A New Approach for Solving the Unit Commitment Problem by Adaptive Particle Swarm Optimization

V.S. Pappala, Student Member, IEEE, and I. Erlich, Senior Member, IEEE

Abstract—This paper presents a new approach for formulating the unit commitment problem which results in a considerable reduction in the number of decision variables. The scheduling variables are coded as integers representing the operation periods of a generating unit. The unit commitment problem is solved using a new parameter free adaptive particle swarm optimization (APSO) approach. This algorithm provides solutions to the major demerits of PSO such as parameter tuning, selection of optimal swarm size and problem dependent penalty functions. The constrained optimization problem is solved using adaptive penalty function approach. The penalty terms adapt to the performance of the swarm. So no additional penalty coefficient tuning is required. This paper describes the proposed algorithm with the new variable formulation and presents test results on a ten unit test system. The results demonstrate the robustness of the new algorithm in solving the unit commitment problem.

Index Terms—Binary and Integer programming, Particle swarm optimization, Penalty function, Unit commitment.

I. INTRODUCTION

UNIT commitment (UC) is a nonlinear mixed integer optimization problem to schedule the operation of the generating units at minimum operating cost while satisfying the demand and reserve requirements. The UC problem has to determine the on/off state of the generating units at each hour of the planning period and optimally dispatch the load and reserve among the committed units. UC is the most significant optimization task in the operation of the power systems. Solving the UC problem for large power systems is computationally expensive. The complexity of the UC problems grows exponentially to the number of generating units.

Several solution strategies have been proposed to provide quality solutions to the UC problem and increase the potential savings of the power system operator. These include deterministic and stochastic search approaches. Deterministic approaches include the priority list method [1], dynamic programming [2], Lagrangian Relaxation [3] and the branch-and-bound methods [4]. Although these methods are simple and fast, they suffer from numerical convergence and solution quality problems. The stochastic search algorithms such as particle swarm optimization [5]-[8], genetic algorithms [9], evolutionary programming [10], simulated annealing [11], ant colony optimization [12] and tabu search [13] are able to overcome the shortcomings of traditional optimization techniques. These methods can handle complex nonlinear constraints and provide high quality solutions. However, all these algorithms suffer from the curse of dimensionality. The increased problem size adversely effects the computational time and the quality of the solutions.

This paper addresses a new approach to handle the unit commitment variables. The scheduling of a unit is expressed as operation periods or duty cycles. Hence the unit commitment variables are coded as integers. This formulation drastically reduces the number of decision variables and hence can overcome the shortcomings of stochastic search algorithms for UC problems. Due to simplicity and less parameter tuning, adaptive particle swarm optimization is used for solving the problem.

II. PROBLEM FORMULATION

The objective of the UC problem is to minimize the total operating costs subjected to a set of system and unit constraints over the scheduling horizon.

It is assumed that the production cost, \( PC_i \) for unit \( i \) at any given time interval is a quadratic function of the generator power output, \( p_i \).

\[
PC_i = a_i + b_i p_i + c_i p_i^2
\]  

(1)

Where \( a_i, b_i, c_i \) are the unit cost coefficients. The generator start-up cost depends on the time the unit has been switched off prior to the start up, \( T_{off} \). The start-up cost \( SC_i \) at any given time is assumed to be an exponential cost curve.

\[
SC_i = \sigma_i + \delta_i \left( 1 - \exp \left( \frac{-T_{off} - T}{\tau_i} \right) \right)
\]  

(2)

Where \( \sigma_i \) is the hot start-up cost, \( \delta_i \) the cold start-up cost and \( \tau_i \) is the cooling time constant.

The total operating costs, \( OC_T \) for the scheduling period \( T \) is the sum of the production costs and the start-up costs.

\[
OC_T = \sum_{i=1}^{T} \sum_{t=1}^{N} PC_{i,t} U_{i,t} + SC_{i,t} (1 - U_{i,t-1}) U_{i,t}
\]  

(3)

Where \( U_{i,t} \) is the binary variable to indicate the on/off state of the unit i at time t. \( U_{i,t} = 1 \) if unit i is committed at time t, otherwise \( U_{i,t} = 0 \).

The overall objective is to minimize \( OC_T \) subject to a number of system and unit constraints. All the generators are

This Project is supported by the German Federal Ministry of Education and Research (BMBF) under the grant 03SF0312A.

Pappala Venkata Swaroop (venkata.pappala@uni-due.de) is with the institute of electrical power systems, Universität Duisburg-Essen.

Istvan Erlich (istvan.erlich@uni-due.de) is with the institute of electrical power systems, Universität Duisburg-Essen, 47057- Duisburg, Germany.

©2008 IEEE.
assumed to be connected to the same bus supplying the total system demand. Therefore, the network constraints are not taken into account.

- Power balance constraint
  The total generated power at each hour must be equal to the load of the corresponding hour, \( D_i \).
  \[
  \sum_{i=1}^{N} p_{i,j} U_{i,j} = D_i
  \]  
  (4)

- Spinning reserve constraint
  For reliable operation, the power system has to maintain a certain megawatt capacity as spinning reserve, \( R_i \).
  \[
  \sum_{i=1}^{N} p_{i,j}^{\text{max}} U_{i,j} \geq D_i + R_i
  \]  
  (5)

- Generation limit constraint
  \[
  p_{i,j}^{\text{min}} \leq p_{i,j} \leq p_{i,j}^{\text{max}}
  \]  
  (6)

- Minimum up/down time constraints
  The minimum up/down time constraints indicate that a unit must be on/off for a certain number of hours before it can be shut off or brought online, respectively.
  \[
  T_{i,j}^{\text{on}} \geq \text{MUT}_i
  \]  
  (7)

\[
T_{i,j}^{\text{off}} \geq \text{MDT}_i
\]  
(8)

- The initial unit states at the start of the scheduling period must be taken into account.

Where \( p_{i,j}^{\text{on}} \) and \( p_{i,j}^{\text{off}} \) are the minimum and maximum generation limit of the \( i \)th unit, \( T_{i,j}^{\text{on}} \) & \( T_{i,j}^{\text{off}} \) represents the duration during which the \( i \)th unit is continuously on and off respectively and \( \text{MUT}_i \) & \( \text{MDT}_i \) are the minimum up-time and down-time respectively.

III. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) was first proposed by Kennedy and Eberhart [14] in 1995. PSO is a population based searching algorithm. This approach simulates the simplified social system such as fish schooling and birds flocking. PSO is initialized by a population of potential solutions called particles. Each particle flies in the search space with a certain velocity. The particle’s flight is influenced by cognitive and social information attained during its exploration. It has very few tunable parameters and the evolutionary process is very simple. It has been successfully applied to solve nonlinear, combinatorial, multimodal and multi-objective problems. It is capable of providing quality solutions to many complex power system problems.

In this study adaptive particle swarm optimization (APSO) [15], [16] is used for solving the UC problem. APSO is a parameter free technique. The algorithm is able to adjust the flight of the particles based on their performances. No special parameter tuning is required. It is also capable of finding the most appropriate swarm size. In this algorithm, different sized groups of particles called Tribes move in the search space to find the optimal solution. All the particles in a group share their flying experiences in order to find a local minimum. The tribes explore several promising areas simultaneously and associate among each other to decide on the global minimum.

IV. PARTICLE FORMULATION

All algorithms solve the UC problem by binary programming. So the particle is coded as a binary string representing the on/off status of the generators for each hour of the scheduling period \( T \). But in the new approach presented here, the binary UC variables are formulated as integers as shown below.

\[
\text{Fig. 1. The new UC variable formulation}
\]

The unit commitment schedule in fig. 1 is represented by five integers. Each integer represents the continuous on/off period of a generating unit. Negative integer indicates the period when the unit is switched off and positive integer indicates the period when the unit is in operation. The number of integers required to represent the scheduling decisions should be defined before the start of the optimization. For a system with \( N \) generating units and \( P \) integers representing the UC schedule of each unit, the particle consists of \( N*P \) integer variables representing the power generation levels of units at each hour and \( N*P \) integer variables representing UC schedule of the units at each hour. This formulation would reduce the number of UC decision variables compared to the binary coding by \( (T-P)/T\% \).

V. APSO APPROACH TO UC

The major drawbacks of evolutionary algorithms are parameter tuning, optimal population/swarm size, premature convergence and problem dependent penalty coefficients. These demerits can be successfully overcome by using the adaptive particle swarm optimization algorithm. APSO can be used as a black box. It is independent of the problem being solved. The integers representing operating schedule of a unit \( i \) is depicted as \( I_{i,p} \), \( p = 1,2,....,P \). The optimization process can be explained by the following steps.

Step 1: Swarm initialization

The evolutionary process starts with a single particle representing a single tribe. The particle is randomly initialized within the specified limits. The UC integer variables are generated as shown below:

\[
I_{i,p} = \begin{cases} 
\text{rand}_i \left( -\text{int} \left( \frac{T}{P-1} \right) , \text{int} \left( \frac{T}{P-1} \right) \right) , & \text{p} = 1,\ldots,P-1 \\
T - \sum_{p=1}^{P-1} I_{i,p} , & \text{p} = P 
\end{cases}
\]  
(9)

A fitness value is calculated using (3) and is assigned to the
particle. The constraints (4)-(8) are verified and a penalty term is evaluated for each violation. This penalty is then added to the particles fitness value. The particle memorizes its previous two performances. Initially these variables are initialized to the particle’s current position.

Step 2: Initialize information groups.
Each particle has a group of informers to assist in its search process. The size of this group is adaptive. Initially this group consists of the particle itself and the best particle in the tribe.

Step 3: Update the current position of the particles.
The performance of the particles can be an improvement (+), status quo (=) or a deterioration (-). The particles are graded based on their previous two performances. A particle is considered excellent if its previous two performances are improvements (++) , good if it has just improved its performance (++), neutral if it has the following improvements (++), good if it has just improved its performance (=+), neutral if its previous performance (=+), bad if its previous performance is deteriorations (--) . If the particle happens to be bad or neutral, then the pivot method is used as update strategy whereas if the particles are excellent or good, then Gaussian update strategy is used. These strategies are explained below.

A. Pivot strategy
Identify the individual best performance of the particle, $p$ and best informer of the particle, $g$. Define two hyperspheres, $H_p$ and $H_g$ with $p$ and $g$ as centers and radius equal to the distance between them. A point is randomly chosen in each of these hyperspheres. A weighted combination of these points gives the new position vector, $x_{id}^{k+1}$ of the particle.

$$ x_{id}^{k+1} = c_1 \text{random}(H_p) + c_2 \text{random}(H_g) $$

$$ c_1 = \frac{f(p)}{f(p) + f(g)} $$

$$ c_2 = \frac{f(g)}{f(p) + f(g)} $$

Where $c_1$ and $c_2$ are the weight factors. These values are calculated using the objective function (4), $f$ corresponding to $p$ and $g$.

B. Gaussian Update strategy
This update strategy is very similar to standard PSO update equations. The velocity of the particle $i$ at iteration $k+1$ depend on its previous velocity ($v_{id}^k$), its previous best performance ($p_{id}$) and performance of the best informer ($p_{gd}$) as shown below.

$$ \delta_p = p_{id} - x_{id}^k $$

$$ \delta_g = p_{gd} - x_{id}^k $$

$$ v_{id}^{k+1} = v_{id}^k + \text{gauss_rand}(\delta_p, \sigma_p)/2 + \text{gauss_rand}(\delta_g, \sigma_g)/2 $$

$$ x_{id}^{k+1} = x_{id}^k + \chi(v_{id}^{k+1}) $$

Where $\text{gauss_rand}(\mu, \sigma)$ generates a normal distributed numbers with mean $\mu$ and standard deviation $\sigma$. The Gaussian distribution will provide both global and local exploration around $p$ and $g$. This approach eliminates the use of acceleration coefficients. Hence the update strategy is totally independent of the tuning parameters. The velocity of integer UC variables is bounded to $v_{min}$ and $v_{max}$. The update procedure for the integer UC variables can be explained by the fig.2.

To each integer or duty cycle a certain velocity is added. The arrows indicate the direction of velocity. The left arrow implicates a negative velocity which causes a reduction in the corresponding integer or operation period. Where as a right arrow indicates a positive velocity which helps to increase the duty cycle or the corresponding integer. There is no need to assign a sigmoid transition probability or mutation probability for altering the on/off status of the units. The UC integer variables are treated like any other decision variables. After updating the integer variables they are ceiled to the nearest integers.

A fitness value is calculated using (3) and is assigned to the particle. The constraints (4)-(8) are verified and a penalty term is evaluated for each violation. This penalty is then added to the particles fitness value. The update procedure for the particles of all tribes constitute one cycle. The tribes are allowed to explore and evolve for a certain number of cycles.

Step 4: check the bounds on all variables
The Generation limit constraint is checked for all continuous variables. If the variables cross the bounds, they are made to stay on the boundary. The sum of the integer UC variables of each unit must be equal to the scheduling period $T$.

$$ \sum_{p=1}^{P} |I|^p_i = T \quad \forall i \in \{1, ..., N\} $$

This constraint is implemented by using the following fuzzy rules.

$$ \text{if} \left( \sum_{p=1}^{P} |I|^p_i < T \right) \text{ Rule1: } I_i^p = \pm \left( T - \sum_{p=1}^{P} |I|^p_i \right) $$

$$ \text{if} \left( |I|^p_i + |I|^p_j > T \right) \text{ Rule2: } I_i^p = \pm T $$

$$ \text{if} \left( \sum_{p=1}^{P} |I|^p_i > T \right) \text{ Rule7: } I_i^p = \pm \left( T - \sum_{p=1}^{P} |I|^p_i \right) $$

Step 5: Check for the termination criterion
Check if the current iteration number has reached the termination limit. If yes, exit otherwise check if the number of
cycles is less than the total size of current information groups. This condition will ensure that all the tribes have enough time to explore, evolve and share information with their informers. If the second condition is obeyed go to step 3 or else proceed to step 6.

Step 6: Adaptations
Evaluate the performance of the tribes. A tribe is considered good/bad if it has more number of good/bad particles. Since the bad tribe do not have the capability to converge, it needs more information or particles to find an optimal solution. The best particle of the bad tribe will generate a new particle for the new tribe. Each bad tribe contributes a particle each to the new tribe. The new particles will be in contact to their parents by adding them to their respective information groups. On the other hand, the good tribe is capable of converging to an optimal solution. It may not need all its particles for further exploration. Therefore the worst particle is eventually removed from the tribe. The information groups of all the particles are updated by replacing this deleted particle with the best particle of that tribe. This will ensure that the search process do not spend time on unwanted particles. Hence the algorithm can find the optimum with minimum function evaluations. After the adaptations on the tribes, continue to step 3.

VI. CONSTRAINTS HANDLING
Constrained optimization problems are usually solved using penalty function approach. The infeasible particles will be penalized for the constraint violation by adding a penalty term to the fitness value. Many static and dynamic penalty functions [17] have been successfully tested on a number of problems. But all these penalty functions require the best penalty coefficients which can be determined only through repeated trials. Hence a self adaptive penalty function approach [18] is used. Two penalty terms, \( d(x) \) and \( p(x) \) are added to each particle. These terms depend on the performance of the whole swarm. The new fitness value is the sum of the distance value, \( d(x) \) and the penalty value, \( p(x) \). The distance value is defined as follows:

\[
d(x) = \begin{cases} 
\frac{b'(x)}{\sqrt{OCT^2 + b'(x)^2}} & \text{if } r_f = 0 \\
\text{otherwise} 
\end{cases}
\]

Where

\[
r_f = \frac{\text{number of feasible particle}}{\text{swarm size}}
\]

\( OCT \) is the normalized operation cost and \( b'(x) \) is the sum of the normalized violation of each constraint divided by the total number of constraints. For a swarm with no feasible particles, the first and foremost objective of the optimization is to find a feasible particle i.e. reduce \( b'(x) \). When there are feasible particles in the swarm then the algorithm tries to reduce the root mean square sum of the \( OCT \) and \( b'(x) \). The information carried by the infeasible particle can be very helpful in finding the optimal solution. For instance, when there are few feasible particles, infeasible particles with low constraint violation can help in finding more feasible particles. On the other hand when there are more feasible particles, infeasible particles with low objective value can give more information about the global optimum. So at every time stage, the search process requires an appropriate set of infeasible particles to find the optimal solution. This phenomenon is implemented by the penalty value, \( p(x) \).

In binary programming, special solution techniques have to be applied to satisfy the minimum up-time and down-time constraints. The particles which violate constraints (7) and (8) should be corrected by using a repair or mutation operator [19]. Some algorithms provide arbitrary feasible particles at the start of the search process. But the APSO approach with the new UC variable formulation can handle the MUT, MDT constraints without any correction operators.

VII. NUMERICAL EXAMPLE
The APSO algorithm was tested on a ten generator system considering a time horizon of 24 hours. The date is given in Table I and II. The spinning reserve is assumed to be 5% of the load demand. The APSO program was implemented in visual c language. Test results are compared against the binary programming results in [6]. In binary PSO the particle consists of 10*24 continuous variables representing the generation levels of the ten units and 10*4 integer variables representing the operation schedule of the units at each hour. Whereas, the new approach, the particle consists of 10*24 continuous variables and 10*5 integer variables. Hence there is approximately 80% reduction in the number of UC decision variables compared to the binary programming approach.

<table>
<thead>
<tr>
<th>Unit</th>
<th>( p_{min} ) (kW)</th>
<th>( p_{max} ) (kW)</th>
<th>Fuel cost</th>
<th>Start-up cost</th>
<th>MUT (hrs)</th>
<th>MDT (hrs)</th>
<th>INS (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>545</td>
<td>150</td>
<td>1000</td>
<td>16.19</td>
<td>0.000484</td>
<td>4500</td>
<td>4500</td>
</tr>
<tr>
<td>2</td>
<td>545</td>
<td>150</td>
<td>970</td>
<td>17.26</td>
<td>0.000311</td>
<td>5000</td>
<td>5000</td>
</tr>
<tr>
<td>3</td>
<td>130</td>
<td>20</td>
<td>700</td>
<td>16.60</td>
<td>0.002000</td>
<td>550</td>
<td>550</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>20</td>
<td>680</td>
<td>16.50</td>
<td>0.002111</td>
<td>560</td>
<td>560</td>
</tr>
<tr>
<td>5</td>
<td>162</td>
<td>25</td>
<td>450</td>
<td>19.70</td>
<td>0.003589</td>
<td>900</td>
<td>900</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>20</td>
<td>370</td>
<td>22.26</td>
<td>0.007126</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>7</td>
<td>85</td>
<td>25</td>
<td>480</td>
<td>27.74</td>
<td>0.000798</td>
<td>260</td>
<td>260</td>
</tr>
<tr>
<td>8</td>
<td>55</td>
<td>10</td>
<td>660</td>
<td>25.92</td>
<td>0.004133</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>55</td>
<td>10</td>
<td>665</td>
<td>27.27</td>
<td>0.002229</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
<td>10</td>
<td>670</td>
<td>27.79</td>
<td>0.001730</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hour 1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand (MW)</td>
<td>700</td>
<td>750</td>
<td>850</td>
<td>950</td>
<td>1000</td>
<td>1000</td>
<td>1100</td>
<td>1150</td>
<td>1200</td>
<td>1300</td>
<td>1400</td>
</tr>
<tr>
<td>Hour</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>Demand (MW)</td>
<td>1400</td>
<td>1300</td>
<td>1200</td>
<td>1050</td>
<td>1000</td>
<td>1100</td>
<td>1200</td>
<td>1400</td>
<td>1300</td>
<td>1100</td>
<td>900</td>
</tr>
</tbody>
</table>

The total operating costs using binary programming are $565,804 according to [6] whereas for the new approach, the costs are $561,586. The UC schedules of the two approaches
are presented in Tables III and IV. The results imply that the new approach exploits the search space far better than the binary PSO approach.

**TABLE III**

<table>
<thead>
<tr>
<th>Unit</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>111111111111111111111111</td>
</tr>
<tr>
<td>2</td>
<td>111111111111111111111111</td>
</tr>
<tr>
<td>3</td>
<td>111111111111111111111111</td>
</tr>
<tr>
<td>4</td>
<td>001111111111111111111111</td>
</tr>
<tr>
<td>5</td>
<td>111111111111111111111111</td>
</tr>
<tr>
<td>6</td>
<td>000000001111111000011000</td>
</tr>
<tr>
<td>7</td>
<td>000000000111100000010000</td>
</tr>
<tr>
<td>8</td>
<td>000000000011000000000000</td>
</tr>
<tr>
<td>9</td>
<td>000000000001000000000000</td>
</tr>
<tr>
<td>10</td>
<td>000000000000000000000000</td>
</tr>
<tr>
<td>OC($)</td>
<td>565,450</td>
</tr>
</tbody>
</table>

**TABLE IV**

<table>
<thead>
<tr>
<th>UNIT SCHEDULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT 1 2 3 4 5</td>
</tr>
<tr>
<td>1 -24 0 0 0 0</td>
</tr>
<tr>
<td>2 -24 0 0 0 0</td>
</tr>
<tr>
<td>3 -3 12 -2 6 -1</td>
</tr>
<tr>
<td>4 -6 15 -3 0 0</td>
</tr>
<tr>
<td>5 -8 8 -3 2 -3</td>
</tr>
<tr>
<td>6 -9 4 -6 1 -4</td>
</tr>
<tr>
<td>7 -10 2 -12 0 0</td>
</tr>
<tr>
<td>8 -11 1 -12 0 0</td>
</tr>
<tr>
<td>9 -24 0 0 0 0</td>
</tr>
<tr>
<td>OC($)</td>
</tr>
</tbody>
</table>

The quality of the solution obtained by the new approach can also be examined using the reserve allocation shown in figure 3. In binary method, there is huge unwanted reserve allocation during the low demand hours. In the new approach there is optimal reserve allocation. The reserve allocation by the new approach is very close to the required reserve capacity. Fig. 4 shows the convergence tendency of APSO with the new UC integer variable formulation.

![Fig. 3. Reserve allocation by binary programming and the new approach](image)

**Fig. 3. Reserve allocation by binary programming and the new approach**

**VIII. CONCLUSION**

The new UC variable formulation causes considerable reduction in the number of decision variables and thus the size of optimization problem. It still requires the definition of operation periods or integers required for the UC variable formulation which can not be determined optimally a priori. However, in the near future, authors are planning to present an extension of the proposed algorithm that allows to overcome this drawback by using adaptive particle size for integer coded UC variables. Also the convergence behavior could be made faster by using special operators that can assist the particles to satisfy the equality demand constraint and to remove the excess reserve allocation.

The proposed adaptive particle swarm optimization makes practical implementations much easier due to the fact that APSO is totally free from parameter tuning. The algorithm is self adaptive and is therefore independent of the problem to be solved. The adaptive penalty function approach introduced reduces the burden of selecting the best penalty coefficients. The particles do not require any repair strategies for satisfying the constraints. As a result, the algorithm is capable of efficiently exploring the search space and generating quality solutions. This research is a contribution to the development of new robust parameter free evolutionary algorithms for solving the unit commitment problem.

**REFERENCES**


Pappala, V.S; Wilch, M.; Singh, S.N.; Erlich, I.; “Reactive power management in offshore wind farms by adaptive PSO”, International conference on intelligent system application to power system 2007. This paper won the best paper award.


BIographies

Venkata Swaroop Pappala received the B.E degree in electrical engineering from Faculty of Electrical engineering, Nagarjuna University, India in 2002, and M.Sc. degree in electrical engineering with emphasis on power and automation from University Duisburg-Essen, Germany in 2005. He is currently a Ph.D. student at the University of Duisburg-Essen, Germany. His research interests include stochastic optimization under uncertainty using evolutionary algorithms.

Istvan Erlich (1953) received his Dipl.-Ing. degree in electrical engineering from the University of Dresden/Germany in 1976. After his studies, he worked in Hungary in the field of electrical distribution networks. From 1979 to 1991, he joined the Department of Electrical Power Systems of the University of Dresden again, where he received his PhD degree in 1983. In the period of 1991 to 1998, he worked with the consulting company EAB in Berlin and the Fraunhofer Institute IITB Dresden respectively. During this time, he also had a teaching assignment at the University of Dresden. Since 1998, he is Professor and head of the Institute of Electrical Power Systems at the University of Duisburg-Essen/Germany. His major scientific interest is focused on power system stability and control, modeling and simulation of power system dynamics including intelligent system applications. He is a member of VDE and senior member of IEEE.