Engineering Applications of Artificial Intelligence 101 (2021) 104231

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Cellular estimation of distribution algorithm designed to solve the energy resource management problem under uncertainty

Yoan Martínez-López [a,b], Ansel Y. Rodríguez-González [c,d,*], Julio Madera [a], Miguel Bethencourt Mayedo [a], Fernando Lezama [e]

[a] *Universidad de Camagüey, Camagüey, Cuba*
[b] *Universidad Central de Las Villas, Santa Clara, Cuba*
[c] *Consejo Nacional de Ciencia y Tecnología, Ciudad de México, Mexico*
[d] *Unidad de Transferencia Tecnológica, Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE-UT3), Tepic, Mexico*
[e] *GECAD, School of Engineering, Polytechnic of Porto, Portugal*

ARTICLE INFO

Keywords:
Energy resource management
Evolutionary algorithm
Estimation of distribution algorithm
Smart grid
Uncertainty

ABSTRACT

The Energy Resource Management (ERM) can be modeled as a Mixed-Integer Non-Linear Problem whose aim is to maximize profits generally using smart grid capabilities more than importing energy from external markets. Due to this, many resources and customers are involved in optimization, making ERM a complex problem. Moreover, when the inherent uncertainty of weather conditions, load forecast, electric vehicles planned trips, or market prices is considered, deterministic approaches might fail in obtaining optimal solutions to the problem. In this context, evolutionary algorithms are a useful tool to find effective near-optimal solutions. In fact, to design and test evolutionary algorithms to solve the ERM problem under uncertainty, the research community has developed a simulation framework. In this paper, we propose the Cellular Univariate Marginal Distribution Algorithm with Normal-Cauchy distribution (CUMDANCauchy) to address the ERM problem in uncertain environments. CUMDANCauchy uses a univariate estimation of the product of Normal and Cauchy distributions over each feature, and produces new individuals not only by the sampling of the learned distributions but also using neighborhoods of individuals from a ring cellular structure. The experiments performed over two case studies show that: CUMDANCauchy is as competitive as the previous dominant class of algorithms in terms of the global fitness achieved; its convergence behavior is among the best in comparison with the other tested algorithms; its running time is similar to the algorithm with the best global fitness achieved in the first case study, and it is the fastest algorithm in the second one.

1. Introduction

The energy mix of the countries is changing with the massive penetration of renewable energy resources like sun and wind, storage facilities including electric vehicles (EV), and demand response (DR) capabilities, which generates less use of fossil fuels (Clark II et al., 2017). However, the change in the energy mix also poses new challenges in the management and operation of the grid. For instance, the energy production from renewable energy resources depends on weather conditions, or the use of EVs and the energy demand depends on the behavior of end-users. These factors can increase the energy imbalance and deteriorate energy quality (Soares et al., 2013; Yan et al., 2013).

Renewable energy resources are used in smart grids. A smart grid is: “an electricity network that can intelligently integrate the actions of all users connected to it — generators, consumers and those that do both in order to efficiently deliver sustainable, economical and secure electricity supplies” (Gelazanskas and Gamage, 2014). These electricity networks “incorporate technologies such as advanced metering, automation, communication, distributed generation, and distributed storage” (Brown, 2008). Current smart grids also include vehicle-to-grid functionalities, DR programs, external suppliers participation, and bidding in local and external markets (Soares et al., 2016).

In smart grids, the energy resource management (ERM) can be modeled as a Mixed-Integer Non-Linear Problem whose aim is to maximize profits generally using control and automatizing capabilities more than importing energy from external markets (Soares et al., 2016, 2017).

In this context, a framework to test algorithms for the ERM problem under uncertain environments was developed in 2017 (Lezama et al.,

* Corresponding author at: Unidad de Transferencia Tecnológica, Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE-UT3), Tepic, Mexico.

E-mail addresses: yoan.martinez@reduc.edu.cu (Y. Martínez-López), ansel@cicese.mx (A.Y. Rodríguez-González), julio.madera@reduc.edu.cu (J. Madera), miguel.bethencourt@reduc.edu.cu (M. Bethencourt Mayedo), flz@isep.ipp.pt (F. Lezama).

https://doi.org/10.1016/j.engappai.2021.104231
Received 25 August 2020; Received in revised form 11 February 2021; Accepted 8 March 2021
Available online 20 March 2021

2019b). Updated versions of the framework, including new testbeds, were introduced in subsequent years (Lezama et al., 2018c, 2019a). The framework includes testbeds with optimization problems that belong to the NP-hard class. In such kind of problems (e.g., the ERM problem under uncertainty), deterministic algorithms usually struggle finding optimal solutions in acceptable running times or without violating memory/computing limitations. Therefore, alternative methods based in heuristics or probabilistic approaches, such as Evolutionary Algorithms (EAs), can be employed efficiently converting these type of problems into decision problems, and applying a probabilistic search depending on a random number generator. In fact, researchers have shown that probabilistic methods have a better performance in solving decision problems than deterministic approaches under specific circumstances (Bazionis and Georgilakis, 2021; Kim and Hur, 2020; Kabirifar et al., 2020).

EAs have provided near-optimal solutions solving the ERM problem under uncertain environments using the mentioned framework. EAs are a class of algorithms inspired by the biological evolution mechanisms proposed by Darwin, Mendel, and Lamark. They are population-based techniques that include operations like mutation, crossover, and selection (e.g., Genetic Algorithms (GAs) (Goldberg, 1989), Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995; Kohler et al., 2019), Differential Evolution (DE) (Ali et al., 2019; Storn and Price, 1997), Firefly Algorithm (Wang and Song, 2019; Yang, 2009) and Estimation of Distribution Algorithms (EDAs) (Larrañaga and Lozano, 2001; Strickler et al., 2018)).

Several variants of PSO, DE, Firefly, and hybrid techniques have been implemented and tested competing against each other using the aforementioned framework. Not all of them have a related publication, but all the implementations are publicly available as part of the series "Competition on Evolutionary Computation in the Energy Domain: Smart Grid Applications"[1] launched each year since 2017 (Lezama et al., 2018c, 2019a).

The main contribution of this paper is a novel EDA, named $CUMDANCauchy$, designed to solve this problem. EDAs Larrañaga and Lozano (2001), Strickler et al. (2018) are a group of EAs that use the estimation of a probability distribution of the best individuals and the production of new individuals by the sampling of the learned probability distribution as a substitute of the crossover and mutation operators of the genetic algorithms. To the best of our knowledge, this is the first time that EDAs are developed and tuned to solve the ERM problem under uncertain environments. Unlike other EDAs, $CUMDANCauchy$ uses not only the product of the Normal and Cauchy distributions but also the neighborhoods of individuals from a ring cellular structure. $CUMDANCauchy$ is as competitive as the previous dominant algorithms class in terms of the global fitness achieved to solve the ERM problem under uncertainty. Also, $CUMDANCauchy$ does not require a priori knowledge of the problem, as other algorithms belonging to the dominant class, which allows $CUMDANCauchy$ to be used for more general-purpose (e.g., to be applied in different case studies). The convergence behavior of $CUMDANCauchy$ is among the bests compared with the other state-of-the-art algorithms. In terms of running time, considering the two case studies presented in this work, $CUMDANCauchy$ is similar to the algorithm with the best global fitness in the first one, and the fastest algorithm in the second case study.

A brief introduction to $CUMDANCauchy$ with only preliminary results was provided in an extended abstract (2 pages article) published in Martínez-López et al. (2019b). The main differences of the current paper with that extended abstract are: (I) an overview of the ERM problem under uncertain environment, the framework used to test algorithms, including the case studies used, and the state-of-the-art algorithms used to solve this problem is provided; (II) the proposed algorithm, CUMDANCauchy, is explained in more detail, including its computational complexity analysis; (III) a broader validation of CUMDANCauchy, comparing it with state-of-the-art algorithms, used to solve the problem for the case studies is carried out. (IV) the evaluation and performance of the algorithm is measured using not only the ranking index but also the runtime and the average convergence rate as metrics. In addition, Shapiro–Wilks test, Friedman test, Iman–Davenport's test, and Wilcoxon test were performed for statistical validation.

The outline of this paper is as follows: in Section 2, the ERM problem under uncertain environments is explained. Section 3 describes the simulation framework used to test the algorithms, and the case study used. Section 4 provides the related work of the research. In Section 5, the proposed algorithm is presented. In Section 6, the experimental results are shown. Finally, in Section 7 conclusions and future work are discussed.

2. Preliminaries

The study of optimal scheduling problems dates back to approximately 1950 when industrial engineering researchers, operational research, and administrators developed new approaches and algorithms whose main objective was the reduction of production costs in the industry (Leung, 2004). Optimal scheduling problems focus on assigning tasks over time, meeting certain constraints (e.g., resource limitations) while optimizing one or more objectives. The vast majority of optimal scheduling problems are computationally complex, and the time required to calculate an optimal solution increases with the size of the problem. Besides, many of these problems belong to the $NP-Hard$ class (Brucker et al., 2004; Lenstra and Rinnooy Kan, 1978) (e.g., in the context of energy: the optimal active/reactive power dispatch problem (Awad et al., 2019; Biswas et al., 2019) and optimal schedule renewable energy sources problem (Gholami and Dehnavi, 2019)).

The ERM under uncertain environments is also considered one of the most complex optimization problems. This fact is due to its nonlinearity, combinatorial nature, and a large number of resources and costumers involved, which causes high dimensionality and processing limitations (Lezama et al., 2018a,b, 2019b).

The next subsections describe the ERM problem under uncertainty, the mathematical formulation, the fitness function modeling, and the simulation framework used to test the optimization algorithms.

2.1. ERM problem under uncertain environments

The ERM problem (Lezama et al., 2018a, 2019b; Soares et al., 2017) under uncertainty considers an energy aggregator (AGG) in order to meet the energy needs through renewable energy resources, distributed generation (DG), and the electricity market. The AGG seeks the minimization of operating costs while obtaining income from energy sales in the available electricity markets. Besides, the AGG has available different flexible assets, such as energy storage systems (ESS), Electric Vehicles (EV), and demand response (DR), to either meet the load demand or reduce power consumption. DR flexibility is modeled as curtailable loads by direct load control programs in which consumers voluntarily participate and receive a monetary compensation if their loads are reduced (Lezama et al., 2018a).

The main target of the AGG is obtaining the scheduling of energy resources for the day-ahead (i.e., the 24 h of the next day). However, the day-ahead scheduling is influenced by uncertainty in different parameters. For instance, uncertainty affecting the schedule of resources includes the prediction of weather conditions (e.g., forecast of renewable generation), load consumption of users, EV coordination of trips, and variations in market prices. Therefore, assumptions considering a "perfect" or "highly accurate" forecast (typically done in many studies) can have catastrophic consequences to the operation and management of the network if the expected predictions differ in a large degree from

[1] http://www.gecad.isep.ipp.pt/ERM-competitions/competitions/.

the real conditions. Therefore, it is desired that the AGG determines solutions that are robust to such uncertainty.

To do that, the AGG must find solutions that provide not only an optimal (or near-optimal) value of operating costs but also that have the characteristic of being as resilient as possible to variations in the parameters with uncertainty.

2.2. Mathematical formulation

The problem can be modeled as a Mixed-Integer Linear Programming (MILP) problem due to the presence of continuous, discrete and binary variables. Thus, the objective of the AGG is to minimize operational costs (OC) while maximizing incomes (In).

The operational costs (OC) are associated with the management of resources and defined as:

$$OC = \sum_{t=1}^{T}\sum_{i=1}^{N_{DG}} P_{DG(i,t)} \cdot C_{DG} + \sum_{t=1}^{T}\sum_{k=1}^{N_k} P_{ext(k,t)} \cdot C_{ext(k,t)} + \sum_{s=1}^{N_s}\sum_{t=1}^{T}\left(\begin{array}{l} \sum_{j=1}^{N_{PV-DG}} P_{PV(j,t,s)} \cdot C_{PV(j,t)} + \\ \sum_{e=1}^{N_e} P_{ESS^-(e,t,s)} \cdot C_{ESS^-(e,t)} + \\ \sum_{v=1}^{N_v} P_{EV^-(v,t,s)} \cdot C_{EV^-(v,t)} + \\ \sum_{l=1}^{N_L} P_{curt(l,t,s)} \cdot C_{curt(l,t)} + \\ \sum_{l=1}^{N_L} P_{imb^-(l,t,s)} \cdot C_{imb^-(l,t)} + \\ \sum_{i=1}^{N_{DG}} P_{imb^+(i,t,s)} \cdot C_{imb^+(i,t)} + \end{array}\right) \cdot \pi(s) \tag{1}$$

where Eq. (1) considers the costs associated with DG, external suppliers, discharge of ESS and EVs, DR by direct load control programs (curtailable loads), penalization of non-supplied demand (negative imbalance) and penalization for excess of DG units generation (positive imbalance). Notice that some of the variables depend on the index s that indicates the probability $\pi(s)$ of occurrence of a given scenario.

In addition, the AGG search for the maximization of incomes (In) received from market transactions and is modeled as:

$$In = \sum_{s=1}^{N_s}\sum_{t=1}^{N_T}\left(\sum_{m=1}^{N_m}\left(P_{buy(m,t)} - P_{sell(m,t)}\right) \cdot MP(m,t,s)\right) \cdot \pi(s) \tag{2}$$

where bids and offers (to buy or sell energy) into two distinctive markets (wholesale and local market) are allowed and depend on the forecasted prices over different scenarios (again, the index s is used to model the probability $\pi(s)$ of a given scenario).

Both objectives (i.e., Eqs. (1) and (2)) can be combined in a single-objective function as:

$$Minimize \qquad f(\vec{x}) = OC - In \tag{3}$$

where $f(\vec{x})$ is the objective function to optimize. The minimum value of $f(\vec{x})$ represents the total cost (or profits if the value is negative) for the AGG considering the whole set of scenarios. Notice also that the formulation needs to consider different constraints related to: (i) energy balance Constraint (i.e., the amount of generation must be equal to the amount of consumption at every time t); (ii) DG maximum and minimum power limits; (iii) ESS constraints related to charging rates and capacity limits; (iv) EV constraints similar to the ones needed for ESS plus scheduling and user requirements; (v) DR constraints stating maximum active power that curtailable loads can reduce; (vi) market constraints stating bidding limits depending the type of market (e.g., local markets impose lower power limits to participate compared with the wholesale market). The reader can be referred to the appendix section for the nomenclature used in this work, and to Soares et al. (2017), Lezama et al. (2018a) to consult the complete mathematical model used in this work including all the constraints.

2.3. Fitness function under uncertainty

In general, the goal is to minimize operational costs (OC) and maximize incomes (In). Therefore, the fitness function can be mapped to a Z function combining both objectives linearly:

$$minimize\ Z = OC - In \tag{4}$$

where OC and IN are equivalent to Eqs. (1) and (2). Taking into account the constraints of the ERM problem, the fitness function f' is defined considering the objective function Z (Eq. (4)), plus penalties found during the evaluation of the solutions:

$$f'(\overrightarrow{X}) = Z + \rho \cdot \sum_{i=1}^{N_c} max[0, g_i] \tag{5}$$

where $\overrightarrow{X}$ represents a given solution, N_c is the number of constraints and g_i is the value of the ith constraint. The parameter ρ is a configurable penalty factor, and generally, it is a high value.

In a more realistic situation, in which the AGG does not have a "perfect" or "highly accurate" forecast of uncertain variables (e.g., renewable generation, load demand, EV trips), the variability associated with some parameters can modify the value of the fitness function. Thus, a set of scenarios considering predictions and their associated deviations (i.e., errors) determined by available historical data (or expert knowledge) is generated under the assumption that such scenarios model close real-world conditions. In particular, this article models uncertainty associated to: (i) PV generation, (ii) load consumption profiles, (iii) EVs' trips, and (iv) wholesale and local market variations of price.

A method based on Monte Carlo Simulation (MCS) to generate a correct set of scenarios that simulate real-world conditions as in Soares et al. (2017) is provided. First, MCS is used to create a substantial number of scenarios applying a probability distribution function of the forecast errors as:

$$X_s(t) = x^{forecast}(t) + \mathcal{N}(0, \sigma_{error}) \tag{6}$$

where $x^{(error,s)}$ represents a normal distribution function having a mean value equal to zero and considering a standard deviation σ_{error} (i.e., the associated error of the function). Also, $x^{forecast}(t)$ represents the forecasted value of a given variable x at time t. For simplicity, a normal distribution function represents the forecast errors of the inputs with uncertainty.

After that, a reduction technique is applied to eliminate scenarios with a low probability of occurrence. This is achieved by combining scenarios that are statistically close to each other, resulting in a reduced set of scenarios that is representative of the more extensive set (for a complete description of these techniques see Soares et al. (2017)).

Consequently, the fitness function is modified by the disturbances as follows:

$$F_s(\overrightarrow{X}) = f'(\overrightarrow{X} + \delta_s) \tag{7}$$

where $F_s(\overrightarrow{X})$ is the fitness in the scenario S and δ_s is the disturbance of the variables and parameters in the same scenario.

Hence, an expected mean value and a standard deviation for a given solution on the set of N_s scenarios considered can be calculated as:

$$\mu FS(\overrightarrow{X}) = \frac{1}{N_s} \cdot \sum_{s=1}^{N_s} F_s(\overrightarrow{X}) \tag{8}$$

$$\sigma FS(\overrightarrow{X}) = \sqrt{\frac{1}{N_s} \cdot \sum_{s=1}^{N_s}\left(F_s(\overrightarrow{X}) - \mu FS(\overrightarrow{X})\right)^2} \tag{9}$$

Finally, the global fitness function GF is defined as the sum of the expected mean value (Eq. (8)) and the standard deviation (Eq. (9)):

$$GF(\overrightarrow{X}) = \mu FS(\overrightarrow{X}) + \sigma FS(\overrightarrow{X}) \tag{10}$$

Table 1
Available energy resources.

Energy resources		Prices (m.u./kWh)	Capacity (kW)	Units
Dispatchable DGs		0.07–0.11	10–100	5
External suppliers		0.07–0.16	0–150	1
ESS	Charge	–	0-16.6	2
	Discharge	0.03	0-16–6	
EV	Charge	–	0–111	34
	Discharge	0.06	0–111	
DR curtailable loads		0.04	4.06–8.95	90
Forecast (kW)				
Photovoltaic		–	0-106.81	1 (17 agg)
Load		–	35.82–83.39	90
Limits (kW)				
Market 1 (WS)		0.021–0.039	0–85	1
Market 2 (LM)		0.021–0.039	0–40	1

3. Framework features

In order to test algorithms solving the ERM problem, a first simulation framework was developed in 2017 (Lezama et al., 2019b). This framework addressed a large-scale optimization problem, but consideration of uncertainty was neglected. Thus, an updated version was introduced in 2018 (Lezama et al., 2018c), with a more challenging scenario recently proposed in 2019 (Lezama et al., 2019a). The framework was implemented in MATLAB © 2016 64-bit and consisted of different scripts with specific objectives in the simulation. Scripts are divided into two groups. The first group includes encrypted files provided by developers. Encrypted files contain the fitness function, variables bounds, case study information, scenarios, and other encrypted parameters. The second group includes modifiable files by users. In this way, the user only needs to implement two scripts: a script to configure the parameters required by the algorithm and a script for the implementation of the proposed solution method. The framework restricts the number of evaluations of the objective function to a maximum of 50000.

3.1. Case study

The large-scale energy resource management problem considers different distributed energy resources and renewables, some of them with associated uncertainty or variability, posing a great deal of complexity to the matter. The case study considered in this paper (as it was in Lezama et al. (2018c, 2019a)) is based on a 25-bus low voltage network representing a microgrid residential area with 6 DGs (5 dispatchable units and 1 PV generator), 1 external supplier, 2 ESSs, 34 EVs, and 90 loads with DR by direct load control capability. Also, it is considered that the AGG has access to two different markets (wholesale and local) to buy/sale energy for the next 24 h (i.e., day-ahead markets). Table 1 summarizes the available resources in the network and their characteristics.

Both 2018 and 2019 case studies consider the same network and resources. The difference between them lies mainly in the number of scenarios considered to represent variables with uncertainty. In the 2018 case study, 100 scenarios were generated for the PV generation, Load demand, and market prices using the method explained in Section 2.3. Also, 100 scenarios of EV trips were considered varying the availability of EVs connected to the network in a given time. In the 2019 case study, apart from considering a higher level of uncertainty on the PV production (i.e., a larger error in the generation of scenarios), the main difference is the consideration of 500 scenarios for the variables with associated uncertainty. The increase in the number of scenarios impacts these main aspects: first, considering more scenarios provides more robustness to the solution found since such solution will show an acceptable performance over a large set of possibilities; second, considering more scenarios increases the complexity of the problem, since the search space becomes larger in relation with the considered number of scenarios (which also has an impact in the computational burden and execution times). As can be seen, it is desired to consider a larger set of scenarios while keeping the computational complexity low. Therefore, 2018 and 2019 case studies offer a common ground to compare EAs over two similar scenarios, although different in complexity.

3.2. Individual encoding

The solutions to the ERM problem under uncertainty are represented in the framework as a vector.

The vector represents a solution with ‘6’ groups of variables (i.e., active power, generators binaries, EVs charge /discharge, ESS charge/ discharge, demand response, and market) that are repeated 24 times (according to the 24 h periods of optimization. The dimension of each group depends on the number of DG, EV, ESS, load, markets, and external suppliers, available in the case study. As a result, the vector contains 3408 variables to be optimized (i.e., the problem is of dimension 3408). As can be observed in the figure, variables from groups active power, EVs charge/discharge, ESS charge/discharge, demand response, and market are continuous variables with bounds matching the power or capacity limits of the associated resources. The group generators binaries corresponds to binary variables that are used to indicate if a generator is connected (’1’ value) or disconnected (’0’ value) to the network. Notice also that variables representing PV generation cannot be controlled (because PV generation cannot be controlled in practice), so even when it is part of the vector solution, variables corresponding to the PV generation (last variable of group active power) take an unalterable value depending on the considered scenario.

The uncertainty considered was introduced into the 2018 and 2019 frameworks through 500 and 100 scenarios, respectively, for the variables with uncertainty. Besides, the 2019 case study considers a higher level of uncertainty in the PV production (i.e., a larger error in the generated scenarios) and limits the amount of energy that the AGG can buy in the electricity market. This difference impacts the options available to find a feasible solution to the problem, increasing its complexity.

4. Related work

This section provides a look into the EAs used to solve the ERM problem under uncertainty. Besides, an introduction to the EDAs, the class to which the proposed algorithm belongs, is provided.

4.1. EAs used to solve the ERM problem under uncertainty

Despite the excellent performance of some of the EAs used to solve the ERM problem under uncertainty, not all of them and their versions have been published, which limits the available information and details on their implementation. Nevertheless, the implementation of the algorithms are publicly available as part of the series “Competition on Evolutionary Computation in the Energy Domain: Smart Grid Applications” (Lezama et al., 2018c, 2019a).

In general, the developed algorithms in this competition are based on Firefly algorithm (Wang and Song, 2019; Yang, 2009), Variable Neighborhood Search (Mladenović and Hansen, 1997; Qiu et al., 2018), Differential Evolution (Ali et al., 2019; Storn and Price, 1997), Particle Swarm Optimization (Kennedy and Eberhart, 1995; Kohler et al., 2019), and hybrid versions of them.

Firefly algorithm (Wang and Song, 2019; Yang, 2009) is inspirited by the flashing light of fireflies. This algorithm has two essential operators: the variation of light intensity and the formulation of attractiveness. The algorithm is described as follows: for each iteration, the intensity of the light of the fireflies is determined by means of the objective function. Then, each firefly is moved depending on its attractiveness to other fireflies, which is calculated using a coefficient of light absorption of the fireflies, the intensity of the light, and the distance between the fireflies. Next, the attractiveness of the fireflies and their intensity of the light is updated. After the last iteration, the fireflies are ranked, and the current best of them is selected.

Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997; Qiu et al., 2018) exploits the idea of systematic neighborhood change, and it is a decent, first-improvement method with randomization. Exploration is carried out to select the best neighbor. After, if the best neighbor is better than the current solution then the current solution, is replaced with this neighbor, changing the neighborhood used by another from a set of a predefined neighborhood, if no movement is done.

Differential Evolution (DE) (Ali et al., 2019; Storn and Price, 1997) is a simple continuous variable evolutionary algorithm. The DE algorithm is described as follows: a P population of individuals is obtained randomly. Then, a candidate individual is created from each parent individual of the P population. DE creates new candidate solutions by combining the parent individual and several other individuals of the same population. Then, the candidate is evaluated, and if it is better than a parent, then it replaces to parent from the P population.

Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995; Kohler et al., 2019) simulates the elegant but unpredictable movement of a flock of birds that can be seen as a social organism, whose individuals are looking for the optima into a multidimensional search space. For each iteration, the fitness of the individuals (particles) is computed. Then from the best position of each particle and the best global position achieved by the algorithm, the speed of each particle is adjusted, and a new position is obtained.

Below is the description of the state-of-the-art algorithms developed or tuned for the ERM problem under uncertain environments:

- Chaotic Evolutionary PSO ($CEPSO$): It is a methodology that uses chaotic agents to search in promising areas that are explored by Particle Swarm Optimization (PSO). This algorithm uses a structure similar to Hopfield neuronal networks with transient chaos to avoid premature convergence of the algorithm in the search for the global optimum (Sun et al., 2009).
- PSO with Global Best Perturbation (PSO-GBP): It is a type of PSO with chaotic dynamic perturbations introduced. This method uses small perturbations to perform the change in the search for the global optimum (Zhang et al., 2009), in which the Global Best Perturbation (GBP) and inertia weight jump threshold are adopted. GBP allows rapid convergence towards the optimum and avoids being trapped in local optima (Chen et al., 2020).
- Unified PSO ($UPSO$): This PSO algorithm is inspired by the behavior of socially organized colonies, where the strengths of the local and global variants of PSO is combined. This combination enhances the ability of PSO to explore and exploit in the search for the global optimum (Parsopoulos and Vrahatis, 2019).
- Variable Neighborhood Search algorithm and Differential Evolutionary Particle Swarm Optimization (VNS-$DEEPSO$): It is an assembly between two metaheuristics VNS (Variable Neighborhood Search) and $DEEPSO$ (Differential Evolutionary Particle Swarm). This algorithm explores the possibilities of distant neighborhoods in the search for the optimum, combining the generation of particles with DE, the power of self-adaptive evolutionary algorithms, the power of PSO exploration, and the probabilities of communication between particles (Garcia-Guarin et al., 2019, 2020).
- Hybrid Levy Particle Swarm Variable Neighborhood Search Optimization (HL-$PSVNSO$): It is a hybridization algorithm between VNS and PSO using Levy Flight. In this algorithm, the effective initialization of the VNS in the population is used, taking a near-optimal solution (Dabhi and Pandya, 2020b).
- Gauss Mapped Variable Neighborhood Particle Swarm Optimization ($GM-VNPSO$): This algorithm uses Gauss-mapped (Hyde et al., 1997) method combined with PSO and VNS. $GM-VNPSO$ generates an initial solution using the Gauss-mapped and then perform the search for the optimum with the VNS guided by the PSO.
- Improved Chaotic Differential Evolutionary Particle Swarm Optimization ($IC-DEEPSO$): This optimization method has its basis in the Particle Swarm Optimization (PSO) method. It is an improved variant of DEEPSO, where a chaotic motion of the particle is introduced to avoid local optima and the solution converges to a global optimum, thus diversifying the search space of the solution (Ma et al., 2019).
- Enhanced Velocity Differential Evolutionary Particle Swarm Optimization ($EVDEPSO$): It is an improved version of the DEEPSO algorithm, which introduces improvements in the velocity and position equation. The improved velocity equation of the hybrid technique built by the different metaheuristics, PSO, and DE, gives the advantage of finding the optimum in a globally correct direction and makes $EVDEPSO$ in a robust algorithm (Dabhi and Pandya, 2020a).
- Differential Evolution with Stochastic Selection ($DESS$): This algorithm uses the stochastic selection in the DE and then applies the mutation and crossover operators. Also, the generation of the population individuals is generated using the Cauchy distribution (Palakonda et al., 2018).
- Hybrid Fast Enhanced Artificial Bee Colony ($HFEABC$): It is an improved, fast and hybrid Artificial Bee Colony (ABC) method, which uses the search equation based on the product of the inverse cumulative distribution function of the Cauchy and normal distribution. This strategy is used to enhance the performance of ABC in terms of higher convergence speed and robustness concerning the original ABC (Yu et al., 2018).
- Cross Entropy Covariance Matrix Adaptation Evolution Strategy (CE-$CMAES$): This method uses the Cross Entropy (CE) for the global exploration of the search space, having as characteristic the fast convergence and uses the Covariance Matrix Adaptation Evolution Strategy ($CMAES$) for the local exploitation of the search space, adapting it as a mechanism for the generation of new individuals, avoiding getting stuck in local minima (Sarda et al., 2019).
- Differential Evolution and Teaching–Learning-Based Optimization (DE-$TLBO$): This algorithm combines DE with Teaching–Learning-Based Optimization ($TLBO$), where DE plays the role of "Teachers Refresher" in $TLBO$. $TLBO$ uses the crossover and mutation operators of DE for learning (Jiang and Zhou, 2013; Rao, 2016).
- Advanced JSO ($AJSO$): This algorithm implements a variant of the iL-$SHADE$ algorithm (Hadi et al., 2021), mainly a weighted version of the mutation strategy. This variant of the DE algorithm modifies the mutation and crossover operator (Brest et al., 2017; Shen et al., 2021).
- Genetic Algorithm, Simulated Annealing and Particle Swarm Optimization ($GASAPSO$): This algorithm is a hybridization between the Genetic Algorithm (GA), PSO, and Simulated Annealing (SA). It uses the local search space of GA, the convergence capability of PSO, guided by the local search of PSO combined with SA (Dou and Xiong, 2017).

As can be seen, different variants of PSO, DE, VNS, and $Firefly$ used to solve the ERM problem have been proposed to obtain better results to find the optimal global solution. However, before this research, Estimation of Distribution Algorithm (EDA) (Larrañaga and Lozano, 2001; Strickler et al., 2018) has not been developed and tuned for this problem. In the following subsection, this type of evolutionary algorithm will be described.

4.2. Estimation of distribution algorithms

The EDAs Larrañaga and Lozano (2001), Strickler et al. (2018) are a group of Evolutionary Algorithms that use the estimation of the probability distribution of the best individuals and the production of new individuals by the sampling of the learned distribution as substitute of the crossover and mutation operators of the genetic algorithms. A variant of this algorithm is the Univariate Marginal Distribution Algorithm (UMDA) (Pelikan and Mühlenbein, 1998) that generalizes the previous work (Mühlenbein and Voigt, 1996), in which theoretical bases are introduced. In each generation, UMDA learns a joint probability distribution of the selected population, which is factored as the product of the independent univariate distributions of each feature. The univariate probability distributions are estimated from the marginal frequencies. Then, UMDA produces the individuals of the next generation sampling the learned factored probability distribution.

The Cellular Estimation of Distribution Algorithms (CEDAs) (Alba et al., 2006) were introduced as a decentralized version of EDA and also as a generalization of the cellular models developed for genetic algorithms. They are a type of evolutionary algorithm of discrete groups based on spatial structures that define overlapped neighborhoods. The CEDAs commonly use a grid as the spatial structure, in which each cell contains a set of neighboring individuals, and nearby cells make up neighborhoods. For each cell, a local EDA, that uses the set of individuals in its neighborhood as the population, is applied (Alba et al., 2006; Martínez-López et al., 2019a). Some studies claim that CEDAs need less fitness function evaluations to achieve the same results than other EDAs Madera et al. (2006), Martínez-López et al. (2019a).

5. Cellular Univariate Marginal Distribution Algorithm with Normal-Cauchy distribution ($CUMDANCauchy$)

In this section, the proposed algorithm, $CUMDANCauchy$, is introduced. Besides, an analysis of its computational complexity is provided.

5.1. Proposed algorithm

Cellular Univariate Marginal Distribution Algorithm with Normal-Cauchy distribution ($CUMDANCauchy$) is a cellular Evolutionary Algorithm that uses a univariate estimation of the product of Normal and Cauchy distributions over each gene and produces new individuals by sampling the learned distribution.

In general, $CUMDANCauchy$ can use the Normal distribution, Cauchy, Lévy, Laplace, Uniform or a distribution product for estimating the new values of a random variable. However, several research works (Abbass, 2002; Pant et al., 2009; Price, 2013; Thangraj et al., 2010; Xiaogang and Jingshou, 2009) have shown the advantages of using the Normal and Cauchy distributions into another kind of EA, the Differential Evolution. From this fact, the proposed algorithm, $CUMDANCauchy$, uses a combination of both.

Given that the default population size provided in the framework is 10, the algorithms were developed to use a population size close to this value. However, classic grid-based neighborhoods of Cellular Estimation of Distribution Algorithms involve a large number of individuals to define the neighborhoods. Consequently, we propose to use a ring as the structure on which neighborhoods are defined, which is a more simple structure and requires few individuals.

The neighborhood is defined for each individual as k neighbors on the left, k neighbors on the right and itself (see Fig. 1). The size of neighborhood is $2k+1$, where $k \in \mathbb{N}$ and $2k+1 \leq N$.

Learning a probability distribution from a small set of values is a limitation, therefore, for the ERM problem under uncertain environments, $CUMDANCauchy$ uses the maximum neighborhood size, i.e., the population size.

$CUMDANCauchy$ (Algorithm 1) starts by randomly generating a population of N individuals (step 2). Next, the ring cellular structure is

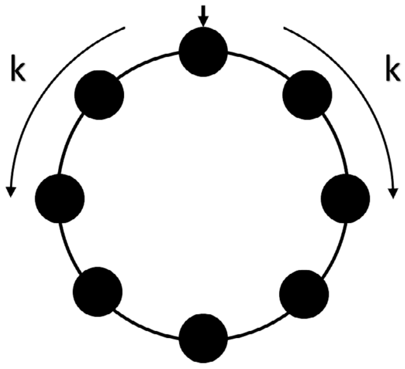

Fig. 1. Representation of ring cellular structure.

created from generated individuals (step 3). Then an iterative process is executed while the maximum number of iterations is not exceeded (steps 4 to 14). In each iteration, a subpopulation of M individuals is selected (step 5). Also, the combination of Normal and Cauchy distributions for each gene is estimated (step 6). Besides, for each cell, new individuals are produced by the sampling of the learned distribution (step 8). The new individuals are inserted into an auxiliary population (step 9). Before finishing each iteration, the current population is replaced by the auxiliary one, and some basic statistics are computed and updated (steps 11 and 12).

5.2. Computational complexity

To determine the computational complexity of $CUMDANCauchy$ is required to obtain the computational complexity of the initialization step and inner iteration steps (i.e., selection, estimation, and sampling):

- Initialization: The initialization step of $CUMDANCauchy$ consists of assigning the values to all the individuals in the initial population, which has complexity $O(Nn)$, where N is the population size, and n is the number of features that describe to each individual.
- Selection: Although $CUMDANCauchy$ can use different selection methods, currently, it uses truncation selection, which has a time complexity of $O(NlogN)$.
- Estimation: The complexity of the estimation step is $O(Mn)$. It is the cost to compute the μ and σ of every variable of the M selected solutions.
- Sampling: The sampling step complexity depends on the total number of individuals produced and the number of features. The time complexity to generate each feature value is constant. Then the complexity of the sampling step is $O(Nn)$.

From the previous analysis, the computational complexity of each iteration of CUMDANCauchy is $O(NlogN + Mn + Nn)$, which is equivalent to $O(NlogN + Nn)$ because of $M \leq N$. Also, considering that n is larger than N for the ERM problem, in this case $O(NlogN + Nn) = O(Nn)$.

Let P be the maximal number of allowed generations. Finally, the complexity of CUMDANCauchy for the ERM problem can be roughly estimated as $O(Nn + PNn) = O(PNn)$. This computational complexity ($O(PNn)$) is the same as the computational complexity of PSO and DE. Still, it is lesser than the computational complexity of the Fireflies Algorithm, which is $O(PN^2n)$ (Yang and He, 2013).

6. Experiment results

In this section, the performance of the proposed algorithm, CUMDANCauchy, is shown and compared with state-of-the-art algorithms.

Algorithm 1 CUMDANCauchy

```
1: $t \leftarrow 1$
2: $Generate\ N\ individuals\ randomly$
3: $Create$ Ring Cellular structure from $N\ individuals$
4: while $t \leq MaximumIteration$ do
5:     Select globally $M \leq SizeOf(RingCellular)$
6:     Estimate the distribution $Normal(\mu,\sigma) \times Cauchy(\mu,\sigma)$ of the $M$ selected individuals for each gene
7:     for all cell do
8:         Sample the $SizeOf(cell)$ new individuals according to the estimated distribution of each gene
9:         Insert the generated individuals in the same cell of an auxiliary population
10:    end for
11:    Replace the current population with the auxiliary one
12:    Compute and update the statistics
13:    $t \leftarrow t + 1$
14: end while
```

Table 2
Set of algorithms and case study for experiments i, ii, iii and iv.

Experiment	Set of algorithms			Case study	
	2018	2019	2020	1	2
i	✓			✓	
ii	✓				✓
iii		✓			✓
iv			✓		✓

The comparison was made regarding three measures: ranking index, average convergence rate, and runtime. The ranking index is defined as the average of the global fitness function (Section 2.1 Eq. (10)) of the best solution obtained in each execution. Since the problem is a minimization problem, the lesser the ranking index is, the better the performance of the algorithm is. Also, it is relevant to know how fast an algorithm converges towards the optimal value in each iteration or sequence. The convergence is usually represented as a graph showing the variation of fitness/cost value with either time unit, the number of iterations or function evaluations (Halim et al., 2020; Tuo et al., 2020). The average convergence rate (He and Lin, 2015; Chen and He, 2019; Tuo et al., 2020) is used as a measure of convergence. The time required by an algorithm to reach its best solution (runtime) is the other measure used.

For the set of experiments to evaluate the performance of $CUMDANCauchy$, we have used the two available case studies (i.e., the 2018 and 2019 frameworks), and the set of algorithms submitted, and openly available, to the 2018, 2019, and 2020 editions of the "Competition on Evolutionary Computation in the Energy Domain: Smart Grid Applications" (http://www.gecad.isep.ipp.pt/ERM-competitions/). With this elements, four experiments are conducted: (i) In the first experiment (Section 6.1), the performance of $CUMDANCauchy$ is compared with the algorithms developed in 2018 using the 2018 framework (i.e., in the case study 1); (ii) In the second experiment (Section 6.2), $CUMDANCauchy$ is compared with the same algorithms developed in 2018, but this time using the 2019 framework (i.e., in the case study 2); (iii) In the third experiment (Section 6.3), $CUMDANCauchy$ is compared with the algorithms developed in 2019 using the framework from 2019 (i.e., in the case study 2); and in the fourth experiment (Section 6.4), $CUMDANCauchy$ is compared with some more recent algorithms developed in 2020, using the framework from 2019. Table 2 summarizes the set of algorithms and case studies used in each experiment.

The experiments i, ii, iii and iv, were designed to provide insights on the robustness of the approaches over different case studies. For instance, the algorithm with the best performance in 2018, the VNS-$DEEPSO$, takes advantage from knowledge of the structure of the problem, defining intelligent rules to explore more efficiently the neighborhoods around promising solutions. In fact, the first three ranked algorithms in 2019 used the properties of VNS, achieving good results. While exploiting the knowledge of the problem to achieve excelled results in different applications of EAs is acceptable (Soares et al., 2016), the goal of designing algorithms with a more general-purpose (e.g., applicable to various case studies) is undermined. $CUMDANCauchy$ does not take advantage of problem-knowledge, and yet is able to achieve competing performance despite the case study to be solved, as is shown in the results.

In addition, a fifth experiment (Section 6.5) is carried out to evaluate the sensitivity of $CUMDANCauchy$ configuration parameters. Finally, a comparison using different statistical tests was made to validate the results (Section 6.6).

The proposed algorithm was implemented in Matlab R2016a 64-bit. All the algorithms tested and the frameworks used are available on the web page of the "Smart grid problems Competitions".[1] Experiments were executed in a PC with Intel (R) Core(TM) i7-4770 @ 3.40 GHz processor and 16 GB of RAM. The experiments' raw results, the algorithms implementations with the parameters used, and the routines for the statistical tests are available on GitHub.[2]

6.1. Experiment (i): set of the 2018 algorithms in case study 1

This experiment consisted of 20 independent executions of the proposed algorithm, $CUMDANCauchy$, and the algorithms set of the 2018 algorithms in case study 1, i.e., Combination of Variable Neighborhood Search algorithm and Differential Evolutionary Particle Swarm Optimization (VNS-$DEEPSO$), Enhanced Velocity Differential Evolutionary Particle Swarm Optimization (EV-$DEPSO$), Chaotic Evolutionary Particle Swarm Optimization ($CEPSO$), Particle Swarm Optimization with Global Best Perturbation (PSO-GBP), Improved Chaotic Differential Evolutionary Particle Swarm Optimization (IC-$DEEPSO$), Unified-PSO ($UPSO$), Improved Differential Evolution ($Improved\ DE$), Firefly algorithm ($Firefly$), and Differential Evolution with Stochastic Selection ($DESS$), which participated in the 2018 Competition on Evolutionary Computation in Uncertain Environments: A Smart Grid Application. $CUMDANCauchy$ used the predefined parameters applied in the 2018 framework (i.e., population size = 10 and max iterations = 499); while the other algorithms used the parameters defined by their developers.

Fig. 2 shows the ranking index of each algorithm. As can be seen, $CUMDANCauchy$ achieved the third position out of 10 algorithms. Notice that, except for $CUMDANCauchy$, PSO variants are among the first seven positions in the competition, while the three approaches that do not use PSO presented the worst performance.

In fact, the third position achieved by $CUMDANCauchy$ divides the block of PSO variants in two, being the performance of $CUMDANCauchy$ better than four of the six PSO variants, and showing its value as a good tool to solve the ERM problem under uncertainty.

[2] https://github.com/cybervalient/GECCO_CEC_smartgrid_experiments.

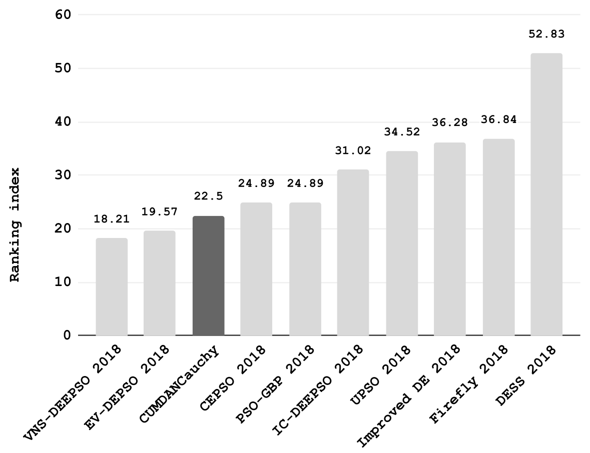

Fig. 2. Ranking index of CUMDANCauchy and the 2018 algorithms in case study 1.

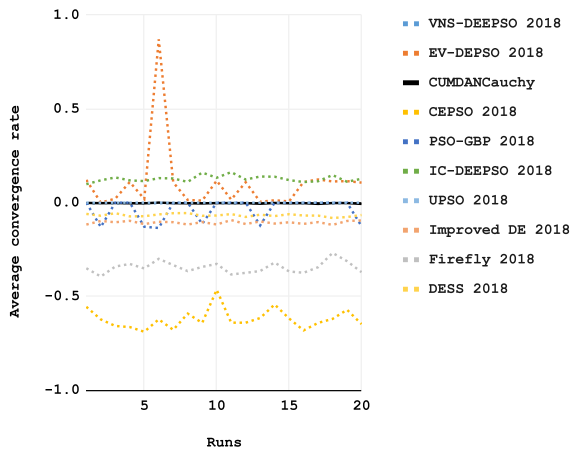

Fig. 3. Average convergence rate of CUMDANCauchy and the 2018 algorithms in case study 1.

Table 3
Runtime of CUMDANCauchy and the 2018 algorithms in case study 1.

Algorithms	Time (sec)
VNS-DEEPSO	246.47
EV-DEPSO	53.57
CUMDANCauchy	267.72
CEPSO	47.04
PSO-GBP	54.25
IC-DEEPSO	42.11
UPSO	19.82
Improved DE	45.66
Firefly	483.69
DESS	37.67

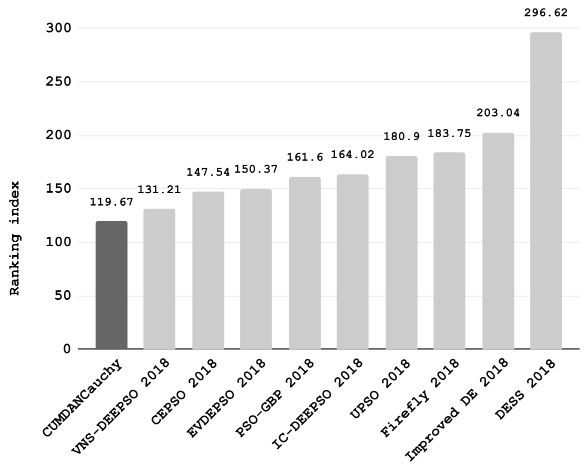

Fig. 4. Ranking index of CUMDANCauchy and the 2018 algorithms in case study 2.

On the other hand, it is noteworthy that the only approach that includes *VNS* (a hybrid approach including *PSO*, *DE*, and *VNS*), which incorporates problem-knowledge rules in the exploration of neighborhoods, achieves the first position.

Table 3 shows the average runtime of the 2018 algorithms in case study 1. *CUMDANCauchy*, had a similar runtime than *VNS-DEEPSO* (the best in terms of the ranking index) with a slight difference of approximately 22 sec, although for this case study, the best runtime was achieved by *UPSO* followed by *DESS*, *IC-DEEPSO*, *Improved DE*, *CEPSO*, *EV-DEPSO* and *PSO-GBP*.

Fig. 3 shows the average convergence rate for each one of the 20 independent executions of the 2018 algorithms in case study 1. Since the lesser the mean absolute value of the average convergence rate is, the better the convergence behavior of the algorithm is. *CUMDANCAUCHY* algorithm had an average convergence rate for each execution close to 0, which allows us to say that it has good convergence behavior towards the optimum. In general, *CUMDANCauchy*, had similar convergence behavior as the best algorithm in terms of the ranking index (*VNS-DEEPSO*). The algorithms that had the best convergence behavior were *UPSO* (0), *VNS-DEEPSO* (0.003523) and *CUMDANCauchy* (0.003166). While, the algorithms *CEPSO* (0.620479), *Firefly* (0.345522), *Improved DE* (0.106371) and *EV-DEPSO* (0.105558), had the worst convergence behavior.

6.2. Experiment (ii): set of the 2018 algorithms in case study 2

In this experiment, the same algorithms (with the same configuration) of the previous experiment were used, but for the case study 2.

Fig. 4 shows the results of this experiment in terms of ranking index. Without taking into account *CUMDANCauchy*, there is very little variation in the ranking index position order of the algorithms compared with the previous experiment. Again, the block of *PSO* variants overcomes the three approaches that do not use *PSO* to the last positions, with the approach that combines *PSO* with *VNS* at the top of the *PSO* block.

Nevertheless, in this case, *CUMDANCauchy* found the closest solution to the optimum and consequently achieved the first position in the ranking. This result shows the robustness of *CUMDANCauchy*, and how it can be applied to different case studies and still provide good performance to solve the ERM problem under uncertainty.

Table 4 shows the average runtime of the 2018 algorithms in case study 2. *CUMDANCauchy*, was the fastest algorithm, surpassing the algorithms *VNS-DEEPSO*, *PSO-GBP*, and *UPSO*. The slowest algorithms were *Firefly*, *ImprovedDE*, *IC-DEEPSO*, *EV-DEPSO*, *CEPSO*, and *DESS*.

Fig. 5 shows the average convergence rate for each one of the 20 independent executions of the 2018 algorithms in case study 2. Like in the previous experiment *CUMDANCauchy* had values close to 0, reaching good convergence behavior. Also, the convergence behavior of *CUMDANCauchy* was better than the convergence behavior of its closest competitors in terms of the ranking index (*VNS-DEEPSO* and *CEPSO*). The algorithms that had the best convergence behavior were *UPSO* (0), *EV-DEPSO* (0.196165), *PSO-GBP* (0.233317), *CUMDANCauchy* (0.296997) and *Firefly* (0.363178). While, the

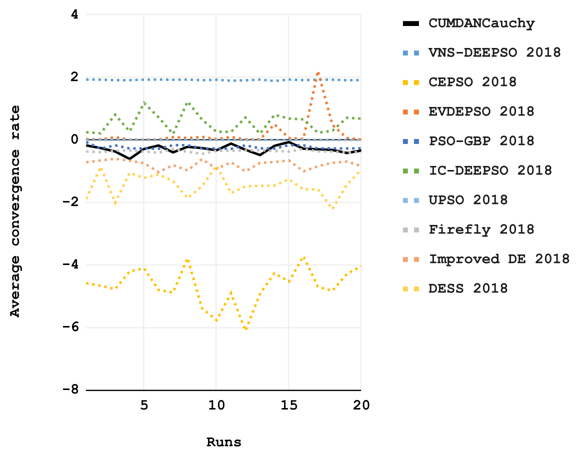

Fig. 5. Average convergence rate of CUMDANCauchy and the 2018 algorithms in case study 2.

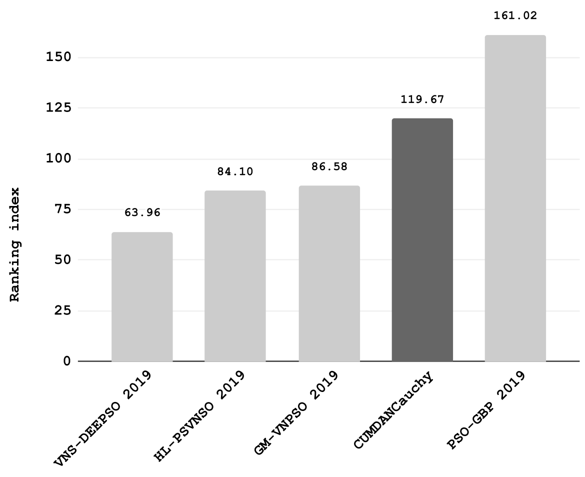

Fig. 6. Ranking index of CUMDANCauchy and the 2019 algorithms in case study 2.

Table 4
Runtime of CUMDANCauchy and the 2018 algorithms in case study 2.

Algorithms	Time (sec)
CUMDANCauchy	53.73
VNS-DEEPSO	116.85
CEPSO	311.06
EV-DEPSO	519.90
PSO-GBP	95.36
IC-DEEPSO	247.35
UPSO	151.90
Firefly	2790.29
Improved DE	786.05
DESS	400.55

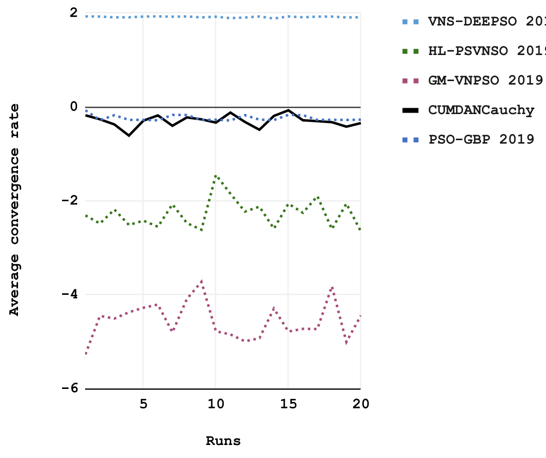

Fig. 7. Average convergence rate of CUMDANCauchy and the 2019 algorithms in case study 2.

algorithms $CEPSO$ (4.664714), $VNS\text{-}DEEPSO$ (1.919219), $DESS$ (1.439189), $Improved\ DE$ (0.78373) and $IC\text{-}DEEPSO$ (0.549923), had the worst convergence behavior.

6.3. Experiment (iii): set of the 2019 algorithms in case study 2

In this experiment, $CUMDANCauchy$ is compared with the set of the 2019 algorithms developed (and some of them updated) for the case study 2, i.e., Combination of Variable Neighborhood Search algorithm and Differential Evolutionary Particle Swarm Optimization ($VNS\text{-}DEEPSO$), Hybrid Levy Particle Swarm Variable Neighborhood Search Optimization ($HL\text{-}PSVNSO$), Gauss Mapped Variable Neighborhood Particle Swarm Optimization ($GM\text{-}VNPSO$) and Particle Swarm Optimization with Global Best Perturbation ($PSO\text{-}GBP$). Analogously to the previous experiment, $CUMDANCauchy$ used a population size = 10, max iterations = 499, and the number of scenarios used per fitness function evaluation = 10; while the 2019 algorithms used the defined parameters by their developers. The ranking index was computed from 20 independent executions of each algorithm.

Unlike the case study 1, in which PSO variants and non-PSO variants were developed, for the case study 2, there were only PSO variants and $CUMDANCauchy$. Again, the combination of PSO with VNS offered the best performance in terms of ranking index of the PSO variants block, but, in this case, the top three (out of 4) PSO variants were combined with the VNS (see Fig. 6). $CUMDANCauchy$, also divided the block of PSO variants, showing its competitiveness with the PSO variants. In fact, $CUMDANCauchy$ was only outperformed by the PSO variants that include VNS as an initial step in their implementations. Therefore, while the top three entries exploit problem-knowledge advantages provided by rules in VNS, $CUMDANCauchy$ is an approach that does not depend on this type of information; making it an approach with competitive performance despite the case study to be solved.

Table 5
Runtime of CUMDANCauchy and the 2019 algorithms in case study 2.

Algorithms	Time Speed(sec)
VNS-DEEPSO	116.85
HL-PSVNSO	101.10
GM-VNPSO	103.20
CUMDANCauchy	53.73
PSO-GBP	95.36

Table 5 shows the average runtime of the 2019 algorithms in case study 2. Again $CUMDANCauchy$ was the fastest algorithm. This behavior of $CUMDANCauchy$ was also described by an independent research (Dabhi and Pandya, 2020b) that use the available online implementation of $CUMDANCauchy$.

Fig. 7 shows the average convergence rate for each one of the 20 independent executions of the 2019 algorithms in case study 2. Once again, the convergence behavior of $CUMDANCauchy$ was among the best. It was only surpassed by the worst algorithm in terms of ranking index ($PSO\text{-}GBP$).

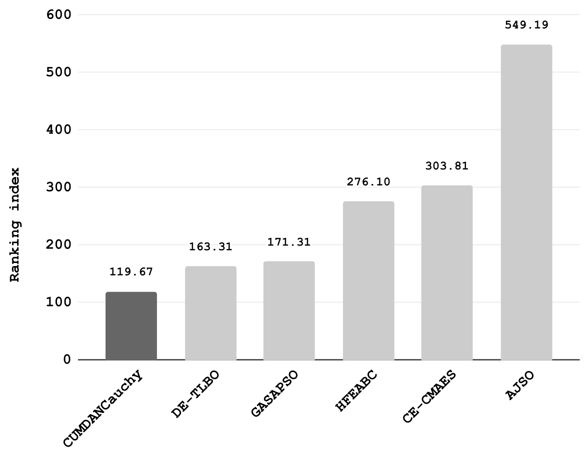

Fig. 8. Ranking index of CUMDANCauchy and the 2020 algorithms in case study 2.

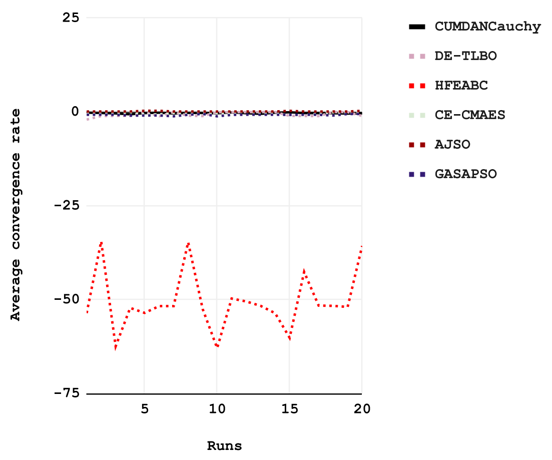

Fig. 9. Average convergence rate of CUMDANCauchy and the 2020 algorithms in case study 2.

Table 6
Runtime of CUMDANCauchy and the 2020 algorithms in case study 2.

Algorithms	Time (sec)
$CUMDANCauchy$	53.73
DE-$TLBO$	839.49
$GASAPSO$	607.89
$HFEABC$	178.77
CE-$CMAES$	1430.99
$AJSO$	404.76

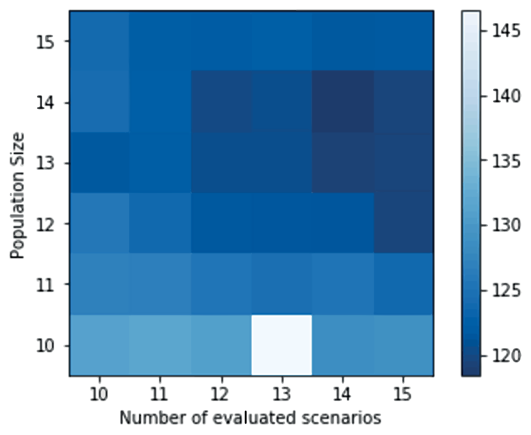

Fig. 10. Average Ranking Index for 1000 independent executions of $CUMDANCauchy$.

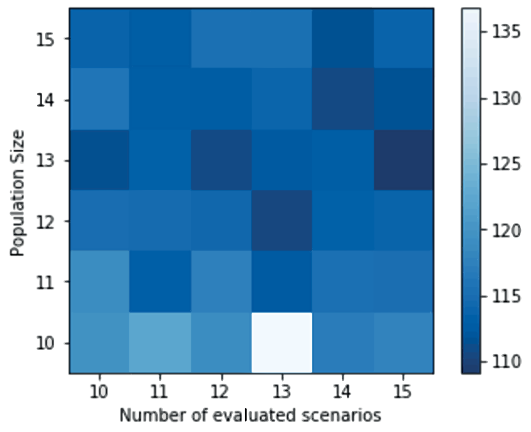

Fig. 11. Minimum Ranking Index for 1000 independent executions of $CUMDANCauchy$.

6.4. Experiment (iv): set of the 2020 algorithms in case study 2

After selecting the algorithms of 2020, which were adaptable to case study 2 (i.e. Enhanced Artificial Bee Colony ($HFEAB$), Cross Entropy Covariance Matrix Adaptation Evolution Strategy (CE-$CMAES$), Differential Evolution and Teaching–Learning-Based Optimization (DE-$TLBO$), Advanced JSO ($AJSO$) and Genetic Algorithm, Simulated Annealing and Particle Swarm Optimization ($GASAPSO$)), the experiments were carried out to compare them with $CUMDANCauchy$.

In this experiment, $CUMDANCauchy$ reached the best ranking index (see Fig. 8), and it was the fastest of the algorithms (see Table 6). Also, like in the previous experiment, the convergence behavior of $CUMDANCauchy$ was among the best (see Fig. 9). The convergence behavior of $CUMDANCauchy$ was only surpassed by the worst algorithm in terms of the ranking index ($AJSO$).

6.5. Experiment (v): $CUMDANCauchy$ sensitivity to its parameter configuration

In this experiment, the performance of $CUMDANCauchy$ in the case study 2 is explored by changing its configuration parameters: population size from 10 to 16 and the number of scenarios used per fitness function evaluation from 10 to 16. For each pair of parameters, 1000 independent full executions of the algorithm were done. We set the number of iterations to the maximum allowed, which is computed as $\frac{maxEval}{N_p \times N_{eval}}$ where $maxEval$, N_p and N_{eval} are the maximum number fitness function evaluation defined by the framework (50000), the population size and the number of scenarios used per fitness function evaluation, respectively.

Figs. 10, 11, 12 and 13 show the Average Ranking Indexes, Minimum Ranking Indexes, Maximum Ranking Indexes and Standard Deviation of the Ranking Indexes respectively for 1000 executions of $CUMDANCauchy$.

Fig. 10 shows that the Average Ranking Index moves in the range $[118.41, 131.69]$, which represents a variation of only 10.08%. Also, it can be seen as a general pattern: the best Average Ranking Index is obtained close to the configuration parameters population size=14 and the number of scenarios =14.

Unlike the above results, when analyzing the behavior of the best (Fig. 11) and worst (Fig. 12) Ranking Index of the 1000 independent

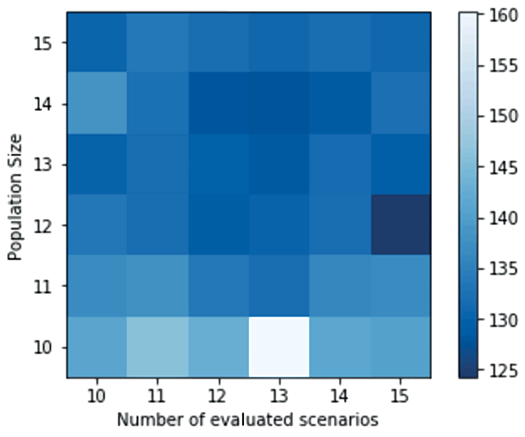

Fig. 12. Maximum Ranking Index for 1000 independent executions of $CUMDANCauchy$.

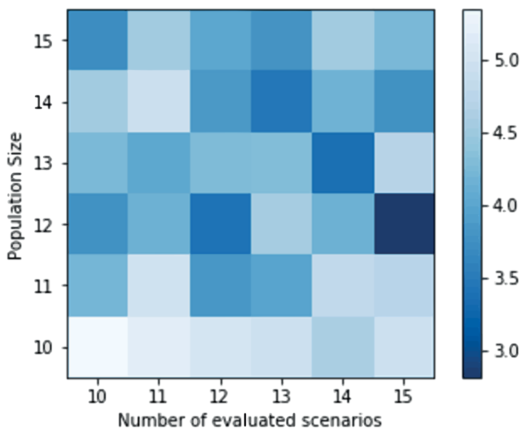

Fig. 13. Standard Deviation of the Ranking Index for 1000 independent executions of $CUMDANCauchy$.

executions of $CUMDANCauchy$ for each pair of configuration parameters, no pattern was found. The minimum and maximum Ranking Indices were 109.15 and 160.15, respectively.

Analogously, no pattern was found when analyzing the standard deviation for each pair of parameters (Fig. 13). The standard deviation was bounded by the range $[2.81, 5.34]$. The highest standard deviation, 5.34, represents, in the worst case, a variation of 4.90% for the Ranking Index.

Also, it is important to highlight that for the best Average Ranking Index, the standard deviation was 4.17, which represents only a variation of 3.51%

6.6. Statistical validation

In this section, a statistical validation is performed, taking into account the results of the runtime and the global fitness for algorithms of 2018, 2019, and 2020 in the case study 2. Different statistical tests were performed to test the normality of the data (Shapiro–Wilks Shapiro et al., 1968; Kuranga et al., 2020), ranking of the algorithms (Friedman Zimmerman and Zumbo, 1993; Benavoli et al., 2016), and analysis of significant differences between algorithms (Wilcoxon Wilcoxon et al., 1970; Koutras and Triantafyllou, 2020).

6.6.1. Analysis of runtime

In this section, the statistical analysis of the competition's algorithms for case study 2 is performed. For this study, the runtime of 20 independent executions for each algorithm is compared.

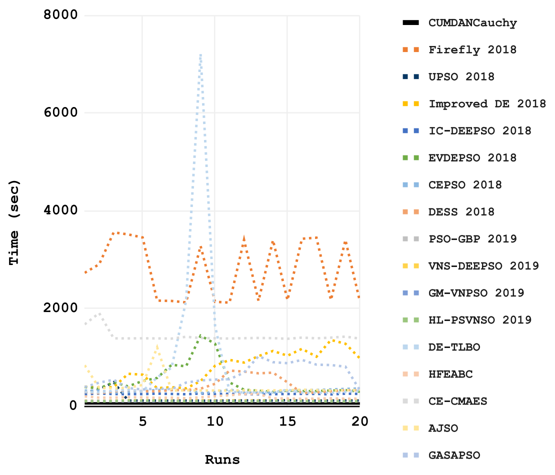

Fig. 14. Runtime of all algorithms in case study 2.

Table 7
Results of Shapiro–Wilks test for the runtime of all algorithms in case study 2.

Algorithms	**W**	$p-value$	**Hypothesis**
$CUMDANCauchy$	0.92605	0.1296	Accept
$Firefly$	0.7583	0.0002198	Reject
$UPSO$	0.46917	1.69E-07	Reject
$Improved\ DE$	0.90065	0.04242	Reject
$IC\text{-}DEEPSO$	0.97181	0.7926	Accept
$EVDEPSO$	0.70535	4.53E-05	Reject
$CEPSO$	0.94023	0.2421	Accept
$DESS$	0.73527	0.0001084	Reject
$PSO\text{-}GBP$	0.97951	0.9276	Accept
$VNS\text{-}DEEPSO$	0.91766	0.08932	Accept
$GM\text{-}VNPSO$	0.8675	0.01062	Reject
$HL\text{-}PSVNSO$	0.83135	0.002637	Reject
$DE\text{-}TLBO$	0.41081	5.46E-08	Reject
$HFEABC$	0.83267	0.002768	Reject
$CE\text{-}CMAES$	0.43142	8.07E-08	Reject
$AJSO$	0.44959	1.15E-07	Reject
$GASAPSO$	0.89514	0.03346	Reject

As shown in Fig. 14, $CUMDANCauchy$ had the best runtime concerning the rest of the algorithms. To statistically validate this result, the Shapiro–Wilks test is applied to see if the behavior of each algorithm was normal. The results are shown in Table 7. The algorithms $Firefly$, $UPSO$, $Improved\ DE$, $EVDEPSO$, $DESS$, $GM\text{-}VNPSO$, $HL\text{-}PSVNSO$, $DE\text{-}TLBO$, $HFEABC$, $CE\text{-}CMAES$, $AJSO$, and $GASAPSO$, do not meet the normality criterion, because $p\text{-}value < 0.05$. On the other hand, the $IC\text{-}DEEPSO$, $CEPSO$, $CUMDANCauchy$, $PSO\text{-}GBP$, and $VNS\text{-}DEEPSO$ have normal behavior because $p\text{-}value > 0.05$. As there are algorithms whose behavior is not normal, a non-parametric statistical comparison of them must be performed.

Thus, it is proceeded to perform the non-parametric Friedman test for multiple comparisons of the algorithms, where the following results were obtained (see Table 8). As can be observed in Table 8, $CUMDANCauchy$ was in the first place (1), followed by $PSO-GBP$ (2.15) and $HL\text{-}PSVNSO$ (3.2). To remark, the algorithm with the best global fitness ($VNS\text{-}DEEPSO$), was in fifth place.

The Iman–Davenport's test is carried out (employing F distribution of 16 degrees of freedom for $Nds = 297.28$), in order to find statistical

Table 8
Ranking of runtime of all algorithms in case study 2.

Algorithms	Ranking
CUMDANCauchy	1.00
PSO-GBP	2.15
HL-PSVNSO	3.20
GM-VNPSO	3.75
VNS-DEEPSO	5.80
UPSO	5.85
HFEABC	6.95
IC-DEEPSO	7.75
CEPSO	10.45
DESS	10.85
DE-TLBO	11.25
AJSO	11.70
EVDEPSO	12.35
GASAPSO	13.15
Improved DE	14.10
CE-CMAES	15.80
Firefly	16.90

Table 9
Results of the Wilcoxon test for the runtime of all algorithms in case study 2.

CUMDANCauchy vs	R+	R-	$p-value$	Hypothesis
CUMDANCauchy	210	0	1.00E+00	Accept
Firefly	0	210	4.43E-05	Reject
UPSO	0	210	4.43E-05	Reject
Improved DE	0	210	4.43E-05	Reject
IC-DEEPSO	0	210	4.43E-05	Reject
EVDEPSO	0	210	4.43E-05	Reject
CEPSO	0	210	4.43E-05	Reject
DESS	0	210	4.43E-05	Reject
PSO-GBP	0	210	4.43E-05	Reject
VNS-DEEPSO	0	210	4.43E-05	Reject
GM-VNPSO	0	210	4.43E-05	Reject
HL-PSVNSO	0	210	4.43E-05	Reject
DE-TLBO	0	210	4.43E-05	Reject
HFEABC	0	210	4.43E-05	Reject
CE-CMAES	0	210	4.43E-05	Reject
AJSO	0	210	4.43E-05	Reject
GASAPSO	0	210	4.43E-05	Reject

differences among the algorithms, obtaining a *p-value* of $2.2E-15$. The results of the Iman–Davenport's shows that *CUMDANCauchy* presents indeed significant performance differences in the group (*p-value* $<$ 0.05).

To analyze if there are significant differences between the algorithms, the Wilcoxon test was performed. The results obtained for the test are shown in Table 9. There are significant differences between *CUMDANCauchy* and the other algorithms since *p-value* $<$ 0.05. Therefore, *CUMDANCauchy* is significantly superior to the other algorithms in terms of runtimes.

6.6.2. Analysis of global fitness

In this test, the global fitness of 20 independent executions of each algorithm in the case study 2 are used (see Fig. 15). The algorithms *VNS-DEEPSO*, *HL-PSVNSO*, *GM-VNPSO* and *CUMDANCauchy* had the best global fitness, while the *AJSO*, *CE-CMAES*, *DESS*, *HFEABC* and *Improved DE* the worst. On the other

Table 10
Results of Shapiro–Wilks test of global fitness of all algorithms in case study 2.

Algorithms	W	$p-value$	Hypothesis
CUMDANCauchy	0.86407	0.00926	Reject
Firefly	0.92182	0.1074	Accept
UPSO	0.9682	0.7166	Accept
Improved DE	0.93758	0.2158	Accept
IC-DEEPSO	0.92916	0.1488	Accept
EVDEPSO	0.77524	0.0003777	Reject
CEPSO	0.95472	0.4445	Accept
DESS	0.97403	0.8366	Accept
PSO-GBP	0.90191	0.0448	Reject
VNS-DEEPSO	0.52072	4.92E-07	Reject
GM-VNPSO	0.96725	0.696	Accept
HL-PSVNSO	0.81694	0.001563	Accept
DE-TLBO	0.87052	0.012	Reject
HFEABC	0.78468	0.000515	Reject
CE-CMAES	0.7593	2.27E-04	Reject
AJSO	0.90803	0.05847	Accept
GASAPSO	0.6849	2.57E-05	Reject

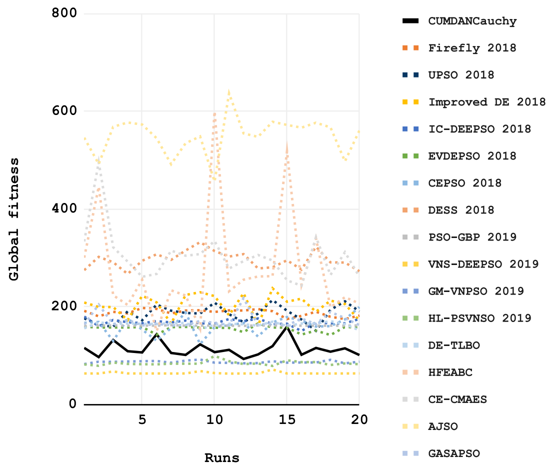

Fig. 15. Global fitness of all algorithms in case study 2.

hand, *EVDEPSO*, *PSO-GBP*, *CEPSO*, *DE-TLBO*, *IC-DEEPSO*, *GASAPSO*, *UPSO* and *Firefly* had a relatively good global fitness.

To statistically validate this result, the Shapiro–Wilks test is applied to see if the behavior of each algorithm was normal. The results are shown in Table 10.

The algorithms *CUMDANCauchy*, *EVDEPSO*, *PSO-GBP*, *VNS-DEEPSO*, *DE-TLBO*, *HFEABC*, *CE-CMAES*, and *GASAPSO* do not meet the normality criterion, since *p-value* $<$ 0.05. On the other hand, the algorithms *Firefly*, *UPSO*, *Improved DE IC-DEEPSO*, *CEPSO*, *DESS*, *GM-VNPSO*, *HL-PSVNSO*, and, *AJSO*, have normal behavior, since *p-value* $>$ 0.05. As there are algorithms whose behavior is not normal, non-parametric tests had to be performed for statistical comparison of the algorithms.

The Friedman test is performed, and the algorithms are ranked. The results are shown in Table 11. *VNS-DEEPSO* was in the first place (1), followed by *GM-VNPSO* (2.25) and *HL-PSVNSO* (2.75). In this test, *CUMDANCauchy* (4.2) was ranked in fourth place. This rank position confirms the excellent behavior of *CUMDANCauchy*, being among the first five algorithms.

Also, the Iman–Davenport's test is carried out (employing F distribution of 16 degrees of freedom for $Nds = 292.75$), in order to find statistical differences among the algorithms, obtaining a *p-value*

Table 11
Ranking of global fitness of all algorithms in case study 2.

Algorithms	Ranking
VNS-DEEPSO	1.00
GM-VNPSO	2.25
HL-PSVNSO	2.75
CUMDANCauchy	4.20
EVDEPSO	5.65
PSO-GBP	7.05
CEPSO	7.75
DE-TLBO	7.95
IC-DEEPSO	9.30
GASAPSO	9.35
UPSO	10.90
Firefly	11.65
Improved DE	12.00
HFEABC	13.65
DESS	15.25
CE-CMAES	15.40
AJSO	16.90

Table 12
Results of the Wilcoxon test for the global fitness of all algorithms in case study 2.

CUMDANCauchy vs	R+	R-	$p-value$	Hypothesis
CUMDANCauchy	210	0	1.00E+00	Accept
Firefly	0	210	4.43E-05	Reject
UPSO	0	210	4.43E-05	Reject
Improved DE	0	210	4.43E-05	Reject
IC-DEEPSO	39	181	6.87E-03	Reject
EVDEPSO	0	210	4.43E-05	Reject
CEPSO	0	210	4.43E-05	Reject
DESS	0	210	4.43E-05	Reject
PSO-GBP	0	210	4.43E-05	Reject
VNS-DEEPSO	0	210	4.43E-05	Reject
GM-VNPSO	0	210	4.43E-05	Reject
HL-PSVNSO	0	210	4.43E-05	Reject
DE-TLBO	0	210	4.43E-05	Reject
HFEABC	0	210	4.43E-05	Reject
CE-CMAES	0	210	4.43E-05	Reject
AJSO	0	210	4.43E-05	Reject
GASAPSO	0	210	4.43E-05	Reject

of $2.22E-10$. The results of the Iman–Davenport's tests show that first place presents indeed significant performance differences in the group (p-$value < 0.05$).

Finally, to analyze if there are significant differences in terms of the global fitness between the algorithms, the Wilcoxon test is performed. The results obtained are shown in Table 12. There are significant differences between $CUMDANCauchy$ and the rest of the algorithms presented since the p-$value < 0.05$.

From the statistical validation, it can be concluded that $CUMDANCauchy$ presents significant differences with respect to the other algorithms and excellent behavior in terms of global fitness. Also, it is significantly superior to all the algorithms in terms of the runtime.

7. Conclusions

In this paper, a novel algorithm, $CUMDANCauchy$, was proposed to solve the ERM problem under uncertainty. $CUMDANCauchy$ is a cellular evolutionary algorithm that uses a ring as the structure of the neighborhood. It also uses the product of Normal and Cauchy distribution to estimate the probability distribution of the individuals. Also, this work provides an overview of the ERM problem and the state-of-the-art algorithms proposed to solve it, as well as the developed frameworks to perform testing and experiments.

The performance of $CUMDANCauchy$ had evaluated in two available case studies (i.e., the 2018 and 2019 frameworks), and compared with the set of algorithms submitted and openly available, to 2018, 2019, and 2020 editions of the "Competition on Evolutionary Computation in the Energy Domain: Smart Grid Applications".

The results showed that $CUMDANCauchy$ is ranked third in terms of the global fitness achieved to solve the ERM problem under uncertainty when testing the algorithms that participated in the 2018 competition for the first case study, while it can achieve the 1st rank when such algorithms are tested in the second case study. This performance provides some evidence about the robustness and effectiveness of the winner approaches under different case studies. Besides, $CUMDANCauchy$ was ranked fourth compared to the algorithms developed in 2019. Overall, $CUMDANCauchy$ was competitive against the PSO dominant variants proposed to solve the ERM problem under uncertain environment and overcome many of the other proposed algorithms. $CUMDANCauchy$ was only outperformed by PSO variants, including VNS, an algorithm that exploits specific knowledge of the

Table A.13
Nomenclature of indices, variables and parameters.

Indices	
i	Distributed generation (DG) units
j	PV units
k	External suppliers
e	Energy storage systems (ESSs)
v	Electric vehicles (EVs)
l	loads (EVs)
m	Markets
s	Scenarios
t	Periods
Variables	
P_{DG}	Active power generation (kW)
P_{ext}	External supplied power (kW)
P_{ESS^-}	Discharge power of ESS (kW)
P_{EV^-}	Discharge power of EV (kW)
P_{ESS^+}	Charge power of ESS (kW)
P_{EV^+}	Charge power of EV (kW)
P_{curt}	Power reduction of load (kW)
P_{imb^-}	Non-supplied power for load (kW)
P_{imb^+}	Exceeded power of DG unit (kW)
P_{buy}	Power buy to the market (kW)
P_{sell}	Power sell to the market (kW)
X_{DG}	Binary variable for DG status
Parameters	
N_{DG}	Number of DG
N_{PV}	Number of PV
N_k	Number of external suppliers
N_e	Number of ESSss
N_v	Number of EVs
N_L	Number of loads
N_m	Number of market
N_s	Number of scenario
T	Number of periods
C_{DG}	Generation cost of DG (m.u./kWh)
C_{ext}	Cost of external supplier (m.u./kWh)
C_{pv}	Cost of PV generation (m.u./kWh)
C_{ESS^-}	Discharging cost of ESS (m.u./kWh)
C_{EV^-}	Discharging cost of EV (m.u./kWh)
C_{curt}	Load curtailment cost (m.u./kWh
C_{imb}	Grid imbalance cost (m.u./kWh)
$\pi(s)$	Probability of scenario s
P_{PV}	Photovoltaic generation (kW)
P_{load}	Forecasted load
MP	Electricity market price (m.u./kWh)

problem, as an initial step. Also, $CUMDANCauchy$ was ranked first compared to the algorithms developed in 2020 for case study 2. Experiments also showed that for the configuration parameters explored, the variance of the results for 1000 independent executions is low, demonstrating the robustness of $CUMDANCauchy$.

Besides, the convergence behavior of $CUMDANCauchy$ was among the best in comparison with the other algorithms. Also, in terms of runtime, $CUMDANCauchy$ was similar to the algorithm with the best global fitness for one case study, and it was the fastest for the other case study.

Finally, the statistical validation reveals that $CUMDANCauchy$ presents significant differences with respect to the other algorithms and excellent behavior in terms of global fitness. Also, it is significantly superior to all the algorithms in terms of the runtime.

As future work, two different venues can be explored. The first venue, regarding the development of efficient algorithms, we visualize the study of hybrid approaches (for instance, combining VNS with $CUMDANCauchy$) and the use of new types of neighborhoods as interesting lines of research worth to be explored. Another interesting work is developing a cellular EDA that learns the dependencies amongst the variables involved in the ERM problem under uncertainty. In the second venue, regarding the practical implementation of ERM systems using evolutionary algorithms, this work has considered several assumptions that cannot be neglected in real-world environments. Therefore, future work should pursue the implementation of these tools to solve real-world problems, consider realistic constraints and scenarios, and remove as many assumptions as possible to deal with the control and operation of distributed resources in distribution networks.

CRediT authorship contribution statement

Yoan Martínez-López: Software, Formal analysis, Writing - original draft. **Ansel Y. Rodríguez-González:** Conceptualization, Visualization, Writing - original draft, Writing - review & editing, Supervision. **Julio Madera:** Writing - review & editing, Resources, Supervision. **Miguel Bethencourt Mayedo:** Software, Validation. **Fernando Lezama:** Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partially supported by Consejo Nacional de Ciencia y Tecnología (CONACYT) México and the VLIR-UOS Network University Cooperation Programme - Cuba. Also, this work has received partial funding from Fundo Europeu de Desenvolvimento Regional (FEDER), Portugal Funds through COMPETE program and from National Funds through Fundação para a Ciência e a Tecnologia (FCT), Portugal under the projects UIDB/00760/2020 and MAS-Society, Portugal (PTDC/EEI-EEE/28954/2017).

Appendix. Nomenclature

The nomenclature is shown in Table A.13.

References

Abbass, H.A., 2002. The self-adaptive Pareto differential evolution algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), vol. 1, pp. 831–836.

Alba, E., Madera, J., Dorronsoro, B., Ochoa, A., Soto, M., 2006. Theory and practice of cellular UMDA for discrete optimization. In: Parallel Problem Solving from Nature-PPSN IX. Springer, pp. 242–251.

Ali, I.M., Essam, D., Kasmarik, K., 2019. A novel differential evolution mapping technique for generic combinatorial optimization problems. Appl. Soft Comput. 80, 297–309.

Awad, N.H., Ali, M.Z., Mallipeddi, R., Suganthan, P.N., 2019. An efficient differential evolution algorithm for stochastic OPF based active–reactive power dispatch problem considering renewable generators. Appl. Soft Comput. 76, 445–458.

Bazionis, I.K., Georgilakis, P.S., 2021. Review of deterministic and probabilistic wind power forecasting: Models, methods, and future research. Electricity 2 (1), 13–47.

Benavoli, A., Corani, G., Mangili, F., 2016. Should we really use post-hoc tests based on mean-ranks? J. Mach. Learn. Res. 17 (1), 152–161.

Biswas, P.P., Suganthan, P., Mallipeddi, R., Amaratunga, G.A., 2019. Optimal reactive power dispatch with uncertainties in load demand and renewable energy sources adopting scenario-based approach. Appl. Soft Comput. 75, 616–632.

Brest, J., Maučec, M.S., Bošković, B., 2017. Single objective real-parameter optimization: Algorithm jSO. In: 2017 IEEE Congress on Evolutionary Computation. CEC. IEEE, pp. 1311–1318.

Brown, R.E., Impact of smart grid on distribution system design. In: 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, pp. 1–4.

Brucker, P., Knust, S., Cheng, T.E., Shakhlevich, N.V., 2004. Complexity results for flow-shop and open-shop scheduling problems with transportation delays. Ann. Oper. Res. 129, 81–106.

Chen, S., Abdulselam, I., Yadollahpour, N., Gonzalez-Fernandez, Y., 2020. Particle swarm optimization with pbest perturbations. In: 2020 IEEE Congress on Evolutionary Computation. CEC. IEEE, pp. 1–8.

Chen, Y., He, J., 2019. Average convergence rate of evolutionary algorithms II: Continuous optimisation. In: International Symposium on Intelligence Computation and Applications. Springer, pp. 31–45.

Clark II, W.W., Zipkin, T., Bobo, S., Rong, M., 2017. Global changes in energy systems: Central power and on-site distributed. In: Agile Energy Systems: Global Distributed On-Site and Central Grid Power. Elsevier, p. 61.

Dabhi, D., Pandya, K., 2020a. Enhanced velocity differential evolutionary particle swarm optimization for optimal scheduling of a distributed energy resources with uncertain scenarios. IEEE Access 8, 27001–27017.

Dabhi, D., Pandya, K., 2020b. Uncertain scenario based microgrid optimization via hybrid levy particle swarm variable neighborhood search optimization (HL_PS_VNSO). IEEE Access 8, 108782–108797.

Dou, Y., Xiong, H., 2017. Research on recognition of medical insurance fraud based on modified support vector machine. In: 2017 International Conference on Computer Technology, Electronics and Communication. ICCTEC. IEEE, pp. 1021–1025.

Garcia-Guarin, J., Alvarez, D., Bretas, A., Rivera, S., 2020. Schedule optimization in a smart microgrid considering demand response constraints. Energies 13 (17).

Garcia-Guarin, J., Rodriguez, D., Alvarez, D., Rivera, S., Cortes, C., Guzman, A., Bretas, A., Aguero, J.R., Bretas, N., 2019. Smart microgrids operation considering a variable neighborhood search: The differential evolutionary particle swarm optimization algorithm. Energies 12 (16).

Gelazanskas, L., Gamage, K.A., 2014. Demand side management in smart grid: A review and proposals for future direction. Sustainable Cities Soc. 11, 22–30.

Gholami, K., Dehnavi, E., 2019. A modified particle swarm optimization algorithm for scheduling renewable generation in a micro-grid under load uncertainty. Appl. Soft Comput. 78, 496–514.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, first ed. Addison-Wesley Longman Publishing Co., Inc., USA.

Hadi, A.A., Mohamed, A.W., Jambi, K.M., 2021. Single-objective real-parameter optimization: Enhanced LSHADE-spacma algorithm. In: Heuristics for Optimization and Learning. Springer, pp. 103–121.

Halim, A.H., Ismail, I., Das, S., 2020. Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. Artif. Intell. Rev. 1–87.

He, J., Lin, G., 2015. Average convergence rate of evolutionary algorithms. IEEE Trans. Evol. Comput. 20 (2), 316–321.

Hyde, S., Ninham, B., Andersson, S., Larsson, K., Landh, T., Blum, Z., Lidin, S., 1997. The mathematics of curvature. Lang. Shape 1–42.

Jiang, X., Zhou, J., 2013. Hybrid DE-TLBO algorithm for solving short term hydro-thermal optimal scheduling with incommensurable objectives. In: Proceedings of the 32nd Chinese Control Conference, 2474–2479.

Kabirifar, M., Pourghaderi, N., Rajaei, A., Moeini-Aghtaie, M., Safdarian, A., 2020. Deterministic and probabilistic models for energy management in distribution systems. In: Handbook of Optimization in Electric Power Distribution Systems. Springer, pp. 343–382.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization (PSO). In: Proc. IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948.

Kim, S., Hur, J., 2020. Probabilistic approaches to the security analysis of smart grid with high wind penetration: The case of Jeju Island's power grids. Energies 13 (21), 5785.

Kohler, M., Vellasco, M.M., Tanscheit, R., 2019. PSO+: A new particle swarm optimization algorithm for constrained problems. Appl. Soft Comput. 85, 105865.

Koutras, M.V., Triantafyllou, I.S., 2020. Wilcoxon-type rank-sum statistics for selecting the best population: Some advances. Appl. Stoch. Models Bus. Ind.

Kuranga, J.O., Ayinde, K., Solomon, G.S., 2020. Empirical investigation of type 1 error rate of some normality test statistics. Int. J. Psychosoc. Rehabil. 24 (4).

Larrañaga, P., Lozano, J.A., 2001. Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, vol. 2. Springer Science & Business Media.

Lenstra, J.K., Rinnooy Kan, A., 1978. Complexity of scheduling under precedence constraints. Oper. Res. 26 (1), 22–35.

Leung, J.Y., 2004. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC press.

Lezama, F., Soares, J., Hernandez-Leal, P., Kaisers, M., Pinto, T., Vale, Z., 2018a. Local energy markets: Paving the path towards fully transactive energy systems. IEEE Trans. Power Syst.

Lezama, F., Soares, J., Vale, Z., 2018b. A platform for testing the performance of metaheuristics solving the energy resource management problem in smart grids. Energy Inform. 1 (1), 35.

Lezama, F., Soares, J., Vale, Z., Rueda, J., 2018c. Call for competition on evolutionary computation in uncertain environments: A smart grid application. http://www.gecad.isep.ipp.pt/WCCI2018-SG-COMPETITION/. (Accessed 10 February 2021).

Lezama, F., Soares, J., Vale, Z., Rueda, J., 2019a. Call for competition on evolutionary computation in uncertain environments: A smart grid application. http://www.gecad.isep.ipp.pt/ERM2019-Competition/. (Accessed 10 February 2021).

Lezama, F., ao Soares, J., Vale, Z., Rueda, J., Rivera, S., Elrich, I., 2019b. 2017 IEEE competition on modern heuristic optimizers for smart grid operation: Testbeds and results. Swarm Evol. Comput. 44, 420–427.

Ma, Z., Yuan, X., Han, S., Sun, D., Ma, Y., 2019. Improved chaotic particle swarm optimization algorithm with more symmetric distribution for numerical function optimization. Symmetry 11 (7).

Madera, J., Alba, E., Ochoa, A., 2006. A parallel island model for estimation of distribution algorithms. In: Towards a New Evolutionary Computation. Springer, pp. 159–186.

Martínez-López, Y., Madera, J., Rodríguez-González, A.Y., Barigye, S., 2019a. Cellular estimation Gaussian algorithm for continuous domain. J. Intell. Fuzzy Systems 36 (5), 4957–4967.

Martínez-López, Y., Rodríguez-González, A.Y., Quintana, J.M., Moya, A., Morgado, B., Mayedo, M.B., 2019b. CUMDANCauchy-C1: a cellular EDA designed to solve the energy resource management problem under uncertainty. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM, pp. 13–14.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.

Mühlenbein, H., Voigt, H.-M., 1996. Gene pool recombination in genetic algorithms. In: Meta-Heuristics. Springer, pp. 53–62.

Palakonda, V., Awad, N.H., Mallipeddi, R., Ali, M.Z., Veluvolu, K.C., Suganthan, P.N., 2018. Differential evolution with stochastic selection for uncertain environments: A smart grid application. In: 2018 IEEE Congress on Evolutionary Computation. CEC. pp. 1–7.

Pant, M., Thangaraj, R., Abraham, A., Grosan, C., 2009. Differential evolution with Laplace mutation operator. In: 2009 IEEE Congress on Evolutionary Computation, pp. 2841–2849.

Parsopoulos, K., Vrahatis, M., 2019. UPSO: A Unified Particle Swarm Optimization Scheme. pp. 868–873.

Pelikan, M., Mühlenbein, H., 1998. Marginal distributions in evolutionary algorithms. In: Proceedings of the International Conference on Genetic Algorithms Mendel, vol. 98. Citeseer, pp. 90–95.

Price, K.V., 2013. Differential evolution. In: Handbook of Optimization: From Classical To Modern Approach. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 187–214.

Qiu, Y., Wang, L., Xu, X., Fang, X., Pardalos, P.M., 2018. A variable neighborhood search heuristic algorithm for production routing problems. Appl. Soft Comput. 66, 311–318.

Rao, R., 2016. Review of applications of TLBO algorithm and a tutorial for beginners to solve the unconstrained and constrained optimization problems. Decis. Sci. Lett. 5 (1), 1–30.

Sarda, J., Pandya, K., Shah, M., 2019. Emerging heuristic optimization algorithms for expansion planning and flexibility optimization. In: Advances in Control Systems and its Infrastructure: Proceedings of ICPCCI 2019, vol. 604. Springer Nature, p. 191.

Shapiro, S.S., Wilk, M.B., Chen, H.J., 1968. A comparative study of various tests for normality. J. Amer. Statist. Assoc. 63 (324), 1343–1372.

Shen, Y., Liang, Z., Kang, H., Sun, X., Chen, Q., 2021. A modified jSO algorithm for solving constrained engineering problems. Symmetry 13 (1), 63.

Soares, J., Canizes, B., Ghazvini, M.A.F., Vale, Z., Venayagamoorthy, G.K., 2017. Two-stage stochastic model using benders' decomposition for large-scale energy resource management in smart grids. IEEE Trans. Ind. Appl. 53 (6), 5905–5914.

Soares, J., Ghazvini, M.A.F., Silva, M., Vale, Z., 2016. Multi-dimensional signaling method for population-based metaheuristics: Solving the large-scale scheduling problem in smart grids. Swarm Evol. Comput. 29, 13–32.

Soares, J., Sousa, T., Morais, H., Vale, Z., Canizes, B., Silva, A., 2013. Application-specific modified particle swarm optimization for energy resource scheduling considering vehicle-to-grid. Appl. Soft Comput. 13 (11), 4264–4280.

Storn, R., Price, K., 1997. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. 11 (4), 341–359.

Strickler, A., Castro, O., Pozo, A., Santana, R., 2018. An investigation of the selection strategies impact on MOEDAs: CMA-ES and UMDA. Appl. Soft Comput. 62, 963–973.

Sun, Y., Qi, G., Wang, Z., Van Wyk, B.J., Hamam, Y., 2009. Chaotic particle swarm optimization. In: Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 505–510.

Thangraj, R., Pant, M., Abraham, A., Deep, K., Snasel, V., 2010. Differential evolution using a localized Cauchy mutation operator. In: 2010 IEEE International Conference on Systems, Man and Cybernetics, pp. 3710–3716.

Tuo, S., Geem, Z.W., Yoon, J.H., 2020. A new method for analyzing the performance of the harmony search algorithm. Mathematics 8 (9), 1421.

Wang, C.-F., Song, W.-X., 2019. A novel firefly algorithm based on gender difference and its convergence. Appl. Soft Comput. 80, 107–124.

Wilcoxon, F., Katti, S., Wilcox, R.A., 1970. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. Sel. Tables Math. Stat. 1, 171–259.

Xiaogang, F., Jingshou, Y., 2009. A hybrid algorithm based on extremal optimization with adaptive levy mutation and differential evolution and application. In: 2009 Fifth International Conference on Natural Computation, vol. 1, pp. 12–16.

Yan, Y., Qian, Y., Sharif, H., Tipper, D., 2013. A survey on smart grid communication infrastructures: Motivations, requirements and challenges. IEEE Commun. Surv. Tutor. 15 (1), 5–20.

Yang, X.-S., 2009. Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T. (Eds.), Stochastic Algorithms: Foundations and Applications. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 169–178.

Yang, X.-S., He, X., 2013. Firefly algorithm: recent advances and applications. Int. J. Swarm Intell. 1 (1), 36–50.

Yu, W., Li, X., Cai, H., Zeng, Z., Li, X., 2018. An improved artificial bee colony algorithm based on factor library and dynamic search balance. Math. Probl. Eng. 2018.

Zhang, J., Yang, Y., Zhang, Q., 2009. The particle swarm optimization algorithm based on dynamic chaotic perturbations and its application to K-means. In: 2009 International Conference on Computational Intelligence and Security, vol. 1, pp. 282–286.

Zimmerman, D.W., Zumbo, B.D., 1993. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. J. Exp. Educ. 62 (1), 75–86.