Conference on Electronics, Telecommunications and Computers – CETC 2013

Solving non-Convex and Restricted Problems using Swarms – Economic Dispatch Case

João Pontes\textsuperscript{a}, P.M.Fonte\textsuperscript{a}, Rui Pestana\textsuperscript{a, b}

\textsuperscript{a}Departamento de Engenharia, Rua Conselheiro Emídio Navarro, Lisbon 1959-007, Portugal
\textsuperscript{b}REN - Redes Energéticas Nacionais, Av. Estados Unidos da América 55 Lisbon 1749-061, Portugal

Abstract

In this paper three heuristic techniques based in swarm intelligence are studied, namely its ability to solve non-convex, non-differentiable and highly restricted optimization problems. The performances of Particle Swarm Optimization, Bee Colony Optimization and Cockroach Swarm Optimization solving a set of benchmark functions will be compared. These techniques will also be applied to an electrical engineering problem, to be precise, the economic dispatch with non-convex cost functions. The results obtained up to now, have been demonstrating that these techniques are able to reach good results in the benchmark functions as well as in the problem of economic dispatch.

1. Introduction

Optimization or mathematical programming algorithms are search methods with the purpose to find the best solution to an optimization problem subject to a set of constraints. Despite this simple definition it can hide a number of complex issues, like problems with high number of dimensions, nonlinear, non-continuous and non-differentiable cost functions or even nonlinear boundaries, which may restrict the search area. The search space can be involved with many possible solutions, the characteristics of the problem may change over time, or the problem being optimized may have conflicting objectives. In all of these situations we seek an optimal solution, or at least an acceptable solution by the most efficient and effective manner [1].

The Economical Dispatch (ED) problem is an important aspect inside a power system’s operation. Essentially, ED is intended to assess the power that each online thermal power unit should generate with the lowest economical cost and respecting all the boundaries, \textit{i.e.}, respecting the technical and load demand constrains. Over the past decade, many methods have been developed to solve the ED problem. There are the traditional methods used in...
optimization packages such as Gradient, Lagrangean function, Lambda-iteration method, Dynamic Programming, Newton’s method, Linear Programming and Interior Point method, among others [2][3]. However, some generation cost functions of recent thermal units can be not continuous, not convex, neither differentiable. For instance, cost functions for coal thermal power plants are non-convex due to the non-linearity introduced by the steam valves operation. Commercial tools that are able to solve economical dispatch for thermal units always require convex cost functions. This requirement can be due to the limitations of the optimizing tool or the need of rapidity and non-convex algorithms tend to be slow. When we need hydro-thermal coordination (coordination of thermal and hydro power plants), the simplification on the thermal cost functions are even bigger, adjusting (convex) non-linear function to linear function. It means that there are no economical dispatch commercial tools for non-convex cost function. To overcome this problem, sometimes the technique is to split the space solution in convex sub-spaces and then use conventional algorithms. This technique may create a huge number of solutions, some possible, others not, and the best solution must be found inside the set of feasible results.

Swarm optimization techniques due to its nature should be able to explore non-convex search spaces, finding an optimal solution without the necessity of pre-processing of cost functions.

In figure 1 is shown an example of a non-convex cost function and respective derivative resulting from a thermal power plant with valve-point effect.

![Fig. 1. a) Non-convex cost function with valve point effect; b) Its derivative](image-url)

### Nomenclature

- $P_i$: Power generated by generator $i$
- $a_i, b_i, c_i, e_i, f_i$: Cost function coefficients
- $UR_i / DR_i$: Up / down ramp limits of generator $i$
- $p_{min} / p_{max}$: Minimum and maximum power of generator $i$
- $N_G$: Number of generators
- $P_L$: Power load
- $P_{Loss}$: Power loss
- $P_i^\prime$: Previous power output of generator $i$
- $B_{ji}$, $B_{al}$, $B_{oo}$: Matrix of Kron’s loss equation
- $l_{ij}^{LB} / l_{ij}^{UB}$: Lower / upper bound of fuel $j$ of generator $i$
- $SR_{ij}^{max}$: Maximum spinning reserve of generator $i$
- $SR_{ij}^{max}$: Spinning reserve of generator $i$
- $c_1 / c_2$: Cognitive / social coefficients
- $\omega_{init} / \omega_{final}$: Initial / final value of inertia
- $G_{best} / x_{best}$: Best global / local position
- $P_{BCO,i}$: Probability of each scout $i$ to be followed
- $P_{PSO,i}$: Best particle within visual scope
2. Swarm intelligence

Swarm intelligence has arisen from the understanding of animal behavior. In a very generic way, swarm intelligence can be characterized as the collective behavior of self-organized, distributed, autonomous and dynamic multi-agent system, displaying some sort of intelligence. These systems are formed by a population (swarm) of simple computational agents which have the capacity to understand and modify its environment in a local manner. These skills allow the communication between the agents, which capture the changes in environment generated by the behavior of other agents. There is no central structure of command and no individual agent takes the role of commander of the swarm. Everyone influences with greater or lesser weight the course of the swarm and the solutions are emerges, instead of pre-established one, which means that the solutions are obtained iteratively, with different inputs and surrounding environment in constant change.

There are several algorithms (metaheuristics) which are inspired on biological behaviors like evolutionary programming (EP)[2], evolutionary strategies (ES), particle swarm optimization (PSO)[2], bacteria foraging optimization (BFO)[6], ant colony optimization (ACO)[6], artificial bee colony (ABC)[6] and genetic algorithms (GA) among many others.

2.1. Particle swarm optimization

Particle Swarm Optimization (PSO) [4] is based on bird flock behavior searching for food. The flock increases the probability of finding food when it exhibits a cooperative behavior in the search. Each individual is considered a particle, which represent possible solutions to the given problem and the flock is considered as a swarm. Each particle have to decide between taking a selfish route (due to its cognitive behavior), moving toward the best position it has found so that moment or take a social route (due to its social behavior), in which it follows the position of the swarm’s best particle [6]. This is the metaphoric basis of PSO. Each particle interacts with each other through three behaviors, namely, inertia, cognitive and social. Cognitive behavior gives a higher weight to the best fitness found by each particle and the social behavior gives a higher weight to the particle which found the best fitness of the entire swarm. Inertia limits the velocity changes of the particles. The mathematical expression of the movement of a particle in PSO algorithm is shown in (1) and (2). In (1) $c_1$ and $c_2$ are the cognitive and social parameters, as well as $\omega$ which represent the inertia. The values of $x_{best}$ and $G_{best}$ represent respectively, the position of the best fitness of each particle and the position of the best fitness of entire swarm. In (2) $x$ represents the position of each particle and $v$ the velocity that results from (1). PSO is a simple and fast algorithm, which has been used on several problems of different fields of knowledge. In figure 2 is shown the movement of a particle.

\[
v_{i+1} = \omega \cdot v_i + c_1 \cdot \text{rnd}(.) \cdot (x_{best} - x_i) + c_2 \cdot \text{rnd}(.) \cdot (G_{best} - x_i) \tag{1}
\]

\[
x_{i+1} = x_i + v_{i+1} \tag{2}
\]

2.2. Bee colony optimization

Bee Colony Optimization (BCO) is characterized by mimicking bee’s behavior in search for flowers. BCO creates three different bees: scout, followers and onlookers. The first phase sets the scouts bees in search of flowers in a random path in the search space. Next, the scouts return to the hive and inform the followers through a wiggling dance about the best flower they found. For a bee, the best found flower is the one with more pollen. The followers will decide probabilistically which scouts’ flower to choose. If a follower can find a better flower surrounding the one found by the scout, then that is now the flower being presented to the followers back in the hive. When on the other hand no better flower can be found surrounding a given scout’s flower, then that scout will be reassigned to fly again to find a new flower. This is the final phase, when no improvement happens and onlooker bees set out randomly to another flower [6].

In (3) the probabilities of each scout’s flower to be followed by the followers are calculated by the quotient of
each scout’s fitness and the sum of every scouts’ fitness. Expression (4) gives the position of the follower in the neighborhood of the scout’s position, \( x_{ij} \), by comparing the scout’s position with another randomly selected scout, \( x_{kj} \). A particularity of BCO is that the new position obtained by the follower is a result of comparing only one variable between different scouts, meaning that the \( j \) of \( x_{ij} \) is compared with the same variable in the different position given by the index \( k \).

While in PSO the entire swarm has the tendency to converge to the best position, BCO has no such convergence. Even if it finds the best value right from the start, it is expected to continue searching the entire search space, trying to find a better position. From a processing point of view, PSO needs less processing resources than BCO.

\[
P_{BCO,t} = \frac{\text{fit}(x_i)}{\sum_{n=1}^{N} \text{fit}(x_n)} \quad (3)
\]

\[
v_{ij} = x_{ij} + \text{rand}() (x_{ij} - x_{kj}) \quad (4)
\]

2.3. Cockroach swarm optimization

Some algorithms have been introduced that try to simulate cockroach’s behavior, namely Cockroach Swarm Optimization (CSO) and Roach Infestation Optimization (RIO). Cockroaches show certain behaviors that are presented in CSO, like chase swarming, dispersing and ruthless. Through chase swarming, cockroaches within visual scope tend to follow the best one, creating clusters. Dispersing is a defensive mechanism used when cockroaches sense danger. This behavior is desirable from an optimization perspective, because it may help the algorithm avoid local minimums. Finally, ruthless is the name given to the act of cannibalism among cockroaches. When there isn’t enough food, a bigger cockroach may eat a smaller one. Computationally, this is wanted to avoid wasting resources on similar solutions.

Its mathematical formulation intends to imitate the behaviors of cockroach swarms as chase swarming, dispersing and ruthless. These agents can communicate with each other and feel the environment’s changes, dispersing randomly after a reached certain number of iterations. The algorithm starts by randomly initializing the cockroaches’ positions. As number of iterations increase, the cockroaches begin to form clusters, once cockroaches tend to go towards the best individual within visual scope, like it’s described in expression (5). If a cockroach presents the best fitness within its own visual scope, then it will represent the swarm’s best fitness \( P_g \) as described in expression (6). After a certain number of iterations, a certain number of cockroaches will be dispersed. Ruthless can happen when a certain number of iterations are reached, then swarm loses one cockroach.

\[
x'_{i} = x_{i} + \text{step.rnd}() (P_{i} - x_{i}), \quad x_{i} \neq P_{CSO,t} \quad (5)
\]

\[
x'_{i} = x_{i} + \text{step.rnd}() (P_{g} - x_{i}), \quad x_{i} = P_{CSO,t} \quad (6)
\]

3. Swarm behavior in optimization

To demonstrate the performances of swarms to solve multi-dimensional and non-convex problems, a set of benchmark functions are tested, (7) to (11)[5][6]. The Sphere function (7) is continuous, convex and unimodal with only a global minimum where \( f(x_i) = 0 \) with \( x_i = 0 \). The Rosenbrock function (8) can also be considered unimodal but non-convex with a global minimum inside a long, narrow and parabolic shaped flat valley. The valley is generally trivial to reach but to converge to the global minimum is not. The global minimum is \( f(x_i) = 0 \) at \( x_i = 1 \). The Schaffer function (9) is bidimensional and multimodal presenting many local optima. The minimum is \( f(x_i) = 0 \) at \( (x_1, x_2) = (0,0) \). The Griewank function (10) is widely used to test the optimization algorithms. In a range of \([-600,600]\] has 191 local minima, with the global minimia at \( x_i = 0 \) where \( f(x_i) = 0 \). The Rastrigin function (11) is multimodal where the number of local minimum increases exponentially. There is a minimum at \( x_i = 0 \) where \( f(x_i) = 0 \). The mathematical expressions of the benchmark functions can be seen on table 1.
Table 1. Mathematical expressions of the benchmark functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Mathematical expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere ($f_1$)</td>
<td>$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$</td>
</tr>
<tr>
<td>Rosenbrock ($f_2$)</td>
<td>$f_2(\vec{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$</td>
</tr>
<tr>
<td>Schaffer ($f_3$)</td>
<td>$f_3(\vec{x}) = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$</td>
</tr>
<tr>
<td>Griewank ($f_4$)</td>
<td>$f_4(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \frac{x_i}{\sqrt{i}}\right) + 1$</td>
</tr>
<tr>
<td>Rastrigin ($f_5$)</td>
<td>$f_5(\vec{x}) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)$</td>
</tr>
</tbody>
</table>

With the exception of Schaffer function (9) the remaining ones, were tested with 5 and 10 dimensions to study the behavior of the swarms solving problems with crescent number of dimensions. As heuristic methods may not converge exactly to the same solution at each run due to their stochastic behaviour, their performances could not be judge by the results of a single trial[3]. Due to that, all cases were performed 20 times keeping the average, maximum and minimum reached values. The maximum number of iterations was 3000 to $f_1$ and $f_3$ and 5000 to the remaining. The success rate is measured by the average of the optimal values reached in each run whereas the robustness is measured by the standard deviation of the results [3][6]. Regarding the number of particles chosen, there are not a specific number of particles that each swarm should have. A compromise must be obtained between a high number of particles, which may reach a better solution but take more processing resources, and a low number of particles, which can use lesser processing resources but also reach a poor solution. Due to that, using as starting point some authors [3], [6], [8] and [9] and with some trial and error, in all tests were used 30 particles.

Table 2 shows the parameters of all heuristics used in all the tests of this work.

Table 2. Parameters set into the different applied algorithms.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>PSO</th>
<th>BCO</th>
<th>CSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$c_2$</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\omega_{\text{init}}$</td>
<td>0.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\omega_{\text{final}}$</td>
<td>0.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>attempts$_{\text{max}}$</td>
<td>-</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>Dispersing iteration</td>
<td>-</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td>Dispersion percentage</td>
<td>-</td>
<td>-</td>
<td>100 %</td>
</tr>
<tr>
<td>Visual scope</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3 shows the results of benchmark function with 5 and 10 dimensions as well as the time to find the minimum value and the number of evaluations. In bold are highlighted the best values reached by each algorithm for each function and dimension.

Beyond that, one can also see that there is not an heuristic which demonstrates clear prominence over the others. As example, BCO reached the lowest values for function $f_1$ with 5 and 10 dimensions, as well as the lowest average values and a biggest robustness with the lowest standard deviation. In the case of function $f_4$ was CSO which reached the lowest minimums as well as the lowest averages and smaller standard deviation.
The mathematical expression (12) represents the cost function of units with valve point effects which behavior is depicted in figure 1 a). The power balance equation is modeled by (13) forcing that the total production be equal to the load plus the power losses in the network, while (14) represents the increasing/decreasing ramps and minimum and maximum production limits of each unit \( i \). In (15) are represented the prohibited operation zones, due to vibration in the shafts or problems with the continuous start and stop of the coal mills. Each branch defines the limits of each prohibited operation zone. The formulation (16) and (17) states the spinning reserve which quantify the amount of power that each on-line unit must reverse to be used in case of unexpected change in load or production profile.

\[
\begin{align*}
\min \sum_{i=1}^{N} c_i P_i^2 + b_i P + a_i + |e_i \times \sin(f_i \times (P_{\text{min}} - P_i))| \\
s.t. \\
\sum_{i=1}^{N} P_i &= P_{\text{Load}} + P_{\text{Loss}} \\
\max(p_{\text{min}}, p_0 - D R_i) &\leq P_i \leq \max(p_{\text{max}}, p_0 + U R_i) \\
\left\{ \begin{array}{l}
p^\text{min} \leq P_i \leq P^\text{LB} \\
p^\text{LB} \leq P_i \leq P^\text{UB}, \quad j = 2, 3, ..., N P_i \\
p^\text{UB} \leq P_i \leq P^\text{max} \\
\end{array} \right. \\
S R_i^\text{max} &\leq (1 - x) \times p^\text{max} \\
\sum_{i=1}^{N} \min(p^\text{max} - P_i, S R_i^\text{max}) &\geq S R_i
\end{align*}
\]  

(12)

(13)

(14)

(15)

(16)

(17)

The total transmission power losses \( P_{\text{Loss}} \) of (13) are function of unit power output and can be characterized by (18).

\[
P_{\text{Loss}} = \sum_{i=1}^{N} \sum_{j=1}^{N} P_i B_{ij} P_j + \sum_{i=1}^{N} B_{0i} P_i + B_{00}
\]

(18)

To demonstrate the behavior and compare the results reached by the 3 metaheuristics, 2 cases studies are presented.
4.1. Case study 1

The first test system consisted of 15 thermal units with prohibited operation zones and ramp limits, feeding a load demand of 2630 MW. The network consisted in 30-buses network characterized by the loss coefficients matrix’s $B_{ij}$, $B_{0j}$ and $B_{00}$, considered from [2]. The units had no valve-point effects and the initial values of each unit were defined by $P_i^0$. All parameters used in this case study are described in [2] and the obtained results are shown in table 4. To investigate the influence of the parameters, in figure 2 is shown the minimum cost to the case study 1 changing the social and cognitive coefficients. As can be seen the choice of these parameters will influence the results, due to that they must be chosen carefully. In the case of PSO, generally, in the majority of the publications a value of $c_1=c_2=2$ is widely used.

Fig. 2. Evolution of the cost in function of social and cognitive coefficients

Table 4. Obtained values with different heuristics. [2]

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Min.</th>
<th>Max.</th>
<th>Avg.</th>
<th>Std</th>
<th>t (s)</th>
<th>Eval.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>$33047.9$</td>
<td>$33782.04$</td>
<td>$33429.72$</td>
<td>$240.26$</td>
<td>$8.57$</td>
<td>9900</td>
</tr>
<tr>
<td>BCO</td>
<td>$33178.2$</td>
<td>$33511.00$</td>
<td>$33313.99$</td>
<td>$97.28$</td>
<td>$76.1$</td>
<td>1600</td>
</tr>
<tr>
<td>CSO</td>
<td>$33137.6$</td>
<td>$33662.04$</td>
<td>$33428.86$</td>
<td>$135.49$</td>
<td>$412.3$</td>
<td>9900</td>
</tr>
</tbody>
</table>

4.2. Case study 2

The second case study consists on a system with 10 thermal units with valve-point effects and multi-fuels with a load demand of 2700 MW without prohibited operation zones, ramp limits, neither power losses in transmission lines [9][10]. Due to the thermal units burn different types of fuels, with different costs, the cost function (12) will take the form of (19) where each branch is the cost for burning the fuel $k$, which depends on decision variable $P_i$.

$$F_i(P_i) = \begin{cases} c_{i1}P_i^2 + b_{i1}P_i + a_{i1} + |e_{i1} \times \sin[f_{i1} \times (P_{i_{min}} - P_{i})]|, & \text{fuel 1, } P_{i_{min}} \leq P_i \leq P_{i_{1}} \\ c_{iK}P_i^2 + b_{iK}P_i + a_{iK} + |e_{iK} \times \sin[f_{iK} \times (P_{i_{min}} - P_{i_k})]|, & \text{fuel } k, P_{i_k} \leq P_i \leq P_{i_{max}} \end{cases}$$ (19)

In spite of these simplifications the problem continues to be non-convex, non-continuous and restricted. In the presented case study, one unit burns 2 types of fuels and the remaining 3 types. If this case study was solved by traditional software packages used by the system operator (where the cost functions must be convex) the non-convex equation (19) should be splitted by fuels, transformed in convex functions and then tested all the combinations. In this case it should test all 59049 combinations, which is a massive number of combinations. Solving with the proposed heuristics, the number of iterations was limited to 500 and with the same parameter of table 2.

Table 5. Obtained values with different heuristics.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Min.</th>
<th>Max.</th>
<th>Avg.</th>
<th>Std</th>
<th>t (s)</th>
<th>Eval.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>$648,13$</td>
<td>$1596,18$</td>
<td>$793,65$</td>
<td>$156,84$</td>
<td>$23,9$</td>
<td>14640</td>
</tr>
<tr>
<td>BCO</td>
<td>$667,01$</td>
<td>$1318,54$</td>
<td>$803,09$</td>
<td>$130,91$</td>
<td>$80,83$</td>
<td>6480</td>
</tr>
<tr>
<td>CSO</td>
<td>$654,63$</td>
<td>$1594,88$</td>
<td>$971,18$</td>
<td>$278,89$</td>
<td>$9,8$</td>
<td>1860</td>
</tr>
</tbody>
</table>
As can be seen in table 5, all algorithms were able to reach a solution within the restrictions set and there was no algorithm that proves to be better than the rest. PSO was able to find the best value and although its average is the best, its standard deviation (a good robustness indicator) isn’t, so it wasn’t the most robust. BCO is the most robust and CSO the one which used less processing resources. In figure 3 a) is shown the behavior of the best fitness of all runs, while in b) is shown the average of all fitness’s. All of the heuristics tested reached acceptable values of minimum cost function and comparable with [2] and [9][10]. In spite of the minimum being a good performance indicator, the average value of all runs gives an idea concerning the expected performance of each algorithm. It should be noticed that BCO and CSO in spite of having reached good results with benchmark functions in table 3, in the case of ED problem did not achieve the lowest value. It means that there isn’t an “ideal” heuristic which can solve any kind of problems. As concluded in many papers, different heuristics can have different behaviors and results when applied to the same functions.

5. Conclusions

In this paper was demonstrated and compared the skills of heuristic techniques based on swarm behavior to solve non-convex problems. An important problem in the field of power systems, the economical dispatch, was also tested reaching coherent results. It was shown that this kind of heuristics can fill the gaps of commercial software packages used in power systems. The effectiveness of an algorithm against another cannot be measured comparing with the number of problems that it solves better. By this way, when an algorithm is evaluated, it should be recognized what kind of problems where its performances are good, in order to characterize the type of problems for which the algorithm is suitable.

References