Immune Algorithm for Solving the Dynamic Economic Dispatch Problem

Victoria S. Aragón[1,2] and Susana C. Esquivel[1]

[1]Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis -Ejército de los Andes 950 - (5700) San Luis,
ARGENTINA
[1,2] CONICET, Universidad Nacional de San Luis, Argentina

Abstract. In this paper, an algorithm inspired on the immune system is presented, IA_DED stands for *Immune Algorithm Dynamic Economic Dispatch*, it is used to solve the Dynamic Economic Dispatch problem. IA_DED uses as differentiation process a redistribution power operator and the output power are integer values. The proposed approach is validated using three problems taken from the specialized literature. Our results are compared with respect to those obtained by several other approaches.

Keywords: Artificial immune systems , dynamic economic dispatch problem , metaheuristics

1 Introduction

One of the early problems in power system optimization is the Dynamic Economic Dispatch (DED) problem. Its main objective is to determine the optimal schedule of output powers of on line generating units, over a certain period of time (T intervals), to meet power demands at minimum operating cost [10]. Besides, several constraints associated to the system have to be satisfied, such as load demands, ramp rate limits, maximum and minimum limits, and prohibited operating zones. As DED problem is nonlinear different heuristics have been used to solve it, such as, artificial immune system [8], genetic algorithm [7], particle swarm optimizer [7], algorithm bee colony [7], harmony search [14], [12], [2], imperialist competitive algorithm [11], an hybridized differential evolution [3], among others. Surveys about these techniques can be found in [16] and [9].

In this paper, we propose an algorithm to solve DED problem which is inspired on the immune system. Considering T intervals, the problem is regarded as a sequence of T problems. But, each problem (at time i) depends on its predecessor (at time $i-1$) and it conditions to its successor (at time $i+1$). The algorithm applies a redistribution power operator in order to improve a solution at time t 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 DED 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 DED problem minimizes the total production cost (TC) associated with N dispatch units for a time period:

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

where TC is the fuel cost over the whole dispatch period, T is the number of intervals in the period, N is the number of generating units in the system, P_i^t is the power of i^{th} unit at time t (in MW) and F_i is the fuel cost for the i^{th} unit (in \$/h).

A smooth fuel cost function can be expressed as a single quadratic function:

$$F_i(P_i^t) = a_i(P_i^t)^2 + b_i P_i^t + c_i \tag{2}$$

where a_i , b_i and c_i are the fuel consumption cost coefficients of the i^{th} unit. But, if the valve-point effects are taking into account, the fuel cost function of the i^{th} unit is expressed as the sum of a quadratic and a sinusoidal function in the form:

$$F_i(P_i^t) = a_i(P_i^t)^2 + b_i P_i^t + c_i + \mid e_i sin(f_i(P_{min_i} - P_i^t)) \mid \tag{3}$$

where e_i and f_i are the fuel cost coefficients of the i^{th} unit with valve-point effects.

Regardless the function considered (Eq. 2 or Eq. 3), its minimization is subjected to:

1. Power Balance Constraint: the power generated has to be equal to the power demand required. It is defined as:

$$\sum_{i=1}^{N} P_i^t - P_D^t - P_L^t = 0 \tag{4}$$

where $t = 1, 2, \ldots, T$. P_D^t is the total power demand at time t, and P_L^t is the transmission power loss at time t (in MW). P_L^t is calculated using the B-matrix loss coefficients, and the general form of the loss formula using B-coefficients is:

$$P_{Lt} = \sum_{i=1}^{N} \sum_{j=1}^{N} P_i^t B_{ij} P_j^t + \sum_{i=1} B0_i P_i^t + B00 \tag{5}$$

If transmission power loss is not considered, $P_L^t = 0$.

2. Operating Limit Constraints: units have physical limits about the minimum and maximum power they can generate:

$$P_{min_i} \leq P_i^t \leq P_{max_i} \tag{6}$$

where P_{min_i} and P_{max_i} are the minimum and maximum power output of the i^{th} unit in MW, respectively.

3. 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:

$$\begin{cases} P_j^t - P_j^{(t-1)} \leq UR_j \;\; if \; P_j^t > P_j^{(t-1)} \\ P_j^{(t-1)} - P_{jt} \leq DR_j \; if \; P_j^t < P_j^{(t-1)} \end{cases} \tag{7}$$

where $P_j^{(t-1)}$ is the output power of j^{th} unit at previous hour and UR_j and DR_j are the ramp-up and ramp-down limits of the j^{th} unit in MW, respectively. Due to ramp-rate constraints, Eq. 6 is modified as:

$$max(P_{min_j}^t, P_j^{(t-1)} - DR_j) \leq P_j^t \leq min(P_{max_j}^t, P_j^{(t-1)} + UR_j) \tag{8}$$

such that

$$\begin{cases} P_{min_j}^t = max(P_{min_j}, P_j^{(t-1)} - DR_j) \\ P_{max_j}^t = min(P_{max_j}, P_j^{(t-1)} + UR_j) \end{cases} \tag{9}$$

4. Prohibited Operating Zones: they restrict the operation of the units due to steam valve operation conditions or to vibrations in the shaft bearing. Thus, an unit with prohibited operating zones has a discontinuous input-output power generation characteristic which gives rise to additional constraints on the unit operating range.

$$\begin{cases} P_{min_i} \leq P_i^t \leq PZ_{i,1}^L \quad or \\ PZ_{i,k-1}^U \leq P_i^t \leq PZ_{i,k}^L, \, or \\ PZ_{i,n_1}^U \leq P_i^t \leq P_{max_i} \quad k = 2, 3, ..., n_i \end{cases} \tag{10}$$

where n_i is the number of prohibited zones of the i^{th} unit, k is the index of the prohibited operating zones of the i^{th} unit. $PZ_{i,k}^L$ and $PZ_{i,k}^U$ are the lower and upper bounds of the k^{th} prohibited operating zones of unit i.

3 Our Proposed Algorithm

In this paper, an adaptive immune system model based on the immune responses mediated by the T cells from the immune system is presented. These cells present special receptors on their surface called T cell receptors (TCR), they are responsible for recognizing antigens bound to major histocompatibility complex (MHC) molecules.) [13].

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_DED (Immune Algorithm for Dynamic Economic Dispatch problem) is an adaptation of an algorithm inspired on the activation process [1], which was proposed to solve the economic dispatch problem. IA_DED 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 ($TCR \in \aleph^N$). Each thermal unit is represented by one decision variable, each variable is encoded by an integer value.
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). This level is calculated as $U(1, N)$.
5. TP: it is the power generated by TCR ($\sum_{i=1}^{N} TCR_i$).
6. P_L^t: it is the transmission loss for TCR, according to Eq. 5.
7. ECV: it is the equality constraint violation for TCR. At t time ($\mid TP - P_D - P_L \mid$). 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, $\sum_{i=1}^{N} \sum_{j=1}^{n_i} poz(TCR_i, i, j)$
   $poz(p, i, j) = \begin{cases} \min(p - PZ_{i,j}^L, PZ_{ij}^U - p) \; if p \in [PZ_{i,j}^L, PZ_{ij}^U] \\ 0 \qquad\qquad otherwise \end{cases}$
   where n_i is the number of prohibited operating zones and $[PZ_{i,j}^L, PZ_{ij}^U]$ is the j^{th} prohibited range for the i^{th} 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 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 (say j). i^{th} and j^{th} units are modified according to: $cell.TCR_i = cell.TCR_i - d$ and $cell.TCR_j = cell.TCR_j + d$, where $d = U(1, (int)\rceil(P_1 * min(cell.TCR_i - P_{min_i}^t, P_{max_j}^t - cell.TCR_j)))$, $U(w_1, w_2)$ refers to a random

number with a uniform distribution in the range (w_1,w_2) and P_1 is a change factor ($P_1 \in [0, 1]$). Besides, a probability, P_2, determines if i^{th} and j^{th} units will be modified.

Differentiation for infeasible cells

For infeasible cells, the number of decision variables to be changed is determined by their differentiation level ($differ$). 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. $m = U(1, (int)](cell.ECV + cell.ICS))$. 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 ($cell.TCR_i^{'} = U(cell.TCR_i, P^t_{max_i})$ if m should be added or $cell.TCR_i^{'} = U(P^t_{min_i}, cell.TCR_i)$, otherwise). If the resulting clon is feasible then differentiation process stops, otherwise, the process is applied to the resulting clon instead the original infeasible cell. This methodology follows until $prolif$ differentiations have been applied or a feasible clon has been reached.

Algorithm 1 IA_DED Algorithm

```
1: P ← Initialize_Population();
2: Evaluate_Constraints(P);
3: Evaluate_Objective_Function(P);
4: for t ← 1 to T do
5:    top ← 0;
6:    while A predetermined number of evaluations has not been reached and top <
      5 * 10^7  do
7:       Proliferation_Population(P);
8:       Differentiation_Population(P);
9:       top + +;
10:   end while
11:   best_t ← Search_best_at_Population(P, t);
12:   t + +;
13:   Update_limits(best_t);
14:   Repair_output_power(P);
15:   Update_output_power(P);
16:   Evaluate_Constraints(P);
17:   Evaluate_Objective_Function(P);
18: end for
19: BEST ← (best_1, best_2, ..., best_T);
```

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, the following steps are repeated T times (Step 5 to 17): while a predetermined number of objective function evaluations had not been reached and $5{*}10^7$ iterations are not performed

the cells are proliferated and differentiated according to their feasibility. After activation process, best solution at t time is recorded. The time is increased and new operational limits are updated according Eq. 3. Those units which outputs power falling out the new operational limits are changed by random values from the valid limits. Finally, (Step 19) the whole final solution is sequence of solutions found in time 1, time 2, to time T.

4 Validation

IA_DED performance was validated with three test problems, 5-unit system [11], 15-unit system [5], 54-unit system [4]. Table 1 provides their most relevant characteristics and the maximum number of function evaluations. IA_DED 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.

The required parameters by IA_DED are: size of population, maximum number of objective function evaluations, change factor (P_1) and probability for redistribution operator (P_2). To analyze the effect of the first and third parameters on IA_DED'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: 5, 10 and 50 cells, b) Probability P_1 has three levels: 0.1, 0.5 and 0.9 and c) Probability P_2 has two levels: 0.01 and 0.1.

Thus, we have 18 parameters settings for three problems. They are identified as C<size>-P1<Prob>-P2<Prob>, where C, P1 and P2 indicate the population size and the probabilities, 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 show in the x-axis the parameter combinations and the y-axis indicates the objective function values for each problem. We can see for 5-unit system and 15-unit system the results are robust. For 54-unit system, to increase the change factor improve the results and to increase the probability of application of the redistribution process and size population deteriorate the results. So, the settings parameters were used to compare the results got by IA_DED with those produced by other approaches are: for 5-unit system C=10 and P_1=0.1, P_2=0.1, for 15-unit system C=50 and P_1=0.9, P_2=0.1, for 54-unit system C=5 and P_1=0.9, P_2=0.01. Also, we set ϵ=2.0 for those problems which consider loss transmission.

5 Comparison of Results and Discussion

Several methods are compared with IA_DED. They are listed next with the maximum number of function evaluations performed (if the value was available): AIS [8] (300000), GA [7] (not found), PSO [7] (not found), ABC [7] (not found), HS [14] (50000), ICA [11], for 5-unit system, (20000), DE-SQP [3] (50000), NPAHS

Table 1. Test Problems Characteristics

Problem	Thermal Units	Objective	P_L	Prohibited Zones	P_D (MW)	Evaluations
5-unit system	5	non-smooth	Yes	No	14577	19000
15-unit system	15	smooth	Yes	No	60981	19000
54-unit system	54	non-smooth	No	Yes	111600	30000

[12] (50000) CSADHS [2] (250000), SAMF [6] (not indicated), OCD [15] (not indicated), ICA [11] (80000), for 54-unit system.

Table 2 shows: the best, worst, mean, standard deviation and running times obtained by the approaches. For IA_DED only four decimal digits are shown due to space restrictions. For all the test problems, our proposed IA_DED found feasible solutions in all the runs performed.

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. Analyzing Table 2, IA_DED, for 5-unit system, is outperformed by ICA [11] and DE-SQP [3], but it can found quickly an acceptable solutions performing less evaluations of objective function. For 15-unit system, IA_DED outperformed all approaches which we compare it, it found the best solution both running time and total fuel cost. For 54-unit system, IA_DED outperformed all approaches which we compare it, taking into account the total fuel cost. It need 8.565s to find this solution, however it costs $54547 less than OCD's solution.

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_DED, which was used to solve dynamic economic dispatch problems. IA_DED is able to handle the five types of constraints that are involved in this kind of problems: power balance constraint with and without transmission loss, operating limit constraints, ramp rate limit constraint and prohibited operating zones, and different types of objective function: smooth and non-smooth.

Table 2. Comparison of results. The best values are shown in **boldface**. - denotes that the value was not available in the literature.

Problem/ Algorithm	Best	Worst	Mean	Std.	Time(s)
5-unit system					
IA_DED	43716.6386	46611.1333	44879.9749	649.60	**2.228s**
AIS [8]	44385.43	45553.7707	44758.8363	-	4min
GA [7]	44862.42	-	-	-	-
PSO [7]	44253.24	-	-	-	-
ABC [7]	44045.83	-	-	-	-
HS [14]	44376.23	-	-	-	2.8min
ICA [11]	43117.055	**43209.533**	**43144.472**	19.821	
DE-SQP [3]	**43161**	-		-	-
15-unit system					
IA_DED	**759385.9148**	**759665.1300**	**759478.5852**	55.08	**2.08s**
NPAHS [12]	759603.089	759988.390	759779.467	-	250.0
CSADHS [2]	759689.220	759845.739	759766.233	-	3.36min
SAMF [6]	759406.42	-	-	-	2.951s
54-unit system					
IA_DED	**1718177.0643**	**1718695.4568**	**1718424.5910**	130.85	8.565s
OCD [15]	1772724.032	-	-	-	**0.132s**
ICA [11]	1807081.174	1811388.285	1809664.219	-	-

At the beginning, the search performed by IA_DED 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, always integer values.

The approach was validated with three test problems having different characteristics and comparisons were provided with respect to some approaches that have been reported in the specialized literature. Our proposed approach produced competitive results in all cases, being able to outperform some approaches while performing a lower number of objective function evaluations.

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.

Acknowledgements

The authors acknowledge support from Universidad Nacional de San Luis, project no. P330214/22F435.

References

1. 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.
2. R. Arul, G. Ravi, and S. Velusami. Chaotic self-adaptive differential harmony search algorithm based dynamic economic dispatch. *International Journal of Electrical Power & Energy System*, 50:85–96, 2013.
3. A.M. Elaiw, X. Xia, and A.M. Shehata. Hybrid de-sqp and hybrid pso-sqp methods for solving dynamic economic emission dispatch problem with valve-point effects. *Electric Power Systems Research*, 103:192–200, October 2013. ISSN 0378-7796.
4. M. Shahidehpour [Online]. Available from: motor.ece.iit.edu/data/SCUC_118test.xls, March 2009. [accessed July 2016].
5. Z.L. Gaing and Ting-Chia Ou. Dynamic economic dispatch solution using fast evolutionary programming with swarm direction. In *4th IEEE Conference on Industrial Electronics and Applications. ICIEA 2009*, pages 1538–1544, 2009.
6. S. GAnesan and S. Subramani. Dynamic economic dispatch based on simple algorithm. *International Journal of Computer and Electrical Engineering*, 3(2):1793–8163, 2011.
7. S. Hemamalini and S. Simon. Dynamic economic dispatch using artificial bee colony algorithm for units with valve-point effect. *European Transactions on Electrical Power*, 21:70–81, 2011.
8. S. Hemamalini and S. Simon. Dynamic economic dispatch using artificial immune system for units with valve-point effect. *International Journal of Electrical Power & Energy System*, 33:868–874, 2011.
9. C. Kumar and T. Alwarsamy. Dynamic economic dispatch - a review of solution methodologies. *European Journal of Scientific Research*, 64(4):517–537, 2011.
10. R. Kumar, D. Sharma, and A. Sadu. A hybrid multi-agent based particle swarm optimization algorithm for economic power dispatch. *International Journal of Electrical Power and Energy Systems*, 33(1):115–123, 2011.
11. B. Mohammadi-Ivatloo, A. Rabiee A., Soroudi M., and Ehsan. Iteration {PSO} with time varying acceleration coefficients for solving non-convex economic dispatch problems. *International Journal of Electrical Power & Energy Systems*, 42(1):508 – 516, 2012.
12. Qun Niu, Hongyun Zhang, Kang Li, and George W. Irwin. An efficient harmony search with new pitch adjustment for dynamic economic dispatch. *Energy*, 65:25–43, 2014.
13. Leandro Nunes de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, New York, 2002.
14. V.R. Pandi and B.K. Panigrahi. Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm. *Expert Systems with Applications*, 38:8509–8514, 2011.
15. Abbas Rabiee, Behnam Mohammadi-Ivatloo, and Mohammad Moradi-Dalvand. Dynamic economic dispatch: A review. *IEEE Transactions on Power Systems*, 29(2):982–983, 2014.
16. X. Xia and A. M. Elaiw. Dynamic economic dispatch: A review. *The Online Journal on Electronics and Electrical Engineering (OJEEE)*, 2(2):234–245, 2009.

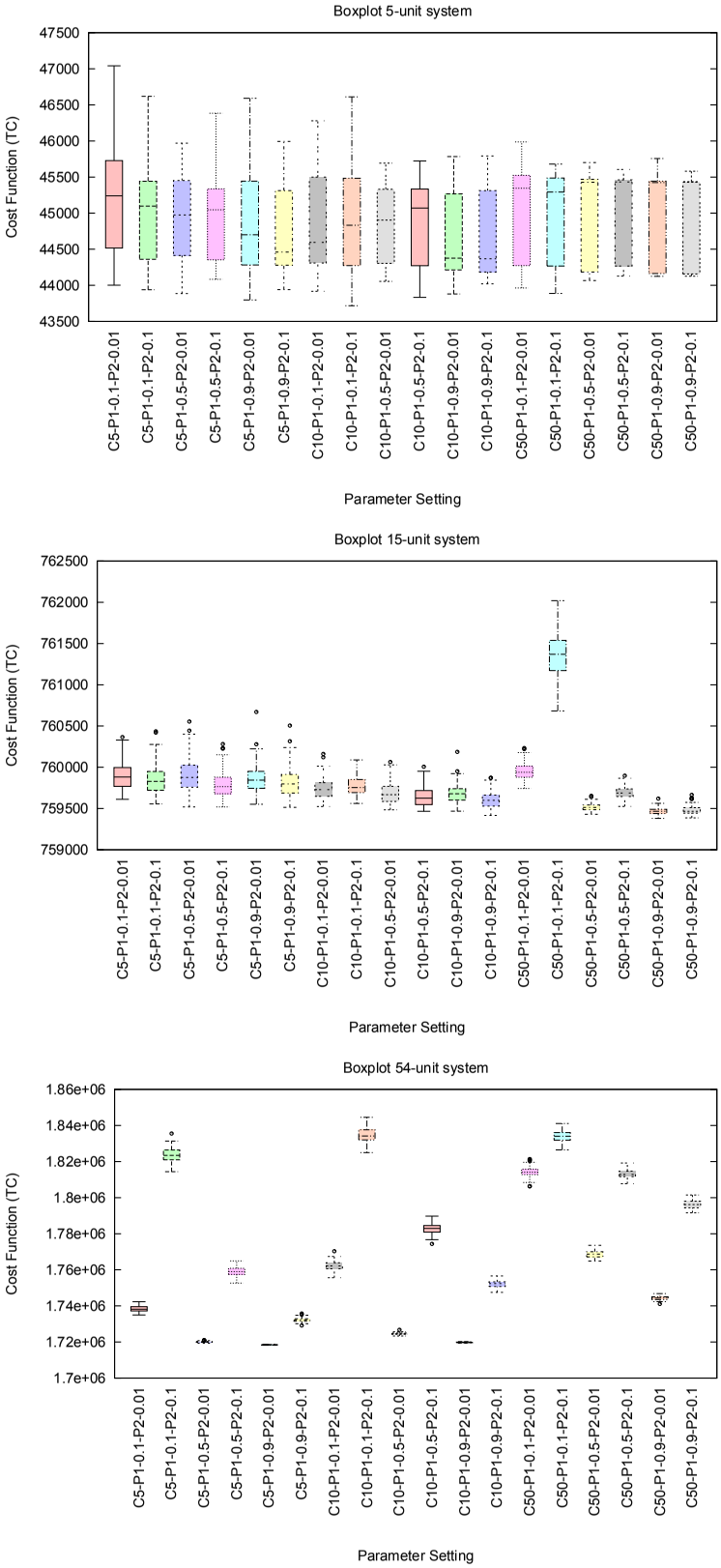

Fig. 1. Box plots for the test problems with the best parameters combination