Comparing Manual vs Automatic Tuning of Differential Evolution Strategies for Energy Resource Management Optimization

José Almeida, Fernando Lezama, João Soares(✉), and Zita Vale

GECAD, LASI, Polytechnic of Porto, 4249-015 Porto, Portugal
{jorga,flz,jan,zav}@isep.ipp.pt

Abstract. The energy resource management problem in energy systems is hard to optimize, mainly due to non-linear restrictions and a large number of variables involved. This is partly because of the increased integration of distributed energy resources. Computational intelligence optimization techniques, namely evolutionary algorithms, are regarded as efficient techniques for identifying optimal and near-optimal solutions. However, these algorithms usually have in their design several parameters that need to be set and, in most cases, tuned for a given problem to find good solutions. This work proposes an automatic configuration approach of different differential evolution strategies using the irace package to solve a centralized day-ahead energy resource management problem. The problem considers an aggregator managing multiple resources, such as renewable generation, battery energy systems, electric vehicles, and loads with demand response capabilities. The aggregator aims to minimize operational costs and maximize revenues to obtain a profit. We compare the results of a "manual" tuning of parameters with the results obtained with the auto-tuned parameters using irace. Results show that the automatic configuration improves the profits of the aggregator in almost all strategies (except for DE/either-or-algorithm/1), getting the best results, an improvement of around 7%, with the automatically tuned DE/target-to-best/1 mutation strategy.

Keywords: Automatic tuning · Differential evolution · Energy resource management · Evolutionary algorithms · Iterated racing · Optimization

1 Introduction

The socio-economic situation of the energy sector is complex, necessitating extensive research and planning. In reality, many of the issues in this area are complicated and have traits like high dimensionality, a large number of restrictions, lack of information, and noisy and corrupted data due to the large uncertainty in the energy system [1,2]. Also, these problems frequently involve temporal limits

B. N. Jørgensen et al. (Eds.): EI.A 2023, LNCS 14467, pp. 44–59, 2024.
https://doi.org/10.1007/978-3-031-48649-4_3

that require solutions to operate in almost real-time [3]. Consequently, finding effective and exact solutions in a reasonable amount of time is still an issue that needs to be addressed for many energy problems.

Consider, for example, the energy resource management (ERM) in distribution networks. In such a problem, due to the grid constraints, we are dealing with a mixed-integer non-linear programming (MINLP) formulation, which is tough for mathematical methods to solve. Algorithms based on computational intelligence (CI), in this context evolutionary algorithms (EAs), have shown to be particularly well suited for this type of problem since they produce good results in a useful time. As such, they provide an effective optimization alternative to mathematical methods for problems in the energy domain [4]. These algorithms are also more tolerant to uncertainty and more suitable for addressing the nonlinearities prevalent in energy problems [5]. EAs are in their majority population-based algorithms with several associated parameters. Due to their stochastic nature, fine-tuning these parameters becomes essential to achieve feasible solutions to a wide range of problems with a minimum requirement for robustness [6]. This tuning can be done in different ways, but many authors have performed this tuning manually through a sensitivity analysis of the parameters [7]. Manual configuration is extremely time-consuming and needs the developers' expertise and knowledge of the specific problem to be solved [8]. Therefore, exploring automatic configuration methods that efficiently search the parameter space to find high-performing configurations and removing the drawbacks from manual configuration is essential in the design of optimizers. Automatic configuration methods can be divided into simple generate-evaluate methods (Brute force [9], and F-Race [10]), high-level generate-evaluate methods (Post-selection [11]), and iterative generate-evaluate methods (CALIBRA [12], Iterated F-Race [13], and others) [14]. To our knowledge, such automatic configuration methods regarding EA optimization for ERM problems have yet to be applied in the literature, so we believe this work will significantly contribute to this field.

In this paper, we propose an automatic tuning of the parameters of multiple differential evolution (DE) strategies considering the iterated racing F-race approach present in the irace package in [13] to solve an ERM problem [15]. The irace package was chosen for its customization (multiple automatic configuration procedures) and simplicity. In a smart grid problem (SG), a 33-bus distribution network (DN) is considered with distributed generation sources, electric vehicles (EVs), battery energy systems (BES), and demand response (DR) programs for load reduction. An aggregator optimizes resource allocation, minimizing electricity market purchases during peak hours while maximizing power sales to the market and meeting the needs of residential customers, EV users, and the BES. We analyze the experiments made by the irace package to find the best configuration for the parameters of different DE strategies and present the best configuration (final elite configuration) for each. In addition, we compare the results with those obtained with manual tuning made in [15]. The results are compared in terms of operational costs and the incomes of the aggregator.

The organization of this paper is as follows. The proposed methodology regarding the mathematical problem formulation and iterated racing approach is presented in Sect. 2. The case study employed in this work, concerning the setting used by irace and the DN, is presented in Sect. 3. The automatic configuration outcomes and the analyses of results are presented in Sect. 4. Finally, Sect. 5 draws the main takeaways from this work and suggests some possible topics for further research.

2 Proposed Methodology

The aggregator intends to reduce the operational costs in Eq. 1 for day-ahead management while maximizing electricity selling to consumers and market transactions in Eq. 2. The decision variables for energy resource generation power, DG unit commitment, BES and EV schedules, and DR loads, among others, are all included in the ERM model under study for each unit and each period considered. The voltage and angles in each bus must also be considered during scheduling.

2.1 Problem Formulation

The objective function for the day-ahead ERM formulation contemplates the operational costs associated with multiple resources (previously mentioned) for the 24-hour horizon with a time step of 1 h.

Operational Costs. Eq. 1 models the operational costs of the ERM model that the aggregator aims to minimize while maximizing the incomes in Eq. 2. The first and second terms of Eq. 1 include the costs of DG and excess of generation imbalance. The third and fourth terms of the equation represent the costs associated with BES and EV discharging. The fifth and sixth terms represent the costs associated with the incentive for DR programs and the negative imbalance from load not supplied. Finally, the income from selling electricity in the market is represented in the last term.

$$min f_{OC}^{\text{Day+1}} = \sum_{t \in T} \cdot \left(\begin{array}{l} \sum\limits_{i \in I} (p_{(i,t)}^{\text{DG}} \cdot C_{(i,t)}^{\text{DG}} + p_{(i,t)}^{\text{imb}^+} \cdot C_{(i,t)}^{\text{imb}^+}) + \\ \sum\limits_{s \in S} p_{(s,t)}^{\text{Sup}} \cdot C_{(s,t)}^{\text{Sup}} + \\ \sum\limits_{b \in B} p_{(b,t)}^{\text{dis}} \cdot C_{(b,t)}^{\text{dis}} + \sum\limits_{v \in V} p_{(v,t)}^{\text{dis}} \cdot C_{(v,t)}^{\text{dis}} + \\ \sum\limits_{l \in L} (p_{(l,t)}^{\text{Red}} \cdot C_{(l,t)}^{\text{Red}} + p_{(l,t)}^{\text{imb}^-} \cdot C_{(l,t)}^{\text{imb}^-}) \end{array} \right) \cdot \Delta t \qquad (1)$$

where T is the set of the number of periods $(1, 2, 3, ..., 24)$, I is the set of DG units $(1, 2, 3, ..., N_i)$, S the set of external suppliers $(1, 2, 3, ..., N_s)$ B represents the set of BES $(1, 2, 3, ..., N_b)$, V demonstrates the set of EVs $(1, 2, 3, ..., N_v)$, L is the set of different loads participating in the DR program $(1, 2, 3, ..., N_l)$ and M is the set

of electricity markets available for the aggregator transactions $(1, 2, 3, ..., N_m)$. Regarding the parameters, Δt represents the time step which in this case is considered to be one hour. $C^{\mathrm{DG}}_{(i,t)}$ is the cost associated with DG production in unit i for the period t (m.u./kWh), $C^{\mathrm{imb}^+}_{(i,t)}$ represents the cost of the exceeding power of DG unit i in periods t (m.u./kWh) and $C^{\mathrm{Sup}}_{(s,t)}$ the external supplier s electricity price for the period t (m.u./kWh). The discharging costs of the BES and EVs are the parameters $C^{\mathrm{dis}}_{(b,t)}$, and $C^{\mathrm{dis}}_{(v,t)}$ for BES b and EV v, respectively for the period t (m.u./kWh). $C^{\mathrm{Red}}_{(l,t)}$ is the DR cost of the load l (m.u./kWh), and $C^{\mathrm{imb}^-}_{(l,t)}$ is the cost associated with the demand not-supplied to the respective load (m.u./kWh). The decision variables in this equation are the following, $p^{\mathrm{DG}}_{(i,t)}$, which is the active power produced by each DG unit i for the period t (kW), $p^{\mathrm{imb}^+}_{(i,t)}$ describes the exceed active power of each DG unit (kW), $p^{\mathrm{Sup}}_{(s,t)}$ is the active power supplied by external supplier s in period t (kW). The active discharging power of each BES and EV is given by $p^{\mathrm{dis}}_{(b,t)}$ and $p^{\mathrm{dis}}_{(v,t)}$ respectively (kW). The load curtailment power of load l for period t is represented by the variable $p^{\mathrm{Red}}_{(l,t)}$ (kW) and the non-supplied power is given by $p^{\mathrm{imb}^-}_{(l,t)}$ (kW).

Aggregator's Incomes. In Eq. 2, the aggregator earns revenue from BES and EV charging, modeled with the first and second terms; revenue from selling electricity to residential loads with the third term; and offers for the electricity market with the fourth term. The aggregator needs to maximize this function to achieve profits in the day-ahead optimization. That is, the values obtained in Eq. 2 need to be superior to those in Eq. 1.

$$max f^{\mathrm{Day+1}}_{In} = \sum_{t \in T} \cdot \left(\begin{array}{l} \sum\limits_{b \in B} p^{\mathrm{cha}}_{(b,t)} \cdot S^{\mathrm{cha}}_{(b,t)} + \sum\limits_{v \in V} p^{\mathrm{cha}}_{(v,t)} \cdot S^{\mathrm{cha}}_{(v,t)} + \\ \sum\limits_{l \in L} p^{\mathrm{Load}}_{(l,t)} \cdot S^{\mathrm{Load}}_{(l,t)} + \sum\limits_{m \in M} p^{\mathrm{Sell}}_{(m,t)} \cdot S^{\mathrm{Sell}}_{(m,t)} \end{array} \right) \cdot \Delta t \qquad (2)$$

where $S^{\mathrm{cha}}_{(b,t)}$ and $S^{\mathrm{cha}}_{(v,t)}$ are the parameters associated with the prices of BES and EV charging (m.u./kWh). $S^{\mathrm{Load}}_{(l,t)}$ represents the tariff of load l in period t (m.u./kWh) and $S^{\mathrm{Sell}}_{(m,t)}$ is the price of selling electricity in market m in each period t (m.u./kWh). $p^{\mathrm{Load}}_{(l,t)}$ is the parameter of day-ahead load forecast in period t (kW). $p^{\mathrm{cha}}_{(b,t)}$ and $p^{\mathrm{cha}}_{(v,t)}$ are the variables of active charging power of each BES b and EV v (kW). The variable associated with power sold in the electricity market m in each period t is $p^{\mathrm{Sell}}_{(m,t)}$ (kW).

Objective Function. The cost minimization problem is defined in Eq. 3, where the aggregator subtracts the incomes from the operational costs to obtain a profit.

$$minimize\ z(\boldsymbol{x}) = -f^{\mathrm{Day+1}}_{In} + f^{\mathrm{Day+1}}_{OC} \qquad (3)$$

The complete mathematical formulations of the problem regarding all grid constraints and resource constraints that the objective function (Eq. 3) is subject

to can be found in [16]. In this paper, only the main network constraints are shown as follows:

Active and reactive power balance:

$$\left(\begin{array}{l} \sum\limits_{i\in\Omega_I^j}(p_{(i,t)}^{\mathrm{DG}}-p_{(i,t)}^{\mathrm{imb}^+})+\sum\limits_{s\in\Omega_S^j}p_{(s,t)}^{\mathrm{Sup}}+ \\ \sum\limits_{b\in\Omega_B^j}(p_{(b,t)}^{\mathrm{dis}}-p_{(b,t)}^{\mathrm{cha}})+\sum\limits_{v\in\Omega_V^j}(p_{(v,t)}^{\mathrm{dis}}-p_{(v,t)}^{\mathrm{cha}})+ \\ \sum\limits_{l\in\Omega_L^j}(p_{(l,t)}^{\mathrm{Red}}+p_{(l,t)}^{\mathrm{imb}^-}-p_{(l,t)}^{\mathrm{Load}})-\sum\limits_{m\Omega_M^j}p_{(m,t)}^{\mathrm{Sell}}- \\ V_{(j,t)}\cdot\sum\limits_{k\in K}V_{(k,t)}(G_{(j,k,t)}\cdot cos\theta_{(j,k,t)}+ \\ \qquad B_{(j,k,t)}\cdot sin\theta_{(j,k,t)}) \end{array}\right)=0 \qquad \forall t,\forall j,k\neq j \tag{4}$$

$$\left(\begin{array}{l} \sum\limits_{i\in\Omega_I^j}Q_{(i,t)}^{\mathrm{DG}}+\sum\limits_{s\in\Omega_s^j}Q_{(s,t)}^{\mathrm{Sup}}-\sum\limits_{l\in\Omega_L^j}Q_{(l,t)}^{\mathrm{Load}}- \\ V_{(j,t)}\cdot\sum\limits_{k\in K}V_{(k,t)}(G_{(j,k,t)}\cdot sin\theta_{(j,k,t)}- \\ \qquad B_{(j,k,t)}\cdot cos\theta_{(j,k,t)}) \end{array}\right)=0 \qquad \forall t,\forall j,k\neq j \tag{5}$$

where K is the set of buses $(1,2,3,...,N_k)$, Ω_I^j is the set of DG units at bus j of the network, Ω_S^j the set of external suppliers at bus j, Ω_B^j is the set of BES at bus j, Ω_V^j is the set of EVs at bus j, Ω_L^j represents the set of loads at bus j, and Ω_M^j is the set of electricity market buyers at bus j. Regarding $V_{(j,t)}$ represents the voltage magnitude at bus j in the period t (p.u.). $G_{(j,k,t)}$ and $B_{(j,k,t)}$ represent the real and imaginary part of the line admittance from bus j to bus k for the period t (Ω^{-1}). $Q_{(i,t)}^{\mathrm{DG}}$ (kvar), $Q_{(s,t)}^{\mathrm{Sup}}$ and $Q_{(l,t)}^{\mathrm{Load}}$ are the reactive powers of DG unit i for period t (kvar), the reactive power of external supplier s in period t and the reactive load power l for the period t (kvar).

Voltage magnitude and angle levels:

$$V_{(j,t)}^{\min}\leq V_{(j,t)}\leq V_{(j,t)}^{\max} \qquad \forall t,\forall j \tag{6}$$

$$\theta_{(j,t)}^{\min}\leq \theta_{(j,t)}\leq \theta_{(j,t)}^{\max} \qquad \forall t,\forall j \tag{7}$$

where $V_{(j,t)}^{\min}$ and $V_{(j,t)}^{\max}$ represent the minimum and maximum limits for the voltage magnitude at bus j for the period t (p.u.). The $\theta_{(j,t)}^{\min}$ and $\theta_{(j,t)}^{\max}$ are the minimum and maximum voltage phase angles at bus j in period t (rad).

Thermal line limits:

$$\left|\begin{array}{r} V_{(j,t)}([(V_{(j,t)}-V_{(k,t)})y_{(j,k,t)}]^*+ \\ [V_{(j,t)}\cdot\tfrac{1}{2}y_{Shunt\_j}]^*) \end{array}\right|\leq S_{(j,k,t)}^{\max} \qquad \forall t,\forall j,k\neq j \tag{8}$$

where $y_{(j,k,t)}$ is the line admittance from bus j to bus k for the period t t (Ω^{-1}), y_{Shunt_j} is the shunt admittance of the line connected to bus j t (Ω^{-1}) and $S^{\max}_{(j,k,t)}$ is the maximum apparent power flow in the line from bus j to bus k in period t (kVA).

2.2 Differential Evolution Strategies

DE is a population-based EA for continuous optimization. DE combines solutions from the population using a linear operator. New solutions are generated at each iteration and evaluated in a given fitness function to optimize a particular problem. The algorithm retains the solutions with better performance, and solutions with lower fitness values are replaced in the iterative process. The phases of this method are as follows: first, a solution (target vector) is formed; next, a donor vector is generated by mutation (by a combination of different solutions in the population); and last, a trial vector is generated through a recombination operator between the target vector and the donor vector. The way in which the donor vector is created can have variations that give rise to different DE strategies. The reader can consult [15] to get specifics on these DE strategies.

We briefly discuss four well-known DE mutation strategies applied in this work to address the ERM problem and to apply the automatic configuration package. The first DE strategy is the DE/rand/1 strategy, where a linear combination of three randomly selected solutions creates the donor vector. In the second strategy, the DE/target-to-best/1, the base vectors are chosen following a line formed by the target vector and the best-so-far vector (i.e., the best-so-far solution found in the iterative process). In the third strategy, the DE/rand/1 with dither, the operator uses a random variation of the scale factor (dither), which is incorporated in the formulation of the donor vector. Finally, in the DE/rand/1/either-or, either a three-vector pure mutation method (like standard DE), with probability p_m, or a random recombination technique, with probability 1-p_m, is used to create the mutant vector.

For a full explanation of the solution encoding used for DE optimization and the formulations of the different DE strategies, the reader can be directed to [15].

2.3 Iterated Racing

Figure 1 shows the automatic configuration approach based on iterated racing for the multiple DE strategies. Initially, irace needs an input scenario that allows irace to run and evaluate the various configurations based on iterated racing. The finite number of configurations to start the race is related to the maximum experiment budget given in the scenario. Additionally, each DE parameter for irace to configure (i.e., name, type, range) is also set in the scenario, together with an initial configuration that irace first evaluates and forbidden configurations in terms of logical expressions between parameters, which irace does not consider. Each DE strategy is also passed as input to obtain the auto-tuning parameters for the respective strategy.

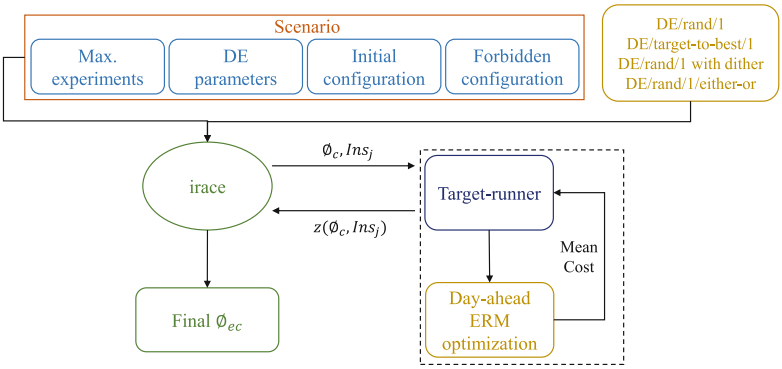

Fig. 1. Proposed method for automatic configuration of the multiple DE strategies using irace.

Irace then calls the target-runner, which is in charge of analyzing a specific target algorithm configuration (ϕ_c) of a particular instance ($Ins_j(instance, seed)$) and returning the appropriate cost value ($z(\phi_c, Ins_j)$). In this case, since we omitted different training instances due to the characteristics of the problem in [15], irace only considers different random seeds as instances. The target-runner evokes the DE day-ahead ERM optimization problem (implemented in MATLAB) to obtain the respective mean cost results over several runs.

After irace finishes the iterative process, it selects the best-performing configurations and the so-called elite configurations (ϕ_{ec}) based on the lowest mean cost values.

3 Case Study

This section describes the case study used to validate the proposed approach. The case study includes data regarding energy resources and the parameterization needed for irace to perform auto-tuning.

3.1 33-Bus Distribution Network

The SG consists of a medium voltage 12.66 kV 33-bus distribution network [17] used to test the multiple DE strategies and automatic tuning of irace. The 33-bus network scenario includes 1800 EVs with V2G capabilities, 67 DGs (including a sizable wind turbine), 10 external providers, and 15 BES. In bus 33, external suppliers are represented as a substation linked to the main grid. Figure 2 shows the total forecasted load demand comprising 32 residential consumers and the total forecasted renewable generation (PV and wind) for the day-ahead optimization. Direct Load Control (DLC) contracts as low as 0.02 m.u./kWh are also considered. Consumers receive this advantage for each lowered energy unit

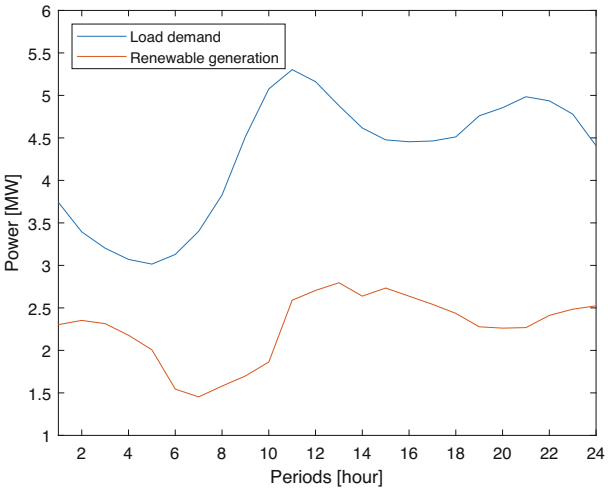

Fig. 2. Total forecasted day-ahead load demand and renewable generation.

instead of paying the 0.14 m.u./kWh supply price agreed upon by the aggregator. The selling price for energy is also fixed at 0.14 m.u./kWh. Additionally, a fleet of 1800 EVs with V2G capabilities is considered, with a forecast of 13.77 MWh total energy needed for 2553 trips. EV and BES have a discharge cost of 0.19 m.u./kWh. EV and BES have charging/discharging efficiencies set at 70% and 90%, respectively.

Modeling these EV trips was done using an EV travel behavior simulator tool suggested in [18]. With the aid of this simulator, we can gather information about each EV's trip, including the maximum charge and discharge rates, the minimal amount of charging necessary for the EV to complete its journey in the upcoming hour (or hours), as well as many other variables that are used as input for the optimization.

3.2 Irace Parameterization

The DE algorithm used in this work only needs four different parameters to be set. Table 1 shows each parameter, where NP is the population size, $maxIt$ is the maximum number of iterations the algorithm performs, and F and Cr are the scale factor and crossover probability, respectively. These parameters must be set and passed to irace, giving the software the type of each parameter (e.g., NP and $maxIt$ are integer parameters). It is also needed to set the range of values for each parameter, i.e., the search space that the irace algorithm uses to find the best configurations. We set these ranges according to the manual tuning performed in [15] and also consider the computation effort in the search space.

Irace starts with an initial configuration to be set and tested first; if results are satisfactory, similar configurations will be generated and tested. Table 1 presents

Table 1. DE parameters.

Parameter	Type	Parameter range	Initial value
NP	i	(10,100)	20
$maxIt$	i	(1,500)	100
F	r	(0.00,1.00)	0.30
Cr	r	(0.00,1.00)	0.50

the values for the initial configuration given to irace, starting from the initial point considered in the previous work where F, Cr, NP, and $maxIt$ were set to 0.3, 0.5, 20, and 100, respectively. These values did not result from an a priori tuning. They were just randomly set and given to irace.

Additionally, to be in conformity with [15], we set the maximum number of function evaluations (FEs) that the algorithm can test to 10,000 ($NP \times maxIt \leq 10,000$). We set this restriction as a forbidden configuration to avoid testing combinations of NP and $maxIt$ that result in a large number of FEs. Finally, a maximum of 300 experiments, which sets the tuning budget, limiting the total number of executions. This number represents the maximum tuning budget used for irace, i.e., the number of configurations evaluated for each instance. We noticed that increasing this value would greatly increase simulation time since more configurations would need to be tested.

We implement and evaluate the irace package on a Linux virtual machine running Ubuntu 22.04.1 LTS equipped with an Intel Xeon Gold 5120 processor operating at 2.20GHz and 16GB of RAM. The irace package used a target-runner developed in Python 3.6.10, and the DE optimization strategies were implemented in MATLAB R2018a.

4 Results and Discussion

This section presents the results and the experiments made using irace for the parameter auto-tuning of the multiple DE strategies. Also, we compare the best-obtained automatic tuning configurations with those found with the manual tuning in [15].

4.1 Auto-tuning Experiments

Irace simulations were done for a total of 10 runs and 10,000 function evaluations (FEs) for each DE optimization in MATLAB. That is, the MATLAB code is run for 10 trials when it is called by the target-runner in irace.

Concerning the experiment process, Fig. 3 shows the performance of the elite configurations in each iteration of irace for the DE/rand/1 strategy. The figure shows the best configurations for ten instances (random seeds) evaluated. The final best configuration found by irace for this strategy was configuration 24,

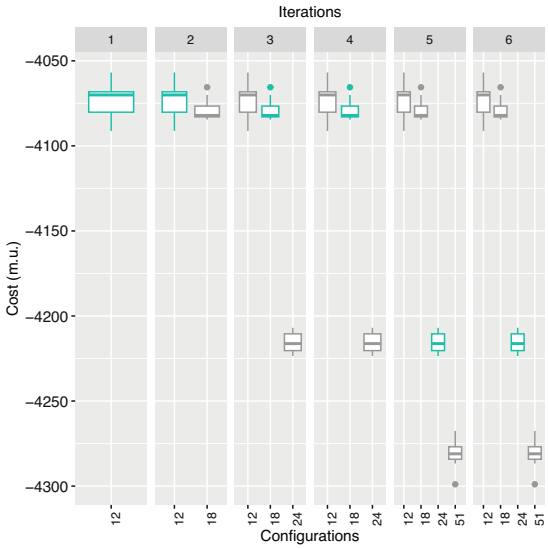

Fig. 3. Elite configuration performance by iteration of irace for DE/rand/1 strategy (in green the best configuration). (Color figure online)

which presented the lowest mean cost value (-4,215.31 m.u.). Notice that, even though configuration 51 gives lower values, irace did not obtain cost results for this configuration in all the ten tested instances (only for 8 of those), which is why this configuration was not chosen as the best. That is, in two out of ten evaluated instances, irace did not obtain any cost results (presented NA results), disregarding this configuration as the best (more robust) for this DE strategy because the race terminated before the instances were considered for this configuration. Figure 4 presents the performance of the best elite configurations for the DE/target-to-best/1 strategy, similar to the preceding case. In this situation, only configuration 49 was considered elite in the final iteration, with the others being discarded by the statistical test done by irace. This configuration presented a mean cost value of -4,053.93 m.u. for nine seeds, a reduction of 189.08 m.u. compared to configuration 2 (best elite in iteration 4).

Figure 5 shows the iterative process regarding the elite configurations for the DE/rand/1 with dither mutation strategy. The figure shows that configuration 27 was the best-performing configuration with the lowest mean cost value compared with configurations 9 and 13, which were also elites in the last iteration. Configuration 27 obtained –4,173.36 m.u., a decrease of 2.78% compared to configuration 13 and 11.38% compared to configuration 9. Similar to the DE/target-to-best/1, only one elite configuration was obtained with DE/rand/1/either-or strategy in the last iteration of irace, with the rest being discarded, as Fig. 6 shows. In this strategy, the best-performing configuration was configuration 52 (the last configuration evaluated by irace), with a mean cost value of –4,144.43 for 9 instances.

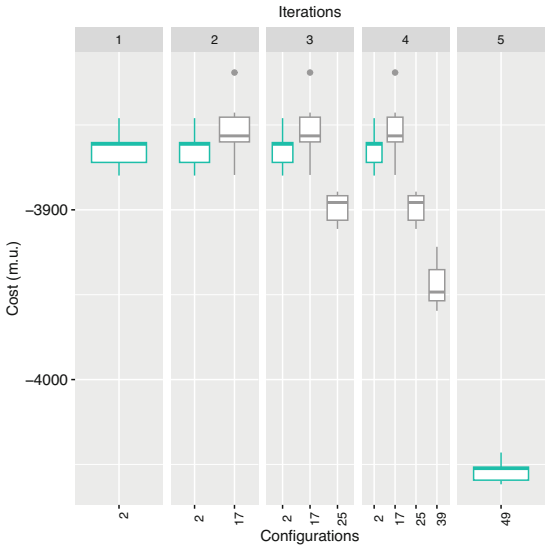

Fig. 4. Elite configuration performance by iteration of irace for DE/target-to-best/1 strategy (in green the best configuration). (Color figure online)

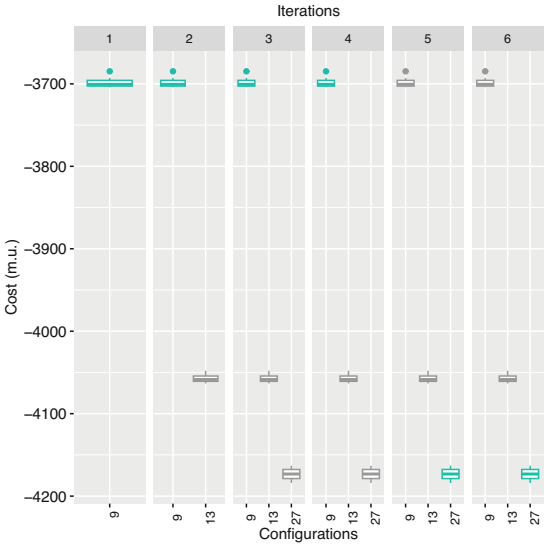

Fig. 5. Elite configuration performance by iteration of irace for DE/rand/1-with-dither strategy (in green the best configuration). (Color figure online)

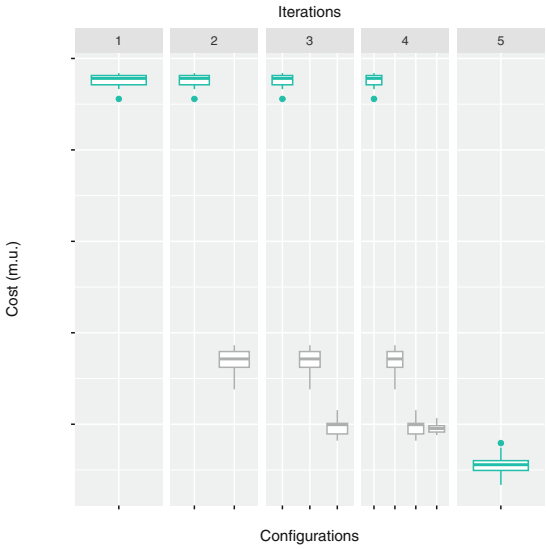

Fig. 6. Elite configuration performance by iteration of irace for DE/rand/1/either-or strategy (in green the best configuration). (Color figure online)

Table 2. Best elite configurations obtained by irace.

Strategy	NP	$maxIt$ (60k FEs)	F	Cr
DE/rand/1	37	249 (1,622)	0.28	0.39
DE/target-to-best/1	21	425 (2,857)	0.57	0.10
DE/rand/1 with dither	27	428 (2,222)	0.03	0.25
DE/either-or-algorithm/1	49	181 (1,224)	0.25	0.33

We took the final best elite configurations for each proposed DE mutation strategy. The parameters of the best elite configurations given by irace are presented in Table 2.

The DE/rand/1 and DE/either-or-algorithm/1 strategies performed better for higher values of NP and lower values of $maxIt$ compared to the other two strategies. Notice that the obtained parameters were found for 10,000 FEs due to the computational effort, which is a smaller number of FEs than the one used in [15] for the final result comparison. Thus, for a more realistic comparison of the final results presented in Sect. 4.2 of [15], we updated the number of iterations based on the NP parameter to match the 60,000 FEs used to solve the ERM optimization problem in the cited paper.

4.2 Manual vs. Automatic Tuning ERM Results

Before starting the comparison with the manual tuning performed in [15], we take the configuration space tested manually and show where our automatic tuning lays in those configuration spaces. Figure 7 show heatmaps representing the performance of configurations tested in [15].

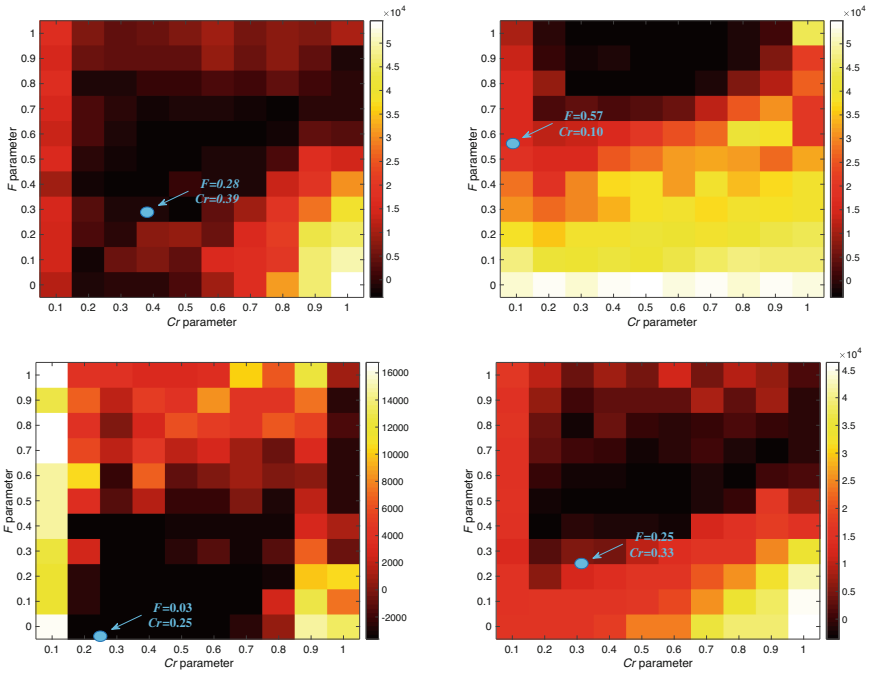

Fig. 7. Auto-tuning points for F and Cr compared to the manual analysis of DE strategies. (a) DE/rand/1. (b) DE/target-to-best/1. (c) DE/rand/1 with dither. (d) DE/rand/1/either-or (in blue, the best F and Cr parameters found with auto-tuning). (Color figure online)

A darker color represents a better performance of a specific combination of F and Cr parameters. We then plot the best configuration found with irace automatic tuning as a blue point in that figure. It can be noticed in Fig. 7(a) that the manual tuning for DE/rand/1 showed good performances for Cr between [0.3,0.8] and F in the range of [0.3,07], whereas irace obtained values for F and Cr near that area. In contrast, Fig. 7(b) shows that for the DE/target-to-best/1 strategy, the F and Cr point acquired by irace does not fall inside the ranges where the manual tuning showed good fitness results (specifically found for higher values of F (between 0.7 and 1) and [0.3,0.7] for Cr). Figure 7(c) shows that, again, the auto-tuning configuration of DE/rand/1 with dither strategy obtained is within the range of values for F and Cr recommended

Table 3. Comparison of each algorithm's manual and automatic profit and cost results.

Strategy	Manual tuning (m.u.)			Automatic tuning (m.u.)		
	Avg. Profits ± std	In	OC	Avg. Profits ± std	In	OC
DE/rand/1	4,458.99 ± 20.48	19,939.98	15,480.99	4,705.11 ± 9.63	19,724.46	15,019.35
DE/target-to-best	4,151.39 ± 28.46	20,356.94	16,205.55	4,465.51 ± 13.26	19,699.04	15,233.53
DE/rand/1 with dither	4,610.24 ± 19.15	19,798.25	15,188.02	4,633.38 ± 14.78	19,809.28	15,175.91
DE/either-or-algorithm/1	4,746.70 ± 6.46	19,624.75	14,878.05	4,307.64 ± 42.00	19,001.98	14,694.34

by the manual tuning. Finally, similarly to what occurred with the DE/target-to-best/1 strategy, Fig. 7(d) shows an automatic tuning configuration for the DE/rand/1/either-or strategy a bit out of the recommended ranges of the manual tuning. Note that in the manual tuning, the NP and $maxIt$ parameters were fixed values, but in irace, these parameters are optimized together with F and Cr, so these figures do not ideally represent the performance of irace since the other parameters also need to be taken into account.

Comparing the manual tuning of the DE strategies for the NP parameter to the automatic configuration, the manual tuning obtained the best results for a NP value of 30 in all strategies. In contrast, the automatic configuration obtained more specific values for each strategy, as shown previously, and the number of iterations was uploaded accordingly to the maximum number of FEs. Table 3 gives the average profit results (Eq. (3)) obtained with the best manual configuration found in [15] and the automatic configuration found using irace. 50 trials were done using the best automatic configurations found (remember that 60,000 FEs were considered for a fair comparison). The table shows that the automatic configuration for each DE strategy showed better results, except for the DE/either-or-algorithm/1, where a decrease of 9.25% compared to the manual configuration was registered. Regarding the average optimization time, the automatic tuning was faster in all strategies. The running time for the manual tuning took around 60 min in all algorithms, while the automatic tuning took about 40 min.

The decrease in performance for the automatic tuning of the DE/either-or-algorithm/1 strategy is justified by the low incomes obtained, shown in Table 3, compared to the manual configuration. The parameters provided by irace for DE/rand/1 showed an increase in profits of 246.12 m.u. in the ERM optimization compared to the original work. This improvement is given mostly by a decrease in operational costs of 2.98% compared to the costs obtained by the manual configuration (Table 3). Concerning the DE/target-to-best/1 strategy, the automatic configuration found a solution that increases the profits by 314.12 m.u. compared to the manual configuration. This increase is accomplished by the reduction in operational costs of 972.02 m.u., even though the auto-tuned obtained less income (657.90 m.u.) compared to the manual tuning, evidenced in Table 3. Regarding the DE/rand/1 with dither mutation strategy, the auto-tuning provided a slightly better solution, with the incomes and operational costs being similar.

5 Conclusions

In this work, we proposed automatically tuning multiple parameters for diverse DE mutation strategies using irace. Irace is a software package that utilizes iterated racing for automatic configuration evaluations. We compared the results obtained with the auto-tuning with those obtained using a manual configuration in a centralized day-ahead ERM optimization problem.

Results showed that the parameters obtained in the automatic configuration found better optimization solutions for all proposed DE strategies except for DE/either-or-algorithm/1. The decrease in performance for this strategy (worse profit results) can be justified by the number of maximum experiments established in the setting of the irace software. An increase in this specific parameter would allow irace to test more configurations, allowing it to test and find better configurations with this strategy (and with the rest of the tested strategies). However, increasing this parameter would also increase execution time as more optimization trials would be required. This is important to recall since automatic tuning is intended to be a more efficient method to configure our algorithms; thus, performing an adequate number of tests is key to achieving such efficiency. Still, the automatic tuning was insufficient to find a better configuration than the one found with the manual tuning for the DE/either-or-algorithm/1. Nevertheless, the automatic tuning found an acceptable solution with DE/rand/1 strategy, a solution that is just 0.88% worse than the best solution found with the manual tuning configuration. These results show that despite the advantages of automatic tuning, there is still room for improvement when using such methods.

As interesting venues for future research, new tests could be implemented with an increase in the maximum tuning budget, increasing the number of evaluated configurations, and increasing the computational time. Also, we could explore the efficiency of the method when initial configurations for each strategy, based, for example, on the best configuration found with the manual tuning, are provided as starting points. Additionally, more instances of the problem with modified characteristics would be required to validate the algorithms' performance and guarantee a more general algorithm parameterization.

Acknowledgments. The present work has received funding from the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES) and through the FCT Portuguese Foundation for Science and Technology (2019.00141.CBM). José Almeida is supported by FCT with Ph.D. Grant 2022.09590.BD. João Soares has received funding from FCT, namely CEECIND/00420/2022. The authors acknowledge the work facilities and equipment provided by the GECAD research center (UIDB/00760/2020 and UIDP/00760/2020) to the project team.

References

1. Yazdanie, M., Orehounig, K.: Advancing urban energy system planning and modeling approaches: gaps and solutions in perspective. Renew. Sustain. Energy Rev. **137**, 110607 (2021)

2. Milford, J., Henrion, M., Hunter, C., Newes, E., Hughes, C., Baldwin, S.F.: Energy sector portfolio analysis with uncertainty. Appl. Energy **306**, 117926 (2022)
3. Hossain, M.A., Pota, H.R., Squartini, S., Abdou, A.F.: Modified PSO algorithm for real-time energy management in grid-connected microgrids. Renew. Energy **136**, 746–757 (2019)
4. Soares, J., Pinto, T., Lezama, F., Morais, H.: Survey on complex optimization and simulation for the new power systems paradigm. Complexity **2018**, 1–32 (2018)
5. Songyuan, Yu., Fang, F., Liu, Y., Liu, J.: Uncertainties of virtual power plant: problems and countermeasures. Appl. Energy **239**, 454–470 (2019)
6. Kazikova, A., Pluhacek, M., Senkerik, R.: Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison? MENDEL **26**(2), 9–16 (2020)
7. Vieira, M., Faia, R., Lezama, F., Vale, Z.: A sensitivity analysis of PSO parameters solving the P2P electricity market problem. In: 2022 IEEE Congress on Evolutionary Computation (CEC), pp. 1–7, Padua, Italy, July (2022). IEEE
8. Stützle, T., López-Ibáñez, M.: Automated design of metaheuristic algorithms. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics. ISORMS, vol. 272, pp. 541–579. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_17
9. Birattari, M.: Tuning metaheuristics: a machine learning perspective. Tuning Metaheuristics **197**, 221 (2009)
10. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, GECCO 2002, pp. 11–18, San Francisco, CA, USA (2002). Morgan Kaufmann Publishers Inc
11. Yuan, Z., Stützle, T., Montes de Oca, M.A., Lau, H.C., Birattari, M.: An analysis of post-selection in automatic configuration. In: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference - GECCO 2013, pp. 1557, Amsterdam, The Netherlands (2013). ACM Press
12. Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Oper. Res. **54**(1), 99–114 (2006)
13. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016)
14. Huang, C., Li, Y., Yao, X.: A survey of automatic parameter tuning methods for metaheuristics. IEEE Trans. Evol. Comput. **24**(2), 201–216 (2020)
15. Lezama, F., Sucar, E., de Cote, E.M., Soares, J., Vale, Z.: Differential evolution strategies for large-scale energy resource management in smart grids. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1279–1286, Berlin Germany, July (2017). ACM
16. Soares, J., Ghazvini, M.A.F., Silva, M., Vale, Z.: Multi-dimensional signaling method for population-based metaheuristics: solving the large-scale scheduling problem in smart grids. Swarm Evol. Comput. **29**, 13–32 (2016)
17. Baran, M.E., Wu, F.F.: Network reconfiguration in distribution systems for loss reduction and load balancing. IEEE Trans. Power Deliv. **4**(2), 1401–1407 (1989)
18. Soares, J., Canizes, B., Lobo, C., Vale, Z., Morais, H.: Electric vehicle scenario simulator tool for smart grid operators. Energies **5**(6), 1881–1899 (2012)