2022 competition proposes one track in the energy domain:

Track 1) Risk-based optimization of aggregators' day-ahead energy resource management (ERM) considering uncertainty associated with the high penetration of distributed energy resources (DER). This test bed is constructed under the same framework of past competitions (therefore, former competitors can adapt their algorithms to this new track), representing a centralized day-ahead ERM in a smart grid located in a 13-bus distribution network using a 15-scenario case study with 3 of these scenarios considered as extreme events (high impact, and low probability). A conditional value-at-risk (CVaR) mechanism is used to measure the risk associated with the extreme events for a confidence level (α) of 95%. We also add some restrictions to the initialization of the initial solution and the allowed repairs and tweak-heuristics.

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

Tip: The most important part for a quick participation in this competition is section 4 of this guidelines, 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 risk-based energy scheduling

Renewable generation, either solar or wind types, is usually weather-dependent. Therefore, their management has always associated an uncertain variability that jeopardizes the operation of the entire energy chain [1]. In this context, an energy management system should be both reliable and resilient, operating as expected in the case of events with a high probability of occurrence but a reasonably small effect (e.g., reliable under different faults in the power system), and prepared to deal with events that despite a low probability, have a considerable influence in the outcome of the operation (e.g., resilient to extreme events such as hurricanes, thunderstorm, etc.) [2]. Recently, energy management models are incorporating risk-based methods to handle the uncertainty associated with variable parameters [3]. In this track, we hypothesize that the incorporation of risk parameters into the formulation results in solutions that, despite their higher cost, protect the aggregator (and, in consequence, the end-user) against possible scenarios that might endanger the entire system.

In this track, we model the energy resource management (ERM) problem under uncertainty of renewable generation, load consumption, market prices, and electric vehicles trips, similar to the works presented in [4], [5]. The stochastic behavior of these parameters is considered through various scenarios that have associated a probability of occurrence. The novelty of this new testbed lies in the incorporation of risk strategies into the formulation, allowing the aggregator to plan its operation considering different degrees of risk associated with diverse scenarios (risk-neutral and risk-averse considerations). Summarizing, the new proposed track can be described as follows:

- day-ahead ERM model considering uncertainties of renewable generation, load consumption, market prices, and electric vehicles trips.
- the incorporation of risk analysis strategies using VaR and $CVaR$ measures to deal with the uncertainty of parameters and get solutions that protect the aggregator against extreme scenarios.
- **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 results, it is interesting to analyze the impact of VaR and $CVaR$ measures over a set of case studies using realistic data in power and energy systems.

Figure 1 illustrates the proposed track. The reader can be referred to the appendix section for more details of the formulation.

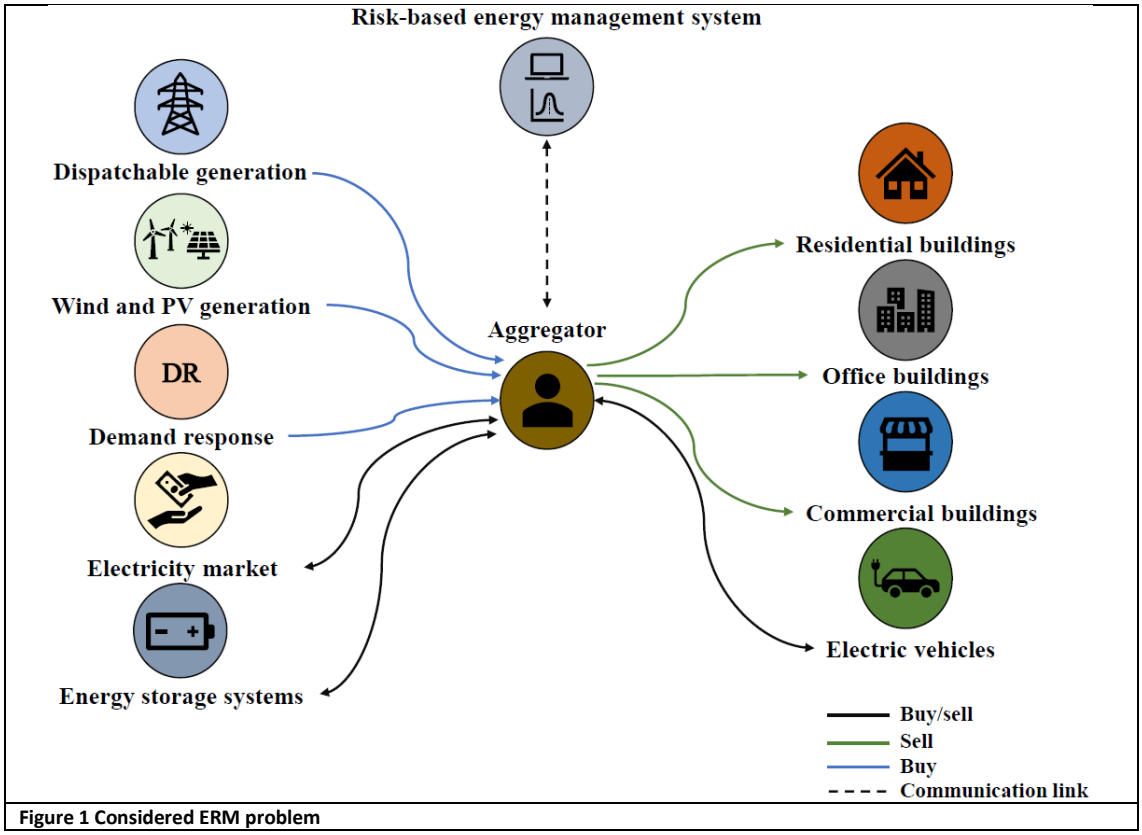

Figure 1 Considered ERM problem

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.

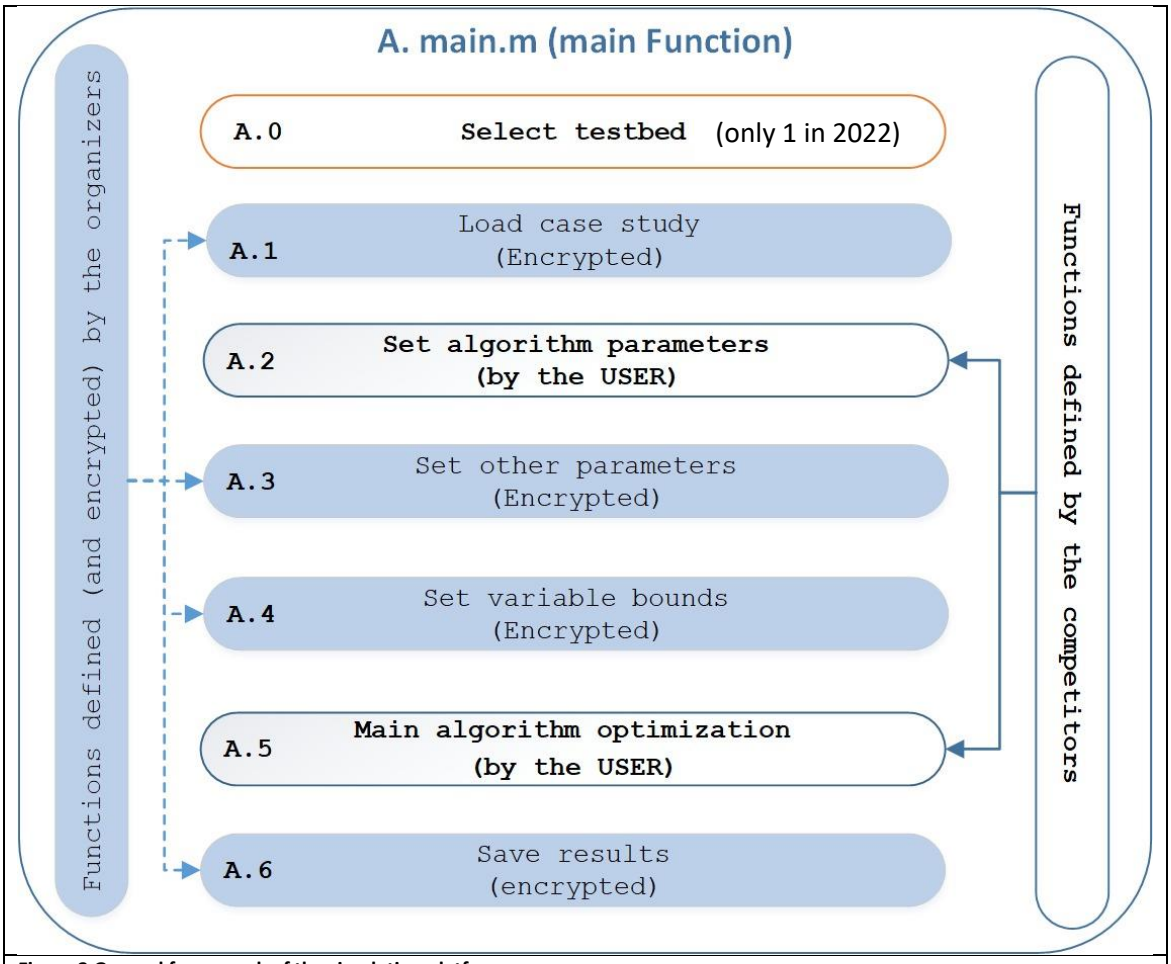

Figure 2 General framework of the simulation platform

The simulation platform has been implemented in MATLAB© 2018 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 is 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 are not allowed.

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 2022 competition follows the vector representation showed in **Figure 3**.

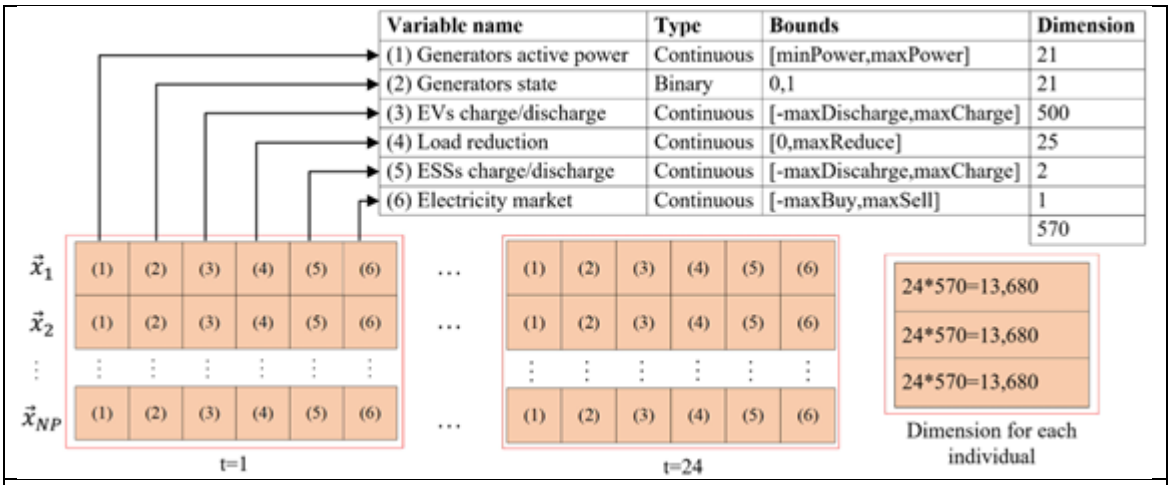

Figure 3 Solution representation in track 1

The metaheuristic's initial solutions are created randomly between the maximum and minimum values set for each variable. Each solution is composed by a group of sequentially repeated variables per each of the 24 periods (t=1, t=2, ..., t=24). The variables in each group are defined by the technologies present in the distribution network. All variables are of continuous type varying according to the specified bounds, except for the state of the generators, which is represented by a binary state (0 indicates that the generator is not connected to the grid, and 1 that is connected).

The first group of variables represents the active power generation (including renewable production). The photovoltaic and wind generation cannot be controlled due to solar and wind factors, respectively. Even though the variables associated with this type of generation are considered in the solution vector, it is essential to note that these variables have an unchangeable value that depends on the uncertainty scenario, so the bounds of these variables are limited to the maximum production value. EV and ESS solutions are set to vary from the maximum active power discharge to the maximum active power charge, assuming the discharging as a negative variable (generation) and the charging as a positive variable (consumption). The demand response (DR) here is only assumed as a load reduction (despite in some other cases a load increase program could be considered), so the DR variable only varies from 0 to the maximum active power reduction. When it comes to market variables, it is assumed that a negative value is the power bought in the market, and a positive variable value is the power sold.

The considered ERM has 13,680 variables per individual given by 570 variables per period, with 21 variables composing the generators' active power, another 21 binary variables representing the generators' state. A total of 500 EVs were also considered with 25 load types, 2 ESSs, and 1 market.

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 4**.

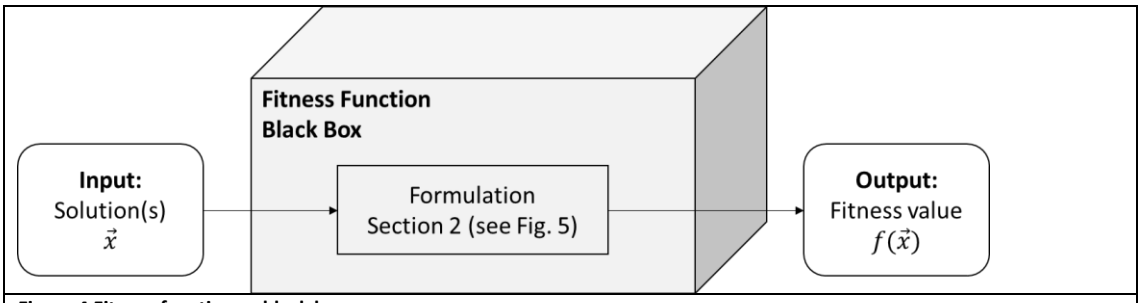

Figure 4 Fitness function as black box.

For participants that want more details on the design of the fitness function, please be referred to [6], and take the following explanation into account. **Figure 5** shows the fitness function that the chosen metaheuristic evaluates for cost minimization. Initially, the database with the formulated scenarios is passed as an argument to the function, and the value of the variable that controls risk aversion is also initialized. Next, each scenario is evaluated according to the equations in the appendix section. This evaluation is done to obtain the cost of each scenario, which is saved to calculate the expected cost.

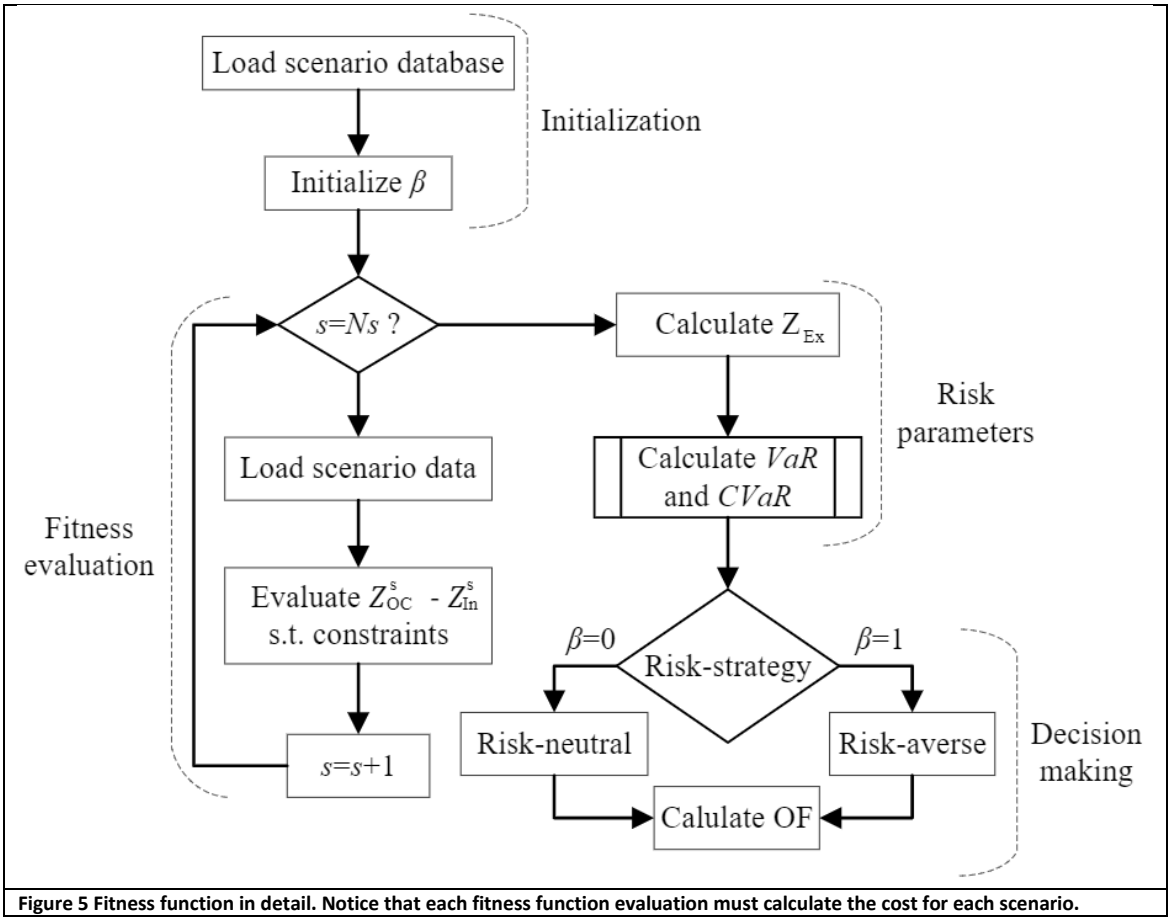

Figure 5 Fitness function in detail. Notice that each fitness function evaluation must calculate the cost for each scenario.

The expected cost, the cost of each scenario, and the probabilities of each scenario are used to calculate the VaR_α and $CVaR_\alpha$ values according to the formulation in [7]. After the parameters that measure risk have been calculated, the aggregator enters a decision process according to the risk aversion factor. That is, through the value of the OF, the aggregator chooses the best strategy.

The metaheuristic does this evaluation in order to minimize the value of the OF in a given number of iterations, so when $\beta = 0$, the metaheuristic will only minimize the expected cost. Still, when the $\beta = 1$, the metaheuristic tries to minimize the expected cost as well as the $CVaR_\alpha$ [3].

A risk-neutral strategy considered for the day-ahead ERM considers the uncertain behavior of an aggregator's technologies such as renewable generation, load consumption, market prices, EV consumption behavior. In this case, the stochastic behavior of these parameters is considered in the approach used through various scenarios with an associated probability of occurrence.

When the risk is not considered, this aggregator's scheduling is formulated based on the expected scenario. The cost and the value of the objective function when a risk aversion strategy is not considered is given by the expected cost, and its formulation is given by:

$$Z_s^{\text{tot}} = Z_s^{\text{OC}} - Z_s^{\text{In}} + P_s$$	*(1)*
$$Z^{\text{Ex}} = \sum_{s=1}^{N_s} (\rho_s * Z_s^{\text{tot}})$$	*(2)*

where Z_s^{tot} is the total objective function (OF) value of each scenario s given by the difference between operational costs in each scenario (Z_s^{OC}) and the income in each scenario (Z_s^{In}) and the penalty for bound violations of any variable (P_s). The expected OF cost is represented by Z^{Ex}, and ρ_s is the probability of the respective scenario.

A risk-aversion strategy considers the risk associated with the uncertainty of the previously mentioned technologies. $CVaR_\alpha$ is an additional cost that is added to Z^{Ex} in $(1-\rho)$% of the scenarios with the highest costs. After calculating the value of VaR_α, the $CVaR_\alpha$ is calculated using the following formulation [7]:

$$CVaR_\alpha(Z_s^{\text{tot}}) = VaR_\alpha(Z_s^{\text{tot}}) + \frac{1}{1-\alpha}\sum_{s=1}^{N_s}(\rho_s * \varphi)$$	*(3)*

where:

$$\varphi = \begin{cases} Z_s^{\text{tot}} - Z^{\text{Ex}} - VaR_\alpha(Z_s^{\text{tot}}) & \text{if } Z_s^{\text{tot}} \geq Z^{\text{Ex}} + VaR_\alpha(z_s^{\text{tot}}) \\ 0 \text{ otherwise} \end{cases}$$	*(4)*
$$VaR_\alpha(Z_s^{\text{tot}}) = z - score(\alpha) * std(z_s^{\text{tot}})$$	

φ is a parameter associated with the cost in the worst scenarios; that is, when the cost of each scenario s exceeds the expected cost with the addition of the VaR_α value. If the opposite occurs, $\varphi = 0$. z-score is calculated using **norminv()** function in MATLAB with $\alpha = 95\%$.

Taking this parameter into account, the fitness value (and OF) of the scheduling problem varies according to the level of risk aversion considered. The model of the fitness value (or OF) in this situation is then given by:

$$OF = Z^{\text{Ex}} + \beta * CVaR_\alpha(Z_s^{\text{tot}})$$	*(5)*

In this situation, the β parameter represents the percentage of aversion to the risk. This parameter can vary between 0 and 1. When $\beta = 0$, the *OF* value is only equal to the expected cost, which is a risk-neutral strategy. On the contrary, $\beta = 1$ means that the strategy has 100% aversion to risk presenting the safest solution when it comes to the worst scenarios.

"In this competition, $\beta = 1$ is set as default". Finally, the equations for operational costs (Z_s^{OC}) and incomes (Z_s^{In}) are provided in the appendix section.

3.C) Scenario overview

A medium voltage (MV), distribution network (DN) of an smart city located in the BISITE laboratory in Salamanca, Spain, was chosen for this case study [8]. This DN features one 30MVA substation in bus 1, 15 DG units (2 wind farms and 13 PV parks), and four 1Mvar capacitor banks. When it comes to consumption, this DN has 25 different loads composed of residential and office buildings and some buildings that provide a service (hospital, fire station, and shopping mall). The smart city in question has seven charging stations allowing EVs to charge their batteries, including four 7.2kW slow charging stations per connection point and 50kW fast charging stations per connection point. The single-line diagram presented in **Figure 6** of the 13-bus 30-kV shows all the technologies listed. This case study considers the high penetration of EVs and renewables.

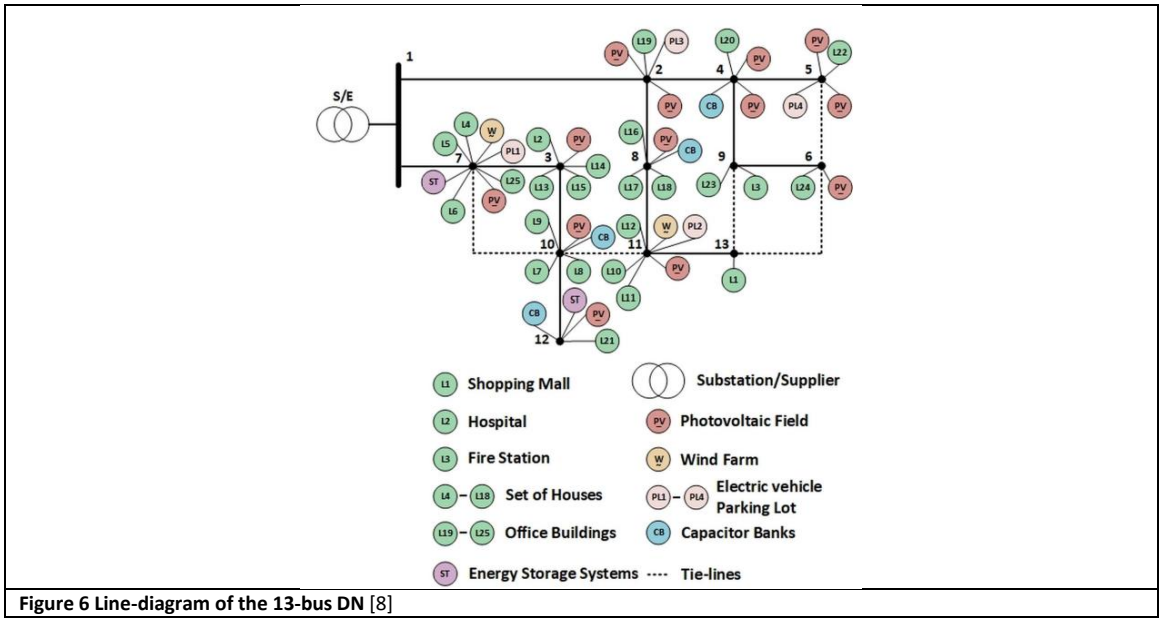

Figure 6 Line-diagram of the 13-bus DN [8]

The aggregator's variable inputs show a lot of variation in the risk scenarios that are formed. Variations in load demand and renewable generation can be noted in this situation. For a total of 24 periods, **Figure 7** shows the total load demand and renewable generation ranging between the minimum and maximum values. From about periods 16 to 22, a considerable variation in load limits can be seen, with the highest load value being 16.20 MW in period 16. When it comes to renewable generation variations, in some situations the variations are minimal due to the extreme events generated involving the renewable production, and a larger variation is demonstrated in periods 13 to 18.

Regarding the considered costs in this scenario, **Figure 8** presents the forecasted wholesale market prices and the external supplier prices. A significant variation can be seen regarding the market prices due to the multiple extreme scenarios that consider an increase in market costs. Here the maximum value of 104.61 m.u. can be seen in period 12, where the minimum value is 43.36 m.u., a rise of 61.25 m.u., which is considerable. The external supplier costs do not suffer a variation and are set to a constant 50 m.u. in off-peak hours and 90 m.u. in peak hours.

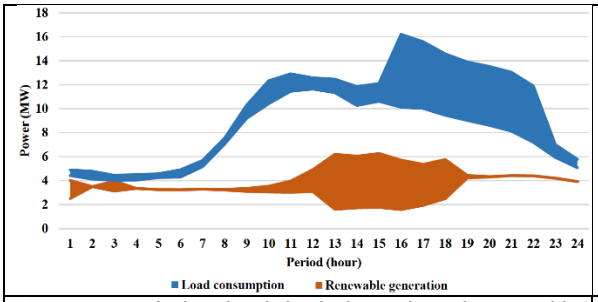

Figure 7 Total day-ahead load demand, and renewable generation forecasted

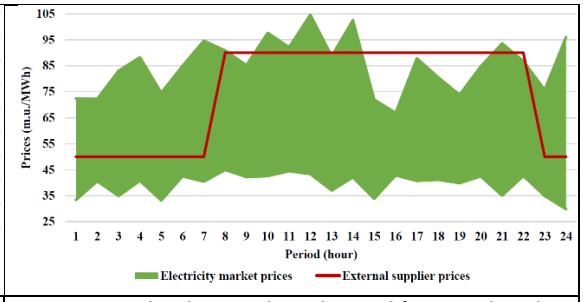

Figure 8 Day-ahead external supplier, and forecasted market prices

The aggregator must manage its respective resources, power bought from the external supplier, and energy bought/sold in the marketplaces to satisfy the consumption. Also are integrated into the DN four capacitors but are

not considered in this problem, so they are set to zero because we did not consider reactive power in the problem formulation. Two ESSs units were also considered in the situation. **Figure 9** presents the energy resources data associated with the aggregator for the day-ahead formulation considering the extreme scenarios. A distinction is made for the minimum and maximum values considered for the prices of the resources, capacity, forecasted values from the renewables and loads, and the number of units corresponding to each resource.

Energy Resources		Prices (m.u./MWh)	Capacity (MW)	Forecast (MW)	Units
		min - max	min - max	min - max	
Photovoltaic		29-29		0.00-0.81	13
Wind		31-31		0.30-3.07	2
External supplier		50-90	0.00-30.00		1
Storage	Charge	110-110	0.00-1.25		2
	Discharge	90-90	0.00-1.25		
Electric vehicles	Charge	0-0	0.01-0.05		500
	Discharge	90-90	0.01-0.05		
Demand response	Reduce program	100-100	0.00-1.21		25
Load		0-0		0.01-2.38	25
Electricity market buy and sell		29.85-104.61			1

Figure 9 Energy resources information for the day-ahead

4. Guidelines for participants

These instructions include as example the metaheuristic differential evolution (DE) [9] 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.

A.0

main.m

```
……
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Select testbed
Select_testbed=1; %Only 1 track in 2022
%Testbed 1: risk-based energy scheduling          A.1
DB=Select_testbed;
% 1: Case study testbed 1


[caseStudyData, DB_name]=callDatabase(DB);
Select_Algorithm=1;                               A.2
%1: DE algorithm (test algorithm)
%2: Your algorithm
algorithm='DE-rand'; %'The participants should include their algorithm here'
DEparameters %Function defined by the participant
No_solutions=deParameters.I_NP;  %  %Notice  that  some  algorithms  are  limited  to  one
individual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Set other parameters                           A.3
otherParameters =setOtherParameters(caseStudyData,No_solutions, Select_testbed);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Set lower/upper bounds of variables            A.4
[lowerBounds,upperBounds] = setVariablesBounds(caseStudyData,otherParameters);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Call the MH for optimization                   A.5
ResDB=struc([]);
    for iRuns=1:noRuns %Number of trails
        tOpt=tic;
        rand('state',sum(noRuns*100*clock))% ensure stochastic indpt trials

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

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %% Save the results and stats              A.6
        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 (only 1 testbed in 2022), 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: risk-based energy scheduling
%% Load Data base
noRuns=20; %this can be changed but final results should be based on 20 trials
DB=Select_testbed;
% 1: Case study testbed 1
% 2: Case study testbed 2
[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);
…..
[ResDB(iRuns).Fit_and_p, ...
                    ResDB(iRuns).sol, ...
                    ResDB(iRuns).fitVector]=                                    ...
deopt_simple(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);
```

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 - *#deopt_simple.m* - Algorithm proposed by the competitor

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

```
[ResDB(iRuns).Fit_and_p, ...
                    ResDB(iRuns).sol, ...
                    ResDB(iRuns).fitVector]=                                    ...
deopt_simple(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 (benchmark_Time.txt)

	timeSpent (s)
Run1	
Run2	
Run3	
…	
Run20	

Table 2. Table_Scenarios: Individual benchmark of the scenario costs (benchmark_Scenarios.txt)

	Avg Scenario	Min Scenario	Max Scenario	Std Scenarios	Var Scenarios
Run1					
Run2					
Run3					
…					
Run20					

Table 3. Table_Fit: Individual benchmark of the trials (benchmark_Results.txt)

	OF	Z^{Ex}	VaR	CVaR	Worst scenario	Bound violations	ConvergenceRate
Run1							
Run2							
Run3							
…							
Run20							

Table 4. Table_TrialStats: Summary statistics or the trials (benchmark_Summary.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 "solutionRiskERM.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 "solutionRiskERM.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 20 trials should be made.**
***5,000 evaluations per trial are allowed for this competition.**

4.B) Fitness function evaluation

***# fitnessFun_riskERM.m* (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 all the scenarios 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*** **(**fitnessFun_riskERM.m") 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 1. 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.

$$RI_{user(a)\_T1} = \frac{1}{N_{trials}} \cdot \left[\sum_{i=1}^{N_{trials}} \left(Fit_a(\vec{X}_{i_{T1}}) \right) \right] \quad (6)$$

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 "`solutionRiskERM.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:

track1_*AlgorithmName*_*ParticipantName*.zip
(e.g., track1_*DE*_*Lezama*.zip).

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

by 15th June 2022 (anywhere on earth)

Appendix: Mathematical formulation

The day-ahead ERM's mathematical formulation takes into account the whole operation cost and profits in each scenario *s*. As a result, the total operational costs and profits for each scenario are calculated as follows [6]:

$$Z_s^{OC} = \sum_{t=1}^{T} \cdot \left(\begin{array}{l} \sum_{i \in \Omega_{\mathrm{DG}}^{\mathrm{d}}} P_{(i,t)}^{\mathrm{DG}} \cdot C_{(i,t)}^{\mathrm{DG}} + \sum_{k=1}^{N_k} P_{(k,t)}^{\mathrm{ext}} \cdot C_{(k,t)}^{\mathrm{ext}} + \\ \sum_{i \in \Omega_{\mathrm{DG}}^{\mathrm{nd}}} P_{(i,t,s)}^{\mathrm{DG}} \cdot C_{(i,t)}^{\mathrm{DG}} + \sum_{l=1}^{N_l} P_{(l,t,s)}^{\mathrm{Curt}} \cdot C_{(l,t,s)}^{\mathrm{Curt}} + \\ \sum_{e=1}^{N_e} ESS_{(e,t,s)}^{\mathrm{Cost}} + \sum_{v=1}^{N_v} EV_{(v,t,s)}^{\mathrm{Cost}} + \\ \sum_{r=1}^{N_r} P_{(r,t,s)}^{ENS} \cdot C_{(r,t)}^{\mathrm{ENS}} + \sum_{i=1}^{N_i} P_{(i,t,s)}^{GCP} \cdot C_{(i,t)}^{\mathrm{GCP}} \end{array} \right) \cdot \Delta t \qquad \forall s \tag{7}$$

$$Z_s^{\mathrm{In}} = \sum_{t=1}^{T} \left(\sum_{m=1}^{N_m} P_{(m,t)}^{\mathrm{Market}} \cdot MP_{(m,t,s)} \right) \cdot \Delta t \qquad \forall s \tag{8}$$

Where:

$$ESS_{(e,t,s)}^{\mathrm{Cost}} = \begin{cases} ESS_{(e,t,s)}^{\mathrm{Power}} \cdot ESSC_{(e,t,)}^{\mathrm{Disch}} & if\ ESS_{(e,t,s)}^{\mathrm{Power}} \leq 0 \\ 0 & otherwise \end{cases} \tag{9}$$

$$EV_{(v,t,s)}^{\mathrm{Cost}} = \begin{cases} EV_{(v,t,s)}^{\mathrm{Power}} \cdot EVC_{(v,t,)}^{\mathrm{Disch}} & if\ EV_{(v,t,s)}^{\mathrm{Power}} \leq 0 \\ 0 & otherwise \end{cases} \tag{10}$$

It's worth noting that $ESS_{(e,t,s)}^{\mathrm{Power}}$ / $EV_{(v,t,s)}^{\mathrm{Power}}$ are continuous variables with a negative value when discharging and a positive value when charging. As a result, in Eq. (7), $ESS_{(e,t,s)}^{\mathrm{Cost}}$ and $EV_{(v,t,s)}^{\mathrm{Cost}}$ are only considered while the battery system is discharging (Eqs. (9)-(10)). To put it another way, the aggregator must pay a fee for the electricity utilized by these technologies. The cost of charging the batteries, on the other hand, is not included in Eq. (7) because it is a payment that the aggregator receives for supplying the requested energy. Similarly, because we only evaluate market transactions in Eq. (8), these payments are not taken into account in this equation.

The suggested approach seeks to minimize Eq. (7) while maximizing Eq. (8) for each case *s* by integrating both components into a minimization problem. The aggregator trades energy as much as feasible to make a profit, depending on market prices and operational costs. The remaining formulations, and constraints can be seen in [6].

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] 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.

[7] 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.

[8] 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.

[9] Fernando. Lezama, Joao. Soares, Enrique. Munoz de Cote, L. Enrique. Sucar, and Zita. Vale, "Differential Evolution Strategies for Large-Scale Energy Resource Management in Smart Grids," 2017.