2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

# A Decomposition-based Algorithm for Dynamic Economic Dispatch Problems

Eman Sayed, Daryl Essam, Ruhul Sarker, Saber Elsayed
School of Engineering and Information Technology
UNSW-Canberra
Canberra, Australia
e.hasan@student.adfa.edu.au, d.essam@adfa.edu.au, r.sarker@adfa.edu.au, s.elsayed@adfa.edu.au

**Abstract**—***Large scale constrained problems are complex problems due to their dimensionality, structure, in addition to their constraints. The performance of EAs decreases when the problem dimension increases. Decomposition-based EAs can overcome this drawback, but their performance would be affected if the interdependent variables were optimized in different subproblems. The use of EAs with variables interaction identification technique handles this issue by identifying better arrangements for decomposing a large problem into subproblems in a way that minimizes the interdependencies between them. The only technique in the literature that has been developed to identify the variables interdependency in constrained problems is the Variable Interaction Identification for Constrained problems (VIIC). This technique is tested in this paper on a real-world problem at three large dimensions which are large scale constrained optimization problems. The performance of the decomposition-based EA that uses VIIC is compared to Random Grouping approach for decomposition, for 5-Units, 10-Units, and 30-Units DED problems.***

***Keywords—Evolutionary Algorithms; Deffirential Evolution; large scale constrained problems; constrained problem decomposition; variables interacntion identificatio; and dynamic economic dispatch problems.***

## I. INTRODUCTION

EAs were used successfully to solve the constrained optimization problems. However, the performance of EAs decreases when the dimensionality of the problem increases. The complexity to model large dimension real-world problems, the noise of their objective function, and the complexity of their constraints are three factors which add to the difficulty of the real-world constrained optimization problems [1]. Decomposing the large scale problems into smaller scale subproblems, and then optimizing them, overcomes this drawback of EA. This decomposition will not be effective unless the interdependent variables of the optimization problem were optimized in common subproblems. Therefore, a dependency identification technique is required to group the interdependent variables into common subproblems.

This paper presents the implementation of a newly proposed Variable Interaction Identification technique for Constrained problems (VIIC) on Dynamic Economic Dispatch problem (DED) at three different dimensions. This technique has been previously successfully tested on 18 problems of varying complexity on three different large scales of 100, 500, and 1000. The problems tested in this paper are DED of 5-Units, 10-Units, and 30-Units. DED problems are challenging problems in the industry of power generating which are large scale and complex problems. There was no single attempt in the literature to solve these problems using the decomposition approach with variable interaction identification. These problems were optimized in the literature in form of unconstrained problem, where the constraints were represented as part of the objective function, without considering any constraint violation information. Therefore, the found solutions may not be feasible. Moreover, representing DED problem in this way, which is large scale problem, and optimizing it without decomposition would be complex in comparison to the decomposition approach.

This paper is organized as follows: Section II is the literature review of the decomposition approaches. Section III discusses the tested problems. The used models are reviewed in section IV. Experiment and results are presented in sections V. Sections VI to VIII discuss the analysis of the results and convergence graphs. Finally, the conclusion and future work in section IX.

## II. LITERATURE REVIEW

Cooperative coevolutionary (CC) [2] is the first approach that was developed to decompose a large scale problem into smaller scale subproblems. One of the techniques decomposes the large scale problem of $N$ dimension into $N$ one-dimension subproblems. Other techniques divide the large problem into two equal size subproblems or into more than two subproblems of equal or different size [3-6]. Most of the decomposition techniques for the constrained problems were developed for problems of specific structure. In 1960, Dantzig and Wolfe developed a decomposition technique for the linear constraints problems that have block-triangular structure [7]. Two years later Benders developed a decomposition approach for mixed-variable optimization problems [8]. Last decade Ryoo, and Hajela proposed a decomposition technique and migration strategy among subproblems [9]. Recently, Elfeky decomposed the large scale problem, that have block-triangular structure, into common problem that contains the objective function and a set of constraints, and subproblems that contain only a subset of constraints, such that no constraint exists in more than one subproblem. Although the decomposition overcome the dimensionality problem, the performance of the decomposition-based EAs decreases when there is interdependency between the subproblems [3]. This interdependency between subproblems should be minimized [4, 10]. This requires using decomposition technique which

is able to detect and group the dependent variables into common subproblems before optimizing them.

In the last decades, some decomposition techniques were developed for the unconstrained problems. Random Grouping (RG) [6] groups the variables randomly into subproblems and it achieved good performance in the literature [6, 11]. One of these techniques is correlation based Adaptive Variable Partitioning [12] which is computationally expensive and is not able to detect nonlinear dependencies of variables. Delta Grouping technique which is a successful technique in identifying dependencies of large scale problems, but it is not very efficient when the problem has more than one nonseparable group [13]. Variable Interactive Learning technique (VIL) [14] starts initially with *N* subproblems which is not suitable for large scale problems specially if it was nonseparable problem. Dependency Identification (DI) create variant groups of the variables and select, between the most recent grouping and the previously best known grouping, the one that achieves the least fitness difference [15, 16]. The most recent technique is Statistical variable interdependence Learning (SL) checks the change that every pair of the decision variables has on each other to quantify the degree of interdependencies among variables [17]. There is only one technique that was developed for decomposing constrained problems based on dependency identification in literature which is VIIC. VIIC is able to find better arrangements of the variables into subproblems that have minimum interdependencies. Although RG does not have a systematic way to detect the dependencies among the variables, it can be implemented directly to the constrained problems. It is tested in this paper against VIIC on DED problems to show the performance of decomposition-based EAs on real-world large scale problems when the interdependencies variables are grouped in common subproblems.

DED gets the cost of generating power from generating units in a given period of time. The demand in DED changes dynamically every hour over a period of 24 hours. DED is a multimodal problem, with nonseparable variables and additional constraints [18]. Although DED is known to be large scale optimization problems, the optimization models which were developed using EAs did not use any decomposition-based technique. Sample of the EAs which were used to optimize DED problems are Genetic Algorithms (GA) [19, 20], Evolutionary Programming (EP) [21, 22], and Differential Evolution (DE) [23-26]. Because of the multimodality and nonseparability of DED, it is a good study case to show the ability of decomposition-based EAs that use variables interdependency identification over those that do not. Three DED problems are tested in this paper, 5-Units, 10-Units, and 30-Units. The dimensions of these three problems are 120, 240, and 720 and the number of constraints is 75, 80, and 100 respectively.

## III. DED PROBLEM

DED problem is one of the challenging industrial large scale constrained optimization problems. Its objective is to minimize the total cost of the power generated from generating units while satisfying the power demand, which varies each hour, and all other constraints. This problem has multimodal search space and its dimension is 24 times larger than the static Economic Load Dispatch problems (ELD), and it has more constraints. The objective function of DED is quadratic function of active power outputs from generating units. Additionally, in the large power plants, where the turbines have steam admission valves, the cost of the valve point effect is added to the objective function [27]. Therefore, the objective function of large scale DED problem is multimodal function when valve point effect is considered and its search space has more local optima [20, 22, 23, 28]. This is the problem that is experimented in this paper, and its objective function is defined as follows:

$$min\, F_c = \sum_{t=1}^{T} \sum_{i=1}^{N_G} F_{it}\,(P_{it}) \qquad (1)$$

$F_{it}\,(P_{it})$ is the cost function of the $i$ generating unit at time $t$ , $F_{it}\,(P_{it}) = a_i P_{it}^2 + b_i P_{it} + c_i + \left|e_i\ sin\left(f_{it}(P_{it}^{min} - Pit,\ i=1,2,3,\ ...,\ NG,\right.\right.$ and *ai*, *bi*, and *ci* are cost coefficients, $P_{it}$ is the power output of generating unit $i$ at time $t$. $e_i$ and $f_i$ are the cost functions of the valve point loading effect. $N_G$ is the number of dispatched generating units (which is 5, 10, 30 in this paper), and $T$ is the total time period of dispatch which is 24.

DED is subject to the following three constraints:

*1- Power balance constraint:*

$$\sum_{i=1}^{N_G} P_{it} - P_{Dt} - P_{Lt} = 0 \qquad (2)$$

$P_{Dt}$ is the total power demand at time $t$, and $P_{Lt}$ is the transmission power loss at time t in MW and is calculated as follows:

$$P_{Lt} = \sum_{i=1}^{N_G} \sum_{j=1}^{N_G} P_{it}\ \beta_{ij}\ P_{jt} \qquad (3)$$

where $\beta$ is the loss coefficients matrix. DED problem has $T$ constraints of this first constraint. The transmission loss coefficients for the 5-Units problem are:

$$\beta = \begin{bmatrix} 0.000049 & 0.000014 & 0.000015 & 0.000015 & 0.000020 \\ 0.000014 & 0.000045 & 0.000016 & 0.000020 & 0.000018 \\ 0.000015 & 0.000016 & 0.000039 & 0.000010 & 0.000012 \\ 0.000015 & 0.000020 & 0.000010 & 0.000040 & 0.000014 \\ 0.000020 & 0.000018 & 0.000012 & 0.000014 & 0.000035 \end{bmatrix} \text{MW.}$$

The transmission loss $P_{Lt}$ is neglected for the 10, and 30 units in this paper, as the problem in [27].

*2- Capacity limit constraint:*

Capacity limit constraints define the range of the power generated from each unit i.

$$P_{i\,min} \le P_i \le P_{i\,max} \quad i = 1, \dots, N_G \qquad (4)$$

$P_{i\,min}$ and $P_{i\,max}$ are the minimum and maximum power outputs of the $i$ generating unit, respectively. DED has $N_G$ Capacity limit constraints.

*3- Ramp rate limits constraint:*

The constraints that keep the thermal gradients inside the turbine with in ramp rate limits increase the life of the units. Pounding the generated power restricts the operating range of all the units $N_G$ to operate only between two limits, the upper ramp rate limit $UR_i$ and the down ramp rate limit $DR_i$.

$$P_{it} - P_{i(t-1)} \le UR_i \quad i = 1, \dots, N_G, t = 1, \dots, T \qquad (5)$$

$$P_{i(t-1)} - P_{it} \le DR_i \quad i = 1, \dots, N_G, t = 1, \dots, T \qquad (6)$$

where $P_{it}$ is the power generated from generating unit $i$ at the time t and $P_{i(t-1)}$ is the power generated from the same uniti at the previous time $t-1$. Each unit in the DED has its own range of power generating. The number of Ramp rate limits constraint in DED problem is $2 \times (T-1)$.

The experiments in the following section use three different problems where $N_G$ is 5, 10, or 30 units. The used data for the 5-Units and the 10-Units are the same as in [21, 25, 29]. The data 5-Units problem is shown in Table 1 and the load demand for 24 hours is shown in Table 2.

The data for the 10-Units problem without loss coefficient are shown in Table 3 and the load demand is presented in Table 4. The data for the 30-Units is three folds of the 10-Units as in Table 3 as the data in [30, 31], and the Load demand for the 30-Units is triple the load demand of the 10-Units in Table 4. The dimensions $N$ of the DED problems are 120, 240, 720 variables, for 5-Units, 10-Units, and 30-Units problems. The number of constraints also increases to 75, 80, and 100 for the three problems respectively. The following section presents a review for DEVIIC and DERGC.

Table 1: Data for 5-Units problem

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $P_{i\,max}(MW)$ | 75 | 125 | 175 | 250 | 300 |
| $P_{i\,min}(MW)$ | 10 | 20 | 30 | 40 | 50 |
| $a_i(\$/h)$ | 25 | 60 | 100 | 120 | 40 |
| $b_i(\$/MWh)$ | 2.0 | 1.8 | 2.1 | 2.0 | 1.8 |
| $c_i(MW^2h)$ | 0.0080 | 0.003 | 0.0012 | 0.0010 | 0.0015 |
| $e_i(\$/h)$ | 100 | 140 | 160 | 180 | 200 |
| $f_i(rad/MW)$ | 0.042 | 0.040 | 0.038 | 0.037 | 0.035 |
| $UR_i(MW)$ | 30 | 30 | 40 | 50 | 50 |
| $DR_i(MW)$ | 30 | 30 | 40 | 50 | 50 |

Table 2: Load demand for 24 hours for 5-Units

| **Hour** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| Demand (MW) | 410 | 435 | 475 | 530 | 558 | 608 | 626 | 654 |
| **Hour** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| Demand (MW) | 690 | 704 | 720 | 740 | 704 | 690 | 654 | 580 |
| **Hour** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| Demand (MW) | 558 | 608 | 654 | 704 | 680 | 605 | 527 | 463 |

Table 3: data for 10-Units problem

| Unit | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $P_{i\,max}(MW)$ | 470 | 460 | 340 | 300 | 243 |
| $P_{i\,min}(MW)$ | 150 | 135 | 73 | 60 | 73 |
| $a_i(\$/h)$ | 958.2 | 1313.6 | 604.97 | 471.6 | 480.29 |
| $b_i(\$/MWh)$ | 21.6 | 21.05 | 20.81 | 23.9 | 21.62 |
| $c_i(MW^2h)$ | 0.00043 | 0.00063 | 0.00039 | 0.0007 | 0.00079 |
| $e_i(\$/h)$ | 450 | 600 | 320 | 260 | 280 |
| $f_i(rad/MW)$ | 0.041 | 0.036 | 0.028 | 0.052 | 0.063 |
| $UR_i(MW)$ | 80 | 80 | 80 | 50 | 50 |
| $DR_i(MW)$ | 80 | 80 | 80 | 50 | 50 |
| Unit | 6 | 7 | 8 | 9 | 10 |
| $P_{i\,max}(MW)$ | 160 | 130 | 120 | 80 | 55 |
| $P_{i\,min}(MW)$ | 57 | 20 | 47 | 20 | 55 |
| $a_i(\$/h)$ | 601.75 | 502.7 | 639.4 | 455.6 | 692.4 |
| $b_i(\$/MWh)$ | 17.87 | 16.51 | 23.23 | 19.58 | 22.45 |
| $c_i(MW^2h)$ | 0.00056 | 0.00211 | 0.0048 | 0.10908 | 0.00951 |
| $e_i(\$/h)$ | 310 | 300 | 340 | 270 | 380 |
| $f_i(rad/MW)$ | 0.048 | 0.086 | 0.082 | 0.098 | 0.094 |
| $UR_i(MW)$ | 50 | 30 | 30 | 30 | 30 |
| $DR_i(MW)$ | 50 | 30 | 30 | 30 | 30 |

Table 4: Load demand for 24 hours for 10-Units problem

| **Hour** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| Demand(MW) | 1036 | 1110 | 1258 | 1406 | 1480 | 1628 | 1702 | 1776 |
| **Hour** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| Demand(MW) | 1924 | 2072 | 2146 | 2220 | 2072 | 1924 | 1776 | 1554 |
| **Hour** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** |
| Demand(MW) | 1480 | 1628 | 1776 | 2072 | 1924 | 1628 | 1332 | 1184 |

## IV. DEVIIC AND DERGC

The optimization models which are used in this paper to optimize the three DED problems are DEVIIC, and DERGC. Both of them use DE for subproblems optimization. DEVIIC uses VIIC for decomposing the large scale problem into smaller subproblems, while DERGC uses RG for decomposition.

### *A. DEVIIC*

DEVIIC is presented in Table 5. DEVIIC initialize a population of $NP$ individuals at generation $Gen = 0$, and each individual has $N$ variables. $S_g$ is the initial arrangement of variables, which represent the variables in a sequential arrangement. The number of different arrangements which are generated is $maxgrp_{iter}$. Each subproblem is optimized for number of iterations ($iter$). The maximum number of fitness evaluations ($max\_FEs$) is predefined before starting the optimization. DEVIIC uses VIIC for decomposition.

Table 5: Basic steps in DEVIIC

**STEP 1 Initialisation:** in generation *Gen* = 0:

- generate an initial random population that has $NP$ individuals and where each individual has $N$ variables. The variable in each individual of the population must be generated within its range by:

  $$x_{i,j} = \underline{L_j} + rand \times \left(\overline{U_j} - \underline{L_j}\right), j \in [1, N], i \in [1, NP]$$

  where $\underline{L_j}$ and $\overline{U_j}$ are the lower and upper bounds of the variable $x_j$ in the individual $j$, and rand is a uniform random number, rand ∈ [0,1].
- identify the number of subproblems, $m$, the subproblem size, $V$, $max\_FEs$

**STEP 2: While** $FEs < max\_FEs$

**STEP 2.1: If** (*Gen* < 25) or ((*Gen*%50) = 0 && (*Gen* < $max\_FEs$ /2)) or (((*Gen*%100) = 0 && (*Gen* > $max\_FEs$ /2))); ***then***

- call VIIC
- find the best arrangement of the N variables ($S_N$)
- save $S_N$ into $Grp\_Pool$

**Else**

- select $S_N$ randomly from $Grp\_Pool$

**STEP 2.2: Problem decomposition**

use $S_N$ to group the variables into $m$ subproblems of size $V$. Each $k$ subproblem contains the number of variables, $V$, where $N = m \times V$.

**STEP 2.3: Subproblems optimization**

use EA to optimize each subproblem $k$, for number of iterations *iter*.

**STEP 2.3: Information Exchange** –

- update the values of subproblem k into the complete solution vector, $S_N = \{x_1, \dots, x_{r_1}, \dots, x_{r_2}, \dots, x_N\}$. So that any $x_j$ that is interdependent among more than one subproblems, where $j \in k \cap [m-k]$, is updated to the latest optimized $x_j$.

**STEP 3: END While**

VIIC uses initial arrangement $S_g$ for objective function and each constraint to produce the final variables arrangement vector $S_N$. $S_N$ is then used to group variables into subproblems that achieve the minimum subproblems interdependency, and maximum variables interdependency. A new $S_N$ is generated every generation for the first 25 generations ($Gen \leq 25$), every 50 generations ($Gen$%50) in the first half of the $FEs$, and every 500 generations ($Gen$%500) in the second half of the $FEs$. $Grp\_pool$ is the repository which contains all the generated $S_N$. The detailed steps for VIIC are in Table 6.

$max\_constr$ is the maximum number of constraints, and $maxgrp_{iter}$ is the times of generating new arrangement. $nc$ is the number of the objective function and constraints which need finding a variable arrangement for, then adding it to the *Frequency Matrix* (*FM*). These $nc$ arrangements can be found using the following calculations as in steps 2.2, 2.3 and 2.4.

Table 6: Basic steps of VIIC

**STEP 1 Initialisation:**
- set $max\_constr = nc$ if no feasible solution has been found or $max\_constr = nc + 1$ otherwise, $maxgrp_{iter} = m \times 10^4$, $V$ is the subproblem size of $m$ subproblems, $c\_num = 0$, $S_N = \emptyset$

**STEP 2: While** $c\_num < max\_constr$

**STEP 2.1:** initialize the starting variables vector $S_g$, generate two random values, $C_1 > 0$ and $C_2 > 0$

**STEP 2.2:** calculate $fit_{all_{C_1C_2}}$ for the fitness function for the subproblems in the arrangement $S_g$ and calculate $fit_{grp_{C_1C_2=k}}$ for all the $k$ subproblems.

**STEP 2.3:** calculate $grps_{diff}$ for the starting arrangement, $S_g$

**STEP 2.4: If** $grps_{diff} \neq 0$ and $grp_{iter} < maxgrp_{iter}$ **; then**
1. generate a new random arrangement of variables $S_{\bar{g}} = \{x_{r_1}, x_{r_2}, \dots, x_{r_{N-10}}, \dots, x_{r_N}\}$
2. calculate $fit_{grp_{C_1C_2=k}}$ for all the $k$ subproblems arranged as in $S_{\bar{g}}$
3. calculate $newgrps_{diff}$ for $S_{\bar{g}}$
4. **If** $newgrps_{diff} < grps_{diff}$; **then** update $S_g = S_{\bar{g}}$ and $grps_{diff} = newgrps_{diff}$
5. ***Go to*** **STEP 2.4**

**STEP 2.5:** add the best arrangement $S_g$ into the $c\_num$ row of *FM* which has $max\_constr$ rows and $N$ columns

**STEP 2.6:** $c\_num = c\_num + 1$

**STEP 3:** starting with $S_N = \emptyset$; *for* $k = 1$ *to* $m$ ***do***
- divide *FM* vertically into $m$ blocks
- record the frequency of all the variables in each block k of *FM* into its corresponding $grp_k$ of *GFM* (there is one $grp_k$ in *GFM* for each *k)*
- count the frequency and record it into the "$total$" row of *GFM*

**STEP 4:** ***for*** $i = 1$ *to* $k$ ***do***
- remove the variables that are in $S_N$, from the "*total*" of the $grp_i$ block in *GFM*
- sort the "*total*" row of the $grp_i$ block in *GFM*
- add the variables, of the highest frequency variable of the $grp_i$ block in *GFM* to $S_N$

**STEP 5:** Return $S_N$

$$fit_{all_{C_1C_2}} = m * [fit_{all_{C_1}} + fit_{all_{C_2}}] \quad (7)$$

$$fit_{all_{C_1}}, x_i = C_1, \forall\ i = [1, N] \quad (8)$$

$$fit_{all_{C_2}}\ x_i = C_2, \forall\ i = [1, N] \quad (9)$$

$$grps_{diff} = \left| fit_{all_{C_1C_2}} - fit_{Groups_{C_1C_2}} \right| \quad (10)$$

$$fit_{Groups_{C_1C_2}} = \sum_{k=1}^{m} fit_{grp_{C_1C_2=k}} \quad (11)$$

$$fit_{grp_{C_1C_2=k}} = fit_{C_1\ grp=k} + fit_{c_2\ grp=k}\ \forall\ k \in m \quad (12)$$

$$x_{c_1} = \begin{cases} c_1\ \forall\ x \in V \\ c_2\ otherwise \end{cases}, \quad (13)$$

$$x_{c_2} = \begin{cases} c_2\ \forall\ x \in V \\ c_1\ otherwise \end{cases}, \quad (14)$$

*where V* is the number of variables in subproblem *k*.

*FM* consists of the best obtained variables arrangements and it is divided into blocks ($grps$) of size $V$. *Groups Frequency Matrix* (*GFM*) contains the frequency of all the variables in each block ($grp$). Its rows represent the occurrence of each decision variable in all the rows of the corresponding block columns in *FM*. Then the highest frequency variables are grouped into one group to get the final grouping vector $S_N$ which is used in the subproblems optimization (step 2 of Table 5) and is saved in $Grp\_Pool$.

DE is used to optimizing all the subproblems. A new generation, "*Gen*", starts after optimizing all the subproblems and the optimization ends when $FEs > max\_FEs$, the optimization process continues and new variables arrangements are used (either from VIIU or $Grp\_Pool$). The DE variant that was used is "*rand/1/bin*". Its mutation and crossover are as follows.

*Mutation:* The mutation vector, $\overrightarrow{V_z}(q)$, at generation $q$, is generated by adding a random vector, $\overrightarrow{x_{r_1}}(q)$, to the multiplication of the amplifier factor, $F$, by the difference of another two random vectors, $\overrightarrow{x_{r_2}}(q)$ and $\overrightarrow{x_{r_3}}(q)$.

$$\overrightarrow{V_z}(q) = \overrightarrow{x_{r_1}}(q) + F.\left(\overrightarrow{x_{r_2}}(q) - \overrightarrow{x_{r_3}}(q)\right) \quad (15)$$

where $r_1, r_2$ and $r_3$ are random number $\in \{1,2, \dots, NP\}$.

*Crossover:* Binomial crossover is applied with probability less than or equal to the Crossover rate *(Cr)*.

$$u_{z,j}(q) = \begin{cases} v_{z,j}(q), & rand < Cr\ or\ j = jrand \\ x_{z,j}(q), & otherwise \end{cases} \quad (16)$$

where $rand \in [0,1]$, and jrand are randomly chosen indexes $\in [1,2,3, \dots, N]$ to guarantee that at least one variable is added to $U_z(q)$ from $V_z(q)$.

*Selection:* The new individuals for generation $q + 1$ are chosen as follows:

$$x_z(q+1) = \begin{cases} u_z(q), & if\ f(u_z(q)) \leq f(x_z(q)) \\ x_z(q), & otherwise \end{cases} \quad (17)$$

### B. DERGC

DERGC follows exactly the same steps of DEVIIC except in **STEP 2.1** (VIIC) which is replaced by RG where $S_N$ is generated randomly at each call for step 2 as long as $FEs < max\_FEs$. RG creates a new random arrangement $S_N$ of $N$ variables, $S_N = \{x_{r_1}, x_{r_2}, \dots, \dots, x_{r_N}\}$, where $r_1$ to $r_N$ are random indices for the variables from 1 to $N$. This random arrangement is decomposed into subproblems $m$ of size $V$ in **STEP 2.2** and optimized sequentially in **STEP 2.3**.

## V. EXPERIMENTS AND RESULTS

Three experiments are carried out in this paper to evaluate the performance of the VIIC against RG on real-world problems. The constraint-handling technique that was used for DED problems in the literature is penalty functions [27, 30, 31]. To test the VIIC technique that is used in DEVIIC, the constraints of DED problems should be handled using different constraint-handling technique than the penalty functions technique. This is because the penalty functions technique adds the constraints with penalty coefficients to the objective function. And to show the ability of VIIC, the objective function and the constraints need to be presented separately. Therefore, the used constraints-handling technique for DED in this paper is the superiority of feasible solution that was used by Deb *et al.* [32]. The individual that achieves the best fitness value from two feasible solutions to be better, any feasible individual is better than an infeasible individual, and for two infeasible individuals; the one that achieves less violation is preferred. The equality constraints $|h_j(\vec{x})| = 0$ are transformed to inequality constraints, $|h_j(\vec{x})| - \epsilon \leq 0$, where $\epsilon$ is the tolerance factor and $\epsilon = 0.0001$ in literature [33]. This technique is simple, flexible, and easy to merge with optimization algorithms. Therefore it has been very popular in constrained optimization using EAs and it was used successfully with DE [34-37]. Its drawback is that it loses the population diversity, but this can be handled by using EA that is able to maintain diversity [38].

***Experimental setting***: the stopping criteria $max\_FEs = 20000 \times N$, where $N$ is the dimension of the problem [33]. $NP = 50$, subproblem size $V = N/2$, Each subproblem is optimized for 30 iterations "$iter$". The DE parameters are: scaling factor $F$ where $0.4 \leq F \leq 0.5$, $Cr = 0.95$. In VIIC, the number of combinations created for each objective function or constraint in the problem is $maxgrp_{iter} = m \times 10^4$, the number of subproblems $m = N/V$. DEVIIC and DERGC were implemented for 25 runs at all the three problems and the results are presented in Table 7. The experiments were conducted using MATLAB 2012 on an Intel(R) Core(TM) i5 CPU, 3.20 GHz, and running Microsoft Windows 7.

It is shown in Table 7 that DEVIIC achieved better "*best*" and "*mean*" value than DERGC for 5-Units problem. Although the "*std*" of DERGC was better than DEVIIC, this does not mean that DERGC is more stable than DEVIIC because DERGC did not find a single feasible solution out of the 25 solutions ("*FR*"=0). However, DEVIIC achieved "*FR*" of 100%. The same pattern of performance occurred for the 10-Units problem, which has 240 dimensions. The "*best*" and "*mean*" values achieved by DEVIIC for 10-Units problem were better than that of DERGC even if it was larger, because it was from feasible solution. DERGC did not find any feasible solution in the experiment at 10-Units problem. When the dimension of the problem is increased to 720 at 30-Units problem the feasibility ratio of DEVIIC decreased than in the previous two problems. However, DVIIC outperformed DERGC at "*best*" value, even if it is larger than DERGC but it represents a feasible solution. The star ($^*$) in Table 7 represent values which are achieved by infeasible solutions.

***Parameter Analysis***: Table 7 views the results of the three problems when the subproblem size $V$ was decreased to $N/4$ for 5-Units and 10-Units problems, and to $N/8$ for 30-Units problem, and $NP$ was increased to 100 for all the problems. It can be seen that this change affected the performance of DEVIIC positively and did not affect DERGC at all. The "*best*" and "*mean*" fitness values of 5-Units, and 10-Units problems were better than those achieved by large subproblem and small population. "*FR*" was still 100% for DEVIIC and 0% for DERG. At 30-Units problem, these parameter settings increased "*FR*" to 92% rather than 12%, when the previous setting of subproblem size and population were used.

In summary, DEVIIC achieved good results at all the three problems using different setting. These results showed that even if DEVIIC requires additional computations to use variable interaction identification technique, and DERG does not, these computations are justified and worth spending computational resources and time doing them. More analysis is conducted in the following section to show if the difference in time between the two used models is significant or not. DEVIIC and DERGC are ranked based on these results in the following section.

Table 7: Results of 5-Units, 10-Units, and 30-Units

| | **NP=50, V=60** | | **NP=100, V=30** | |
|---|---|---|---|---|
| 5-Units | DEVIIC | DERGC | DEVIIC | DERGC |
| *FR* | **100%** | 0% | **100%** | 0% |
| *time* | 5.5995E+03 | 5.5283E+03 | 5.6211E+03 | 5.6518E+03 |
| *best* | **5.0648E+04** | 5.1483E+04 | **5.0649E+04** | 5.1339E+04 |
| *mean* | **5.2206E+04** | 5.2245E+04 | **5.2162E+04** | 5.2178E+04 |
| *median* | 5.2276E+04 | 5.2200E+04 | 5.2225E+04 | 5.2294E+04 |
| *std* | 5.8835E+02 | 4.2418E+02 | 6.9805E+02 | 4.1397E+02 |
| *worst* | 5.3274E+04 | 5.3216E+04 | 5.3433E+04 | 5.2862E+04 |
| | **NP=50, V=120** | | **NP=100, V=60** | |
| 10-Units | DEVIIC | DERGC | DEVIIC | DERGC |
| *FR* | **100%** | 0% | **100%** | 0% |
| *time* | 5.1640E+03 | 5.0881E+03 | 4.9151E+03 | 4.9381E+03 |
| *best* | 1.0712E+06 | **1.0705E+06**$^*$ | **1.0648E+06** | 1.0713E+06 |
| *mean* | **1.0740E+06** | 1.0744E+06 | **1.0680E+06** | 1.0743E+06 |
| *median* | 1.0740E+06 | 1.0743E+06 | 1.0678E+06 | 1.0745E+06 |
| *std* | 1.2085E+03 | 1.8707E+03 | 1.9504E+03 | 1.7068E+03 |
| *worst* | 1.0760E+06 | 1.0775E+06 | 1.0719E+06 | 1.0776E+06 |
| | **NP=50, V=360** | | **NP=100, V=90** | |
| 30-Units | DEVIIC | DERGC | DEVIIC | DERGC |
| *FR* | **12%** | 0% | **96%** | 0% |
| *time* | 2.1800E+04 | 2.2253E+04 | 1.7905E+04 | 1.8016E+04 |
| *best* | 3.2177E+06 | **3.2026E+06**$^*$ | 3.2084E+06 | **3.1281E+06**$^*$ |
| *mean* | 3.2244E+06 | **3.2155E+06**$^*$ | 3.2139E+06 | **3.2029E+06**$^*$ |
| *median* | 3.2237E+06 | 3.2162E+06 | 3.2142E+06 | 3.2110E+06 |
| *std* | 4.2244E+03 | 6.6696E+03 | 2.9203E+03 | 2.1435E+04 |
| *worst* | 3.2372E+06 | 3.2271E+06 | 3.2200E+06 | 3.2222E+06 |

## VI. ALGORITHM RANKING

To clearly rank the optimization models based on their performance including the feasibility ratio they achieve, the scoring technique that was proposed in [39] is used. The results which use $NP = 100$ and $N/4$ or $N/8$ as the subproblem size is used in this section. This scoring technique calculates two score, one based on the best values ($S_{zy}^{best}$), and one based on the average values ($S_{zy}^{average}$), as follows:

$$S_{zy}^{best} = \left\{(1 - \frac{|F_{zy} - BF_y|}{a \times |BF_y - WF_y|})^{\zeta}\right\} \quad (18)$$

$$S_{zy}^{average} = \left\{(1 - \frac{|F_{zy} - BF_y|}{a \times |BF_y - WF_y|})^{\zeta}\right\} \quad (19)$$

where $S_{zy}^{best}$ is the score of a model z based on "*best*" values and $S_{zy}^{average}$ is the score of a model $z$ based on "*mean*" values in Table 7. $a$ is used to differentiate between the worst feasible and any infeasible solution, and ζ is used to emphasis good solutions. These two values in addition to the average feasibility ratio of model $z$ at all the problems, $FR_z$, are used to find the final score for a model $z$, $FS_z$.

$$FS_z = \frac{\left(\sum_{y=1}^{Y} S_{zy}^{best} + \sum_{y=1}^{Y} S_{zy}^{average}\right)}{\mathrm{z}} \times FR_z \quad (20)$$

In $S_{zy}^{best}$, the actual "*best*" fitness values of a problem $y$ using the optimization model $z$ is ($F_{zy}$),and the best of the two models (smallest) is $BF_y = min\,(F_{zy})$. The worst of them (largest) is $WF_y = max\,(F_{zy})$. In $S_{zy}^{average}$, the actual best "*mean*" fitness value, the smallest mean fitness value in this optimization problem, and the worst "*mean*", largest, fitness value of the two models at a problem y, are used. $a = 0.001$, and $\zeta = 2$. The scoring of DEVIIC and DERGC for the three problems are shown in Table 8.

It is seen that DEVIIC is better than DERGC at two problems based on the "*best*" and "*mean*" values. Also, the average feasibility ratio of DEVIIC is higher than that achieved by DERGC as $FR_{DEVIIC} > FR_{DERGC}$. Moreover, it has better final rank scores than DERGC where $FS_{DEVIIC} > FS_{DERGC}$. This scoring technique showed that the decomposition-based EA which use variables interaction identification technique is successful. Certainly, these models require additional computational resources, but this might be justified by the good performance and high feasibility ratio which is highly desired for the constrained problems. The additional resources can be measured and analyzed as discussed in the following section using algorithm complexity measures.

Table 8: DEVIIC and DERGC ranking

| | DED | |
|---|---|---|
| | DEVIIC | DERGC |
| $\sum_{y=1}^{Y} S_{zy}^{best}$ | 2 | 1 |
| $\sum_{y=1}^{Y} S_{zy}^{average}$ | 2 | 1 |
| $FR_z$ | 98.7% | 0% |
| $FS$ | 1.98 | 0 |

## VII. ALGORITHM COMPLEXITY

To measure the time required by each of the tested models when solving DED problems, the technique that was used in [40] is implemented on the 10-Units problem. This technique uses the time in seconds as an indication of the algorithm complexity. The three parameters for this algorithm complexity measurement technique are $T_0$, $T_1$, and $T_2$. $T_0$ is the computing time in seconds for the basic mathematical operations $x = x + x, x = x/2,\ x = x * x, x = sqrt(x), x = ln(x), x = exp(x), y = x/x$, using $x = 5.55$, for 1000000 times. $T_1$ is the computing time in seconds for the optimization problem of dimension $N$ for 200000 $FEs$. $T_2$ is the mean of 5 runs for the computing time in seconds for the complete algorithm using 200000 $FEs$ at the same problem of dimension $N$. The calculations of $T_2$ are based on 5 runs.

The measurements are applied on the two algorithms DEVIIC and DERGC on the 10-Units problem and the results are presented in Table 9. $T_0$ equals 1.5288. The results of this algorithm complexity measurement technique are presented in Table 9. These results showed that the difference in complexity is seconds. This justifies that the additional time used by DEVIIC which achieved notable difference in performance but the time it used is insignificant. The following section shows the convergence of DEVIIC and DERGC for 10-Units problem during the evolution process.

Table 9: Algorithm complexity for 10-Units

| | | $T_1$ | $T_2$ | $(T_2 - T_1)/T_0$ |
|---|---|---|---|---|
| 10-Unit | DEVIIC | 191.5380 | **219.2500** | **18.1266** |
| | DERGC | 191.5380 | 219.9396 | 18.5777 |

## VIII. CONVERGENCE GRAPHS

The graphs of the best solution for each problem of the 5, 10-Units problems using $V = N/4$, and 30-Units problem using $V = N/8$, and $NP = 100$, are discussed in this section. The graphs showed the performance of DEVIIC against DERGC. The subfigures represent magnified convergence curve for the two models after using one third, two thirds, and all the available $FEs$. The first subfigure of the 5-Units problem in Fig. 1 showed that both DEVIIC and DERGC did not achieve a feasible solution after using 4E+5 $FEs$. When the used $FEs$ are in the range of 4E+5 and 8E+5, the convergence curve of DEVIIC was decreasing, which means that it found a feasible solution but DERGC did not. DEVIIC continued converging towards better solutions as shown in sub-graphs 2 and 3 until the end of the evolution process. However, there was hardly any improvement in DERGC specially during the last evolution stage after using two thirds of the available $FEs$.

At 10-Units problem convergence graph (Fig. 2), DEVIIC outperformed DERGC and converged better than it during the evolution process. DERGC converged slowly and it is clear from second subfigure that it did not manage to find a feasible solution even after using two thirds of the available $FEs$. This showed that even when DERGC does not use any $FEs$ for variables interaction identification, these saved $FEs$ were not efficient in the optimization. This

indicates that spending more $FEs$ in the exploration of better variables that should be optimized in common subproblems leads to better convergence and better fitness values.

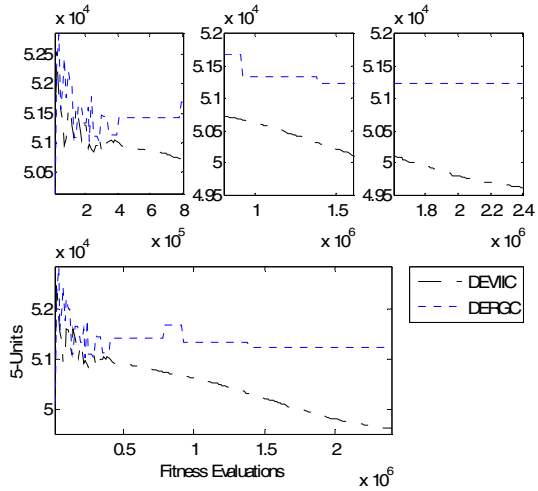


Fig. 1: Convergence graph of 5-Units problem

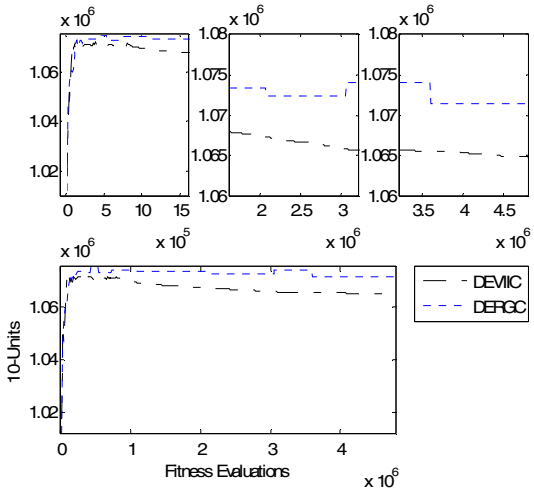


Fig. 2: Convergence graph of 10-Units problem

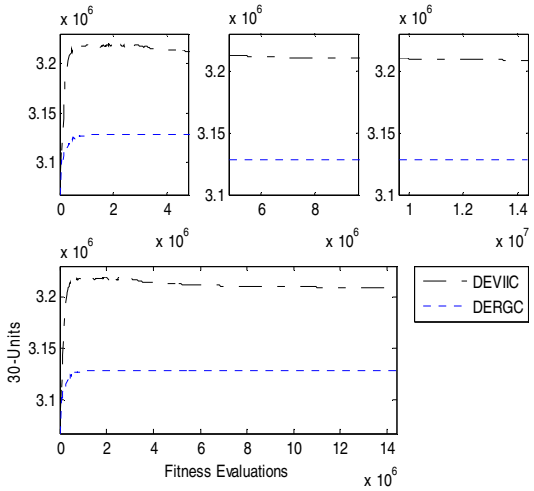


Fig. 3: Convergence graph of 30-Units problem

## IX. Conclusion and Future Work

In conclusion, this paper presents the implementation of decomposition-based EA on a real-world constrained large scale optimization problem. The real-world problem that is tested in this paper is DED which has not been optimized in the literature before using decomposition-based EA. This problem is challenging, has complex structure, and has large number of constraints. Three different dimensions of this problem were tested, 5-Units, 10-Units, and 30-Units, using DEVIIC and DERGC. DEVIIC has been developed recently and it uses a novel variables interaction identification technique (VIIC). DERGC uses one of early developed the decomposition techniques in the literature.

DEVIIC showed a remarkable performance at all the tested real-world large scale problems by achieving competitive results and higher feasibility ratios. Getting feasible solution for the constrained optimization problems is highly desired especially when the problems have high dimensionality. The analysis of the results and the algorithm complexity measures showed that the effort used to identify which variables should be optimized together is justified. Moreover, the difference in complexity between DEVIIC and DERGC showed that the time consumed to implement VIIC is not significant. Therefore, a decomposition-based EA which use a technique for grouping the variables into subproblems is highly recommended for real-world large scale constrained optimization problems. VIIC can be used with different EA to achieve better results for large problems like 30-Units problem. Also, different constraint handling techniques can be used.

## References


[1] Z. Michalewicz, and D. B. Fogel, *How to solve it: Modern Heuristics*: Springer New York, 2000.

[2] M. A. Potter, and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature—PPSN III*, pp. 249-257: Springer, 1994.

[3] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution." pp. 1101-1108.

[4] M. A. Potter, and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary computation,* vol. 8, no. 1, pp. 1-29, 2000.

[5] X. Li, and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms." pp. 1546-1553.

[6] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences,* vol. 178, no. 15, pp. 2985-2999, 2008.

[7] G. B. Dantzig, and P. Wolfe, "Decomposition Principle for Linear Programs," *OPERATIONS RESEARCH,* vol. 8, no. 1, pp. 101-111, January 1, 1960, 1960.

[8] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik,* vol. 4, no. 1, pp. 238-252, 1962.

[9] J. Ryoo, and P. Hajela, "Decomposition-based design optimization method using genetic co-evolution," *Engineering Optimization,* vol. 36, no. 3, pp. 361 - 378, 2004.

[10] W. Chen, and K. Tang, "Impact of problem decomposition on Cooperative Coevolution." pp. 733-740.

[11] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping." pp. 1-8.

[12] T. Ray, and X. Yao, "A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning." pp. 983-989.

[13] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization." pp. 1-8.

[14] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," *Parallel Problem Solving from Nature, PPSN XI*, pp. 300-309: Springer, 2010.

[15] E. Sayed, D. Essam, and R. Sarker, "Using Hybrid Dependency Identification with a Memetic Algorithm for Large Scale Optimization Problems," *Simulated Evolution and Learning*, Lecture Notes in Computer Science L. Bui, Y. Ong, N. Hoai, H. Ishibuchi and P. Suganthan, eds., pp. 168-177: Springer Berlin Heidelberg, 2012.

[16] E. Sayed, D. Essam, and R. A. Sarker, "Dependency Identification technique for large scale optimization problems." pp. 1-8.

[17] L. Sun, S. Yoshida, X. Cheng, and Y. Liang, "A cooperative particle swarm optimizer with statistical variable interdependence learning," *Information Sciences,* vol. 186, no. 1, pp. 20-39, 2012.

[18] X. Han, H. Gooi, and D. S. Kirschen, "Dynamic economic dispatch: feasible and optimal solutions," *Power Systems, IEEE Transactions on,* vol. 16, no. 1, pp. 22-28, 2001.

[19] D. He, F. Wang, and Z. Mao, "A hybrid genetic algorithm approach based on differential evolution for economic dispatch with valve-point effect," *International Journal of Electrical Power & Energy Systems,* vol. 30, no. 1, pp. 31-38, 2008.

[20] D. C. Walters, and G. B. Sheble, "Genetic algorithm solution of economic dispatch with valve point loading," *Power Systems, IEEE Transactions on,* vol. 8, no. 3, pp. 1325-1332, 1993.

[21] P. Attaviriyanupap, H. Kita, E. Tanaka, and J. Hasegawa, "A hybrid EP and SQP for dynamic economic dispatch with nonsmooth fuel cost function," *Power Systems, IEEE Transactions on,* vol. 17, no. 2, pp. 411-416, 2002.

[22] T. Jayabarathi, K. Jayaprakash, D. Jeyakumar, and T. Raghunathan, "Evolutionary programming techniques for different kinds of economic dispatch problems," *Electric Power Systems Research,* vol. 73, no. 2, pp. 169-176, 2005.

[23] M. A. K. Azad, "A modified differential evolution based solution technique for economic dispatch problems," 2009.

[24] R. Balamurugan, and S. Subramanian, "Differential evolution-based dynamic economic dispatch of generating units with valve-point effects," *Electric Power Components and Systems,* vol. 36, no. 8, pp. 828-843, 2008.

[25] R. Balamurugan, and S. Subramanian, "An improved differential evolution based dynamic economic dispatch with nonsmooth fuel cost function," *Journal of Electrical Systems,* vol. 3, no. 3, pp. 151-161, 2007.

[26] Y. Lu, J. Zhou, H. Qin, Y. Wang, and Y. Zhang, "Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects," *Engineering Applications of Artificial Intelligence,* vol. 24, no. 2, pp. 378-387, 2011.

[27] S. Das, and P. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," *Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India*, 2010.

[28] Z.-L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *Power Systems, IEEE Transactions on,* vol. 18, no. 3, pp. 1187-1195, 2003.

[29] C. Panigrahi, P. Chattopadhyay, R. Chakrabarti, and M. Basu, "Simulated annealing technique for dynamic economic dispatch," *Electric power components and systems,* vol. 34, no. 5, pp. 577-586, 2006.

[30] V. Ravikumar Pandi, and B. K. Panigrahi, "Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm," *Expert Systems with Applications,* vol. 38, no. 7, pp. 8509-8514, 2011.

[31] T. Victoire, and A. E. Jeyakumar, "Deterministically guided PSO for dynamic dispatch considering valve-point effect," *Electric power systems research,* vol. 73, no. 3, pp. 313-322, 2005.

[32] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering,* vol. 186, no. 2, pp. 311-338, 2000.

[33] R. Mallipeddi, and P. N. Suganthan, *Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization*, 2010.

[34] J. Tvrdík, and R. Polakova, "Competitive differential evolution for constrained problems." pp. 1-8.

[35] N. M. Hamza, S. M. Elsayed, D. L. Essam, and R. A. Sarker, "Differential evolution combined with constraint consensus for constrained optimization." pp. 865-872.

[36] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing,* vol. 10, no. 2, pp. 629-640, 2010.

[37] M. Ali, and Z. Kajee-Bagdadi, "A local exploration-based differential evolution algorithm for constrained global optimization," *Applied Mathematics and Computation,* vol. 208, no. 1, pp. 31-48, 2009.

[38] E. Mezura-Montes, and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation,* vol. 1, no. 4, pp. 173-194, 2011.

[39] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "On an evolutionary approach for constrained optimization problem solving," *Applied Soft Computing,* vol. 12, no. 10, pp. 3208-3227, 2012.

[40] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *KanGAL Report,* vol. 2005005, 2005.