Immune Algorithm for Solving the Smooth Economic Dispatch Problem

Victoria S. Aragón
LIDIC- Universidad Nacional de San Luis
Ejército de los Andes 950 - (5700) San Luis, ARGENTINA
CONICET

and

Susana C. Esquivel
LIDIC- Universidad Nacional de San Luis
Ejército de los Andes 950 - (5700) San Luis, ARGENTINA

ABSTRACT

In this paper, an algorithm inspired on the T-Cell model of the immune system is presented, it is used to solve Economic Dispatch Problems with smooth objective function. The proposed approach is called IA_EDP_S, which stands for *Immune Algorithm for Economic Dispatch Problem for smooth objective function*, and it uses as differentiation process a redistribution power operator. The proposed approach is validated using five problems taken from the specialized literature. Our results are compared with respect to those obtained by several other approaches.

Keywords: Artificial immune systems , economic dispatch problem , metaheuristics

1. INTRODUCTION

The objective of Economic Dispatch Problem (EDP) is to minimize the total generation cost of a power system while satisfying several constraints associated to the system, such as load demands, ramp rate limits, maximum and minimum limits, and prohibited operating zones. The objective function type (smooth or non smooth) and the constraints which are considered in the problem will determine how hard is to solve the problem.

Over the last years, several methods have been proposed to solve the EDP. They can be divided in three main groups: classical, based on artificial intelligence (AI) and hybrid methods. Classical methods have been proposed to solve EDP, but they suffer from some limitations (for instance, the objective functions and the constraints must be differentiable). On the other hand, modern heuristic algorithms have proved to be able to deal with nonlinear optimization problems, e.g., EDPs. Surveys about these techniques can be found in [14] and [2].

In this paper, we propose an algorithm to solve EDPs which is inspired on the T cells from the immune system. Once the algorithm has found a feasible solution, it applies a redistribution power operator in order to improve the original solution with the aim of keeping such a solution feasible at a low computational cost.

The remainder of this paper is organized as follows. Section 2 defines the economic dispatch problem. In Section 3, we describe our proposed algorithm. In Section 4, we present the test problems used to validate our proposed approach and parameters settings. In Section 5, we present our results and we discuss and compare them with respect to other approaches. Finally, in Section 6, we present our conclusions and some possible paths for future research.

2. PROBLEM FORMULATION

The schedule has to minimize the total production cost and involves the satisfaction of both equality and inequality constraints.

Objective Function

Minimize

$$TC = \sum_{i=1}^{N} F_i(P_i)$$

where TC is the fuel cost, N is the number of generating units in the system, P_i is the power of i^{th} unit (in MW) and F_i is the total fuel cost for the i^{th} unit (in \$/h).

An EDP with a smooth cost function represents the simplest cost function. It can be expressed as a single quadratic function: $Fi(P_i) = a_i P_i^2 + b_i P_i + c_i$, where a_i , b_i and c_i are the fuel consumption cost coefficients of the i^{th} unit.

Constraints

1. Power Balance Constraint: the power generated has to be equal to the power demand required. It is defined as: $\sum_{i=1}^{N} P_i = P_D$

2. Operating Limit Constraints: thermal units

have physical limits about the minimum and maximum power that can generate:$P_{min_i} \leq P_i \leq P_{max_i}$, where P_{min_i} and P_{max_i} are the minimum and maximum power output of the i^{th} unit, respectively.

3. Power Balance with Transmission Loss: some power systems include the transmission network loss, thus Power Balance Constraint equation is replaced by:$\sum_{i=1}^{N} P_i = P_D + P_L$

   The P_L value is calculated with a function of unit power outputs that uses a loss coefficients matrix B, a vector B0 and a value B00:
   $PL = \sum_{i=1}^{N}\sum_{j=1}^{N} P_i B_{ij} P_j + \sum_{i=1} B0_i P_i + B00$

4. Ramp Rate Limits: they restrict the operating range of all on-line units. Such limits indicate how quickly the unit's output can be changed:$max(P_{min_j}, P_j^0 - DR_j) \leq P_j \leq min(P_{max_j}, P_j^0 + UR_j)$, where P_j^0 is the previous output power of the j^{th} unit(in MW) and, UR_j and DR_j are the up-ramp and down-ramp limits of the j^{th} unit (in MW/h), respectively.

5. Prohibited Operating Zones: they restrict the operation of the units due to steam valve operation conditions or to vibrations in the shaft bearing:

   $$\begin{cases} P_{min_i} \leq P_i \leq P_{i,1}^l \\ P_{i,j1}^u \leq P_i \leq P_{i,j}^l, & j = 2, 3, ..., nj \\ P_{i,nj}^u \leq P_i \leq P_{max_i} \end{cases}$$

   where nj is the number of prohibited zones of the i^{th} unit, $P_{i,j}^l$ and $P_{i,j}^u$ are the lower and upper bounds of the j^{th} prohibited zone.

3. OUR PROPOSED ALGORITHM

In this paper, an adaptive immune system model based on the immune responses mediated by the T cells is presented. These cells present special receptors on their surface called T cell receptors (TCR: are responsible for recognizing antigens bound to major histocompatibility complex (MHC) molecules.) [6].
The model considers some processes that T cells suffer. These are proliferation (to clone a cell) and differentiation (to change the clones so that they acquire specialized functional properties); this is the so-called activation process.
IA_EDP_S (Immune Algorithm for Economic Dispatch Problem with Smooth Objective Function) is an adaptation of an algorithm inspired on the activation process [2], which is proposed to solve the EDP with Smooth Objective Function. IA_EDP_S operates on one population which is composed of a set of T cells.

For each cell, the following information is kept:

1. TCR: it identifies the decision variables of the problem ($TCR \in \Re^N$). Each thermal unit is represented by one decision variable.

2. $objective$: objective function value for TCR, ($TC(TCR)$).

3. $prolif$: it is the number of clones that will be assigned to the cell, it is N for all problems.

4. $differ$: it is the number of decision variables that will be changed when the differentiation process takes place (if applicable).

5. TP: it is the power generated by TCR ($\sum_{i=1}^{N} TCR_i$).

6. P_L: it is the transmission loss for TCR (if the problem does not consider transmission loss, then $P_L = 0$).

7. ECV: it is the equality constraint violation for TCR ($\mid TP - P_D - P_L \mid$). If $ECV > 0$, then the power generated is bigger than the demanded power, and if $ECV < 0$ then the power generated is lower than the required power.

8. ICS: it is the inequality constraints sum, $\sum_{i=1}^{nj} poz(TCR_i, i)$, where $poz(p, i) =$
   $\begin{cases} \min(p - PZ_{ll_i}, PZ_{ul_i} - p) & if p \in PZ \\ 0 & otherwise \end{cases}$
   where nj is the number of prohibited operating zones and $PZ = [PZ_{ll_i}, PZ_{ul_i}]$ is the prohibited range for the i^{th} thermal unit.

9. feasible: it indicates if the cell is feasible or not. A cell is considered as feasible if: 1) $ECV = 0$ for problems without transmission network loss and $0 \leq ECV < \epsilon$ for problems with transmission loss. This means that if a solution generates less than the demanded power, then it is considered as infeasible ($ECV < 0$) and 2) $ICS = 0$ for problems which consider prohibited operating zones.

Differentiation for feasible cells - Redistribution Process: The idea is to take a value (called d) from one unit (say i) and assign it to another unit (variable). i^{th} unit is modified according to: $cell.TCR_i = cell.TCR_i - d$, where $d = U(prob * D, D)$, $D = \min(cell.TCR_i - ll_i, U(min, max)))$, $U(w_1, w_2)$ refers to a random number with a uniform distribution in the range (w_1,w_2), max is the maximum power that can be generated by the other units according to their current outputs (i.e. $max = \max_{n=1 \wedge n \neq i}^{N}(ul_n - cell.TCR_n)$, min is the minimum power that can be generated by the other units according to their

current outputs (i.e. $min = \min_{n=1 \wedge n \neq i}^{N}(ul_n - cell.TCR_n)$).
d was designed to avoid: 1) that the i^{th} unit falls below its lower limit and 2) to take from the i^{th} unit more power of what other units can generate. Next, d has to increase the power of another unit (say k). In a random way k is selected considering $cell.TCR_k + d \leq ul_k$.
The main difference between IA_EDP_S and the algorithm proposed in [2] arises in the number of variables that are modified. This version just changes i and k while version [2] changes i and one o more variables. Note this operator only preserve the feasibility of solutions by taking into account the power balance constraints.

Differentiation for infeasible cells: For infeasible cells, the number of decision variables to be changed is determined by their differentiation level. This level is calculated as $U(1, N)$. Each variable to be changed is chosen in a random way and it is modified according to: $cell.TCR_i^{'} = cell.TCR_i \pm m$, where $cell.TCR_i$ and $cell.TCR_i^{'}$ are the original and the mutated decision variables, respectively. $m = U(0,1) * \mid cell.ECV + cell.ICS \mid$. In a random way, it decides if m will be added or subtracted to $cell.TCR_i$. If the procedure cannot find a TCR_i' in the allowable range, then a random number with a uniform distribution is assigned to it ($cell.TCR_i^{'} = U(cell.TCR_i, ul_i)$ if m should be added or $cell.TCR_i^{'} = U(ll_i, cell.TCR_i)$, otherwise).
The algorithm works in the following way (see Algorithm 1). First, the TCRs are randomly initialized within the limits of the units (Step 1). Then, ECV and ICS are calculated for each cell (Step 2). Only if a cell is feasible, its objective function value is calculated (Step 3). Next, while a predetermined number of objective function evaluations had not been reached or if after 50 iterations the best value does not improve (Steps 4-6) the cells are proliferated and differentiated considering if they are feasible or infeasible. Finally, statistics are calculated (Step 8).

Algorithm 1 IA_EDP_S Algorithm

```
1: Initialize_Population();
2: Evaluate_Constraints();
3: Evaluate_Objective_Function();
4: while A predetermined number of evaluations has not
   been reached or Not improve do
5:     Proliferation_Population();
6:     Differentiation_Population();
7: end while
8: Statistics();
```

4. VALIDATION

IA_EDP_S performance was validated with five test problems, SYS_3U, SYS_6U, SYS_15U, SYS_18U and SYS_20U (see [2] for full description). Table 1 provides their most relevant characteristics and the maximum number of function evaluations (Eval). IA_EDP_S was implemented in Java (version 1.6.0_24) and the experiments were performed in an Intel Q9550 Quad Core processor running at 2.83GHz and with 4GB DDR3 1333Mz in RAM.

Table 1: Test Problems Characteristics - PZ indicates if prohibited zones are considered

Problem	N	P_L	PZ	P_D (MW)	Eval
SYS_3U	3	No	No	850.0	1000
SYS_6U	6	Yes	Yes	1263.0	3000
SYS_15U	15	Yes	Yes	2630.0	20000
SYS_18U	18	No	No	365.0	40000
SYS_20U	20	Yes	No	2500.0	20000

The required parameters by IA_EDP_S are: size of population, number of objective function evaluations, and probability for redistribution operator. To analyze the effect of the first and third parameters on IA_EDP_S's behavior, we tested it with different parameters settings. Some preliminary experiments were performed to discard some values for the population size parameter. Hence, the selected parameter levels were: a) Population size (C) has four levels: 1, 5, 10 and 20 cells and b) Probability has three levels: 0.01, 0.1 and 0.5. Thus, we have 12 parameters settings for five problems. They are identified as C<size>-Pr<Prob>, where C and Pr indicate the population size and the probability, respectively. For each problem, 100 independent runs were performed.

The box plot method was selected to visualize the distribution of the objective function values for each power system. This allowed us to determine the robustness of our proposed algorithm with respect to its parameters. Figure 1 shows in the x-axis the parameter combinations and the y-axis indicates the objective function values for each problem. We can see that better results are reached with the lowest probability value and the highest population size. So, C=5 and Pr=0.01 were used to compare the results got by IA_EDP_S with those produced by other approaches.

Considering the lowest number of objective function evaluations used by the other approaches (see [2]) we take as maximum number of function evaluations, 1000, 40000, 3000, 20000 and 20000 for SYS_3U, SYS_18U, SYS_6U, SYS_15U and SYS_20U, respectively. Also, we set ϵ=0.1 for those problems which consider loss transmission (e.d. SYS_6U, SYS_15U and SYS_20U).

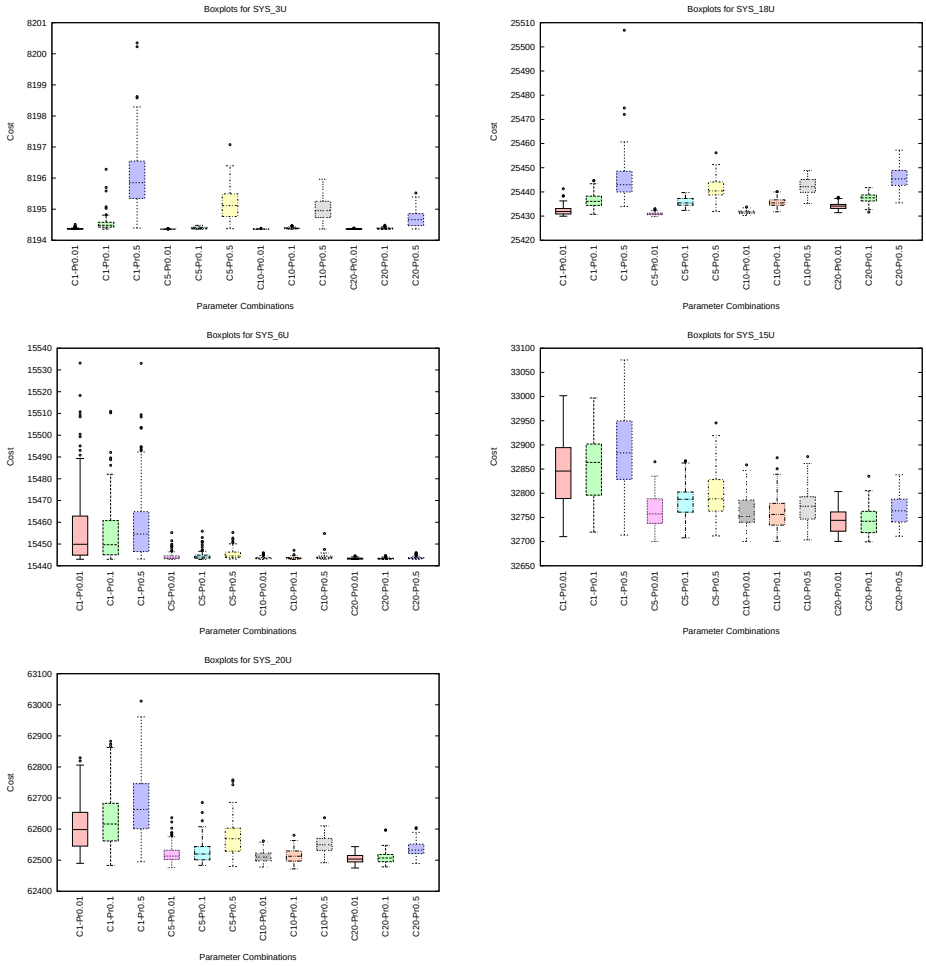

Figure 1: Box plots for the test problems with the best parameters combination

5. COMPARISON OF RESULTS AND DISCUSSION

Table 2 shows: the best, worst, mean, median, standard deviation and number of function evaluations obtained by IA_EDP_S. Only four decimal digits are shown due to space restrictions. For all the test problems, our proposed IA_EDP_S found feasible solutions in all the runs performed.

Problems which do not consider transmission loss, rate ramp limits or prohibited zones, i.e., SYS_3U and SYS_18U, do not seem to be a challenge for IA_EDP_S. The standard deviations obtained by IA_EDP_S are lower than 1. Additionally, the problem dimensionality does not seem to affect the performance of our proposed approach either. For problems which consider transmission loss, rate ramp limits and prohibited zones, SYS_6U_a and SYS_15U, the standard deviations increase with the problem dimensionality.

For the only problem which considers transmission loss but not rate ramp limits or prohibited zones, SYS_20U, the standard deviation is lower than SYS_15U's standard deviation.

Eleven methods are compared with respect to IA_EDP_S. They are cited in Table 3. The running time of each algorithm is affected by both the hardware environment and the software environment. That is the reason why the main comparison criterion that we adopted for assessing efficiency was the number of objective function evaluations performed by each approach. For having a fair comparison of the running times of all the algorithms considered in our study, they should all be run in the same software and hardware environment (something that was not possible in our case, since we do not have the source code of several of them). Clearly, in our case, the emphasis is to identify which approach requires the lowest number of objective function evaluations to find solutions of a certain acceptable quality.

However, the running times are also compared in an indirect manner, to give at least a rough idea of the complexities of the different algorithms considered in our comparative study. For all test problems IA_EDP_S found the best cost in the lowest time. Except for SYS_3U, where fast-PSO just required 0.01 second and IA_EDP_S spent 0.18 seconds to find the best solution.

Table 3 summarizes the performance IA_EDP_S

Table 2: Results obtained by IA_EDP_S

Problem	Best	Worst	Mean	Median	SD.	Ev.
SYS_3U	8194.3561	8194.3784	8194.3597	8194.3584	0.004	987.16
SYS_18U	25429.8005	25433.0655	25430.9312	25430.8415	0.614	35103.15
SYS_6U	15442.8962	15455.2466	15444.3082	15443.6071	1.877	1490.62
SYS_15U	32700.2971	32865.2657	32763.5364	32758.1897	35.765	18321.5
SYS_20U	62476.1186	62636.5875	62522.3703	62513.2753	30.371	8151.36

Table 3: Comparison of results. The best values are shown in **boldface**.

Problem/ Algorithm	Best	Worst	Mean	Std.	Time(s)	Ev.
SYS_3U						
IEP[10]	**8194.35**	-	-	-	-	-
MPSO[9]	**8194.35**	-	-	-	-	-
IPSO [11]	**8194.35**	-	-	-	0.42	3000
ModPSO [12]	8194.40	-	-	-	-	-
fast-CPSO[4]	**8194.35**	-	-	-	0.01	3000
IA_EDP_S	**8194.35**	**8194.37**	**8194.35**	0.004	0.18	987
SYS_18U						
ICA-PSO [13]	25430.16	25462.34	25440.89	-	18.585	40000
IA_EDP_S	**25429.80**	**25433.06**	**25430.93**	0.614	1.168	35103
SYS_6U						
IHS[7]	15444.30	-	15449.86	4.531	-	100000
BBO [3]	15443.09	**15443.09**	**15443.09**	-	-	50000
ICA-PSO[13]	15443.24	15444.33	15443.97	-	-	20000
IA_EDP_S	**15442.89**	15455.24	15444.30	1.877	0.828	1490
SYS_15U						
CCPSO[8]	32704.45	**32704.45**	**32704.45**	0.0	16.2	30000
MDE[1]	32704.9	32711.5	32708.1	-	-	160000
SA-PSO [5]	32708.00	32789.00	32732.00	18.025	12.79	20000
IA_EDP_S	**32700.29**	32865.26	32763.53	35.76	1.328	18321
SYS_20U						
IA_EDP_S	**62476.11**	62636.58	62522.37	30.371	2.016	8151

with respect to that of the other methods. As shown in Table 3, considering the best cost found, IA_EDP_S outperforms all other approaches. Considering running times, IA_EDP_S requires less than one second to find solutions with an acceptable quality for SYS_3U and SYS_6U. It requires less than 1.4 second for SYS_15U and SYS_18U. And it requires less than 2.1 second for SYS_20U.

We could not found an approach that report feasible solutions for SYS_20U, so IA_EDP_S obtained the best results.

6. CONCLUSIONS AND FUTURE WORK

This paper presented an adaptation of an algorithm inspired on the T-Cell model of the immune system, called IA_EDP_S, which was used to solve economic dispatch problems. IA_EDP_S is able to handle the five types of constraints that are involved in an economic dispatch problem: power balance constraint with and without transmission loss, operating limit constraints, ramp rate limit constraint and prohibited operating zones.

At the beginning, the search performed by IA_EDP_S is based on a simple differentiation operator which takes an infeasible solution and modifies some of its decision variables by taking into account their constraint violation. Once the algorithm finds a feasible solution, a redistribution power operator is applied. This operator modifies two decision variables at a time, it decreases the power in one unit, and it selects other unit to generate the power that has been taken.

The approach was validated with five test problems having different characteristics and comparisons were provided with respect to some approaches that have been reported in the specialized literature. Our results indicated that dimensionality increases standard deviations when the same types of constraints are considered but prohibited zones have more impact on the performance than dimensionality. Our proposed approach produced competitive results in all cases, being able to outperform the other approaches while performing a lower number of objective function evaluations than the other approaches.

As part of our future work, we are interested in

redesigning the redistribution operator in order to maintain the solutions' feasibility when a problem involves prohibited operating zones.

7. REFERENCES

[1] N. Amjady and H. Sharifzadeh. Solution of non-convex economic dispatch problem considering valve loading effect by a new modified differential evolution algorithm. *International Journal of Electrical Power and Energy Systems*, 32(8):893–903, 2010.

[2] V.S. Aragon, S.C. Esquivel, and C.A. Coello Coello. An immune algorithm with power redistribution for solving economic dispatch problems. *Information Sciences*, 295(0):609 – 632, 2015.

[3] A. Bhattacharya and P.K. Chattopadhyay. Biogeography-based optimization for different economic load dispatch problems. *IEEE Transactions on Power Systems*, 25(2):1064–1077, 2010.

[4] Leticia Cecilia Cagnina, Susana Cecilia Esquivel, and Carlos A. Coello Coello. A fast particle swarm algorithm for solving smooth and non-smooth economic dispatch problems. *Engineering Optimization*, 43(5):485–505, 2011.

[5] Cheng-Chien Kuo. A novel coding scheme for practical economic dispatch by modified particle swarm approach. *IEEE Transactions on Power Systems*, 23(4):1825–1835, 2008.

[6] Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, New York, 2002.

[7] V.R. Pandi, K.B. Panigrahi, M.K. Mallick, A. Abraham, and S. Das. Improved harmony search for economic power dispatch. In *Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on*, volume 3, pages 403–408, 2009.

[8] J.-B. Park, Y.-W. Jeong, J.-R. Shin, and K.Y. Lee. An improved particle swarm optimization for nonconvex economic dispatch problems. *IEEE Transactions on Power Systems*, 25(1):156–166, 2010.

[9] Jong-Bae Park, Ki-Song Lee, Joong-Rin Shin, and K.Y. Lee. A particle swarm optimization for economic dispatch with nonsmooth cost functions. *Power Systems, IEEE Transactions on*, 20(1):34–42, 2005.

[10] Y. M. Park, J. R. Won, and J. B. Park. A new approach to economic load dispatch based on improved evolutionary programming. *Eng. Intell. Syst. Elect. Eng. Commun.*, 6(2):103–110, June 1998.

[11] P. Sriyanyong, Y.H. Song, and P.J. Turner. Particle swarm optimisation for operational planning: Unit commitment and economic dispatch. In KeshavP. Dahal, KayChen Tan, and PeterI. Cowling, editors, *Evolutionary Scheduling*, volume 49 of *Studies in Computational Intelligence*, pages 313–347. Springer Berlin Heidelberg, 2007.

[12] S. Siva Subramani and P. Raja Rajeswari. A modified particle swarm optimization for economic dispatch problems with nonsmooth cost functions. *International Journal of Soft Computing*, 3(4):326–332, 2008.

[13] J.G. Vlachogiannis and K.Y. Lee. Economic load dispatch a comparative study on heuristic optimization techniques with an improved coordinated aggregation-based pso. *IEEE Transactions on Power Systems*, 24(2):991–1001, 2009.

[14] Ling Wang and Ling po Li. An effective differential harmony search algorithm for the solving non-convex economic load dispatch problems. *International Journal of Electrical Power & Energy Systems*, 44(1):832 – 843, 2013.