A Variable-Dimension Optimization Approach to Unit Commitment Problem

Venkata Swaroop Pappala and Istvan Erlich, *Senior Member, IEEE*

***Abstract*—This paper proposes a variable-dimension optimization approach to address the high dimensionality issues in solving the unit commitment problem. This method introduces the concept of adaptive search space dimension. The proposed approach is implemented in particle swarm optimization algorithm. The optimization process starts with an arbitrary problem dimension, adapts with respect to the swarm progress and finally selects the optimal dimensional space. The efficiency of this method is tested on a ten-unit test system. The results are compared with binary programming and fixed duty cycle approaches. The simulation results show that the proposed method results in considerable reduction of problem dimension, faster convergence and improved quality of the final solution.**

***Index Terms*—Evolutionary programming, particle swarm optimization, unit commitment.**

I. Introduction

THE objective of the unit commitment problem (UCP) is to determine the optimal scheduling of the generating units along with their generation levels at minimum operating costs while satisfying the system and unit constraints. The decision variables include the binary unit commitment (UC) variables and real-valued economic dispatch (ED) variables. The UC variables describe the ON/OFF status while the ED variables indicate the generation levels of the generators at each hour of the planning period. The dimension of the problem increases rapidly with longer planning time and increased size of the power system. Solving this high dimensional problem is a challenging task.

Numerous deterministic and stochastic optimization techniques such as dynamic programming [1], branch-and bound [2], Lagrange relaxation [3], genetic algorithm [4], particle swarm optimization (PSO) [5], tabu search [6], etc. have been developed to solve the UCP. PSO is used to solve the UCP in this paper. PSO is an evolutionary computational technique proposed by Kennedy and Eberhart in 1995 [7]. The search process is initiated by a population or swarm of particles. A particle represents a solution to a given problem.

The particles traverse the entire search space at a certain velocity. The particle's flight and velocity is guided by its own flying experience and motivated by the social behavior of the particles in its neighborhood. All the algorithms including PSO suffer from the curse of dimensionality, i.e., the numerical convergence and solution quality drastically deteriorates with problem dimension. Many researchers have proposed hybrid algorithms [8]–[11] to address the above issue but no attempt was made to reduce the problem dimension. This paper presents an innovative UCP modeling which helps in considerable reduction in the problem dimension.

The first attempt in this process was done by converting the binary coding of UC schedule to integer representation [12], [13]. The UC schedule is no more represented by the switching status of the units at each hour but described by the continuous operation (duty cycles) of the unit over a period of time. This modeling brings about a huge reduction in the number of UC variables. The number of duty cycles (integers) required to describe the schedule of any unit should be defined before the start of the optimization process. This number is common to all units in the power system. This is not a good procedure because some units might supply the base load so their switching is minimum (few duty cycles), where as some units have small start-up delay time and might switch rapidly to support sudden changes in load (more duty cycles). So having a common number of duty cycles might result in idle duty cycles for base load generation units or high delay time units and insufficient duty cycles for units with low delay time.

This paper proposes a variable-dimension optimization approach to address the above issue. The number of duty cycles need not be defined beforehand and can vary from unit to unit. The number of duty cycle required by each unit is decided during the optimization process. The total number of UC variables is therefore not fixed and varies during the search process. The length or dimension of the particles is adaptive. The dimension of the search space continuously varies until it finds the right dimension. Hence derives the name variable-dimension optimization. Besides describing the duty cycles for the units, the particle also provides information regarding the number of duty cycles required by each unit through a header index. This index is not an optimization decision variable but controlled by the special reserve manager operator. The main advantage of this process is that there are no idle duty cycles for any unit and every dimension of the search space is effectively utilized. The aim of this paper is to present the variable-dimension optimization approach to solve the high dimensional unit commitment problem with reduced search dimension.

II. Swarm Outlook

In variable-dimension optimization approach, the swarm consists of particles which differ also in dimension. The dimension-

Manuscript received July 16, 2009; revised October 18, 2009. First published January 26, 2010; current version published July 21, 2010. This work was supported by the German Federal Ministry for Education and Research (BMBF) under Grant 03SF0312A. Paper no. TPWRS-00554-2009.

The authors are with the Institute of Electrical Power Systems, Universität Duisburg-Essen, 47057-Duisburg, Germany (e-mail: venkata.pappala@uni-due.de; istvan.erlich@uni-due.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TPWRS.2009.2038921

ality of a particle's search space is completely different from its neighbors. However the particles do share information regarding their flying experiences among each other. This information exchange is only between identical dimensions. The particles do not anticipate or borrow information from its alien or unknown space dimension.

III. Problem Definition

The objective of the UC problem is to minimize the total operating costs of all the generators, N, subjected to a set of system and unit constraints over the scheduling period, T. The UC problem can be formulated as a minimization problem as follows:

$$\min\left[\sum_{j=1}^{T}\sum_{i=1}^{N}(\mathrm{FC_i}(p_{ij}))\right]*u_{ij}+\left[\sum_{j=1}^{T}\sum_{i=1}^{N}\mathrm{SC_{ij}}\right]*u_{ij}(1-u_{ij-1}). \tag{1}$$

The fuel costs (FC) are assumed to be a quadratic function of the generator output power, p. The start-up costs are expressed as an exponential function of the down time. The decision variables include the binary UC variables, U, describing the commitment of the units and the continuous or economic dispatch variables, Z, describing the generation level of the units at every hour of the planning period.

The system constraints include:

— power balance constraint;

— spinning reserve constraint.

The unit constraints include:

— initial operating status;

— minimum and maximum generation limits;

— ramp rates;

— minimum up/down time (MUT/MDT) limits.

The constraints are handled using adaptive penalty function approach [12]. The challenging task in UC problem is handling the binary UC variables and their related constraints (MUT/MDT constraints). In evolutionary programming, the binary variables are handled by a probability calculated using the sigmoid function. This probability decides whether a binary variable can be either zero or one. So it is very difficult to control a set of binary variables together in order to satisfy the MUT/MDT constraints. Repair techniques are usually used for this purpose. More over modeling the UC by binary programming results in a high dimension problem. Highly efficient algorithms are required to solve these problems.

IV. Particle Formulation

The binary programming approach for UC problem results in a binary particle representing the on/off status of the generators at each hour of the planning period. So for a scheduling period of one day, the particle coding for one unit consists of 24 binary bits as shown in Fig. 1.

The duty cycle approach presented in [13] and [14] converts the binary programming to integer programming as shown in Fig. 2.

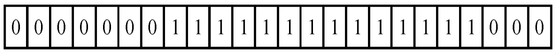

Fig. 1. Particle representation using binary programming.

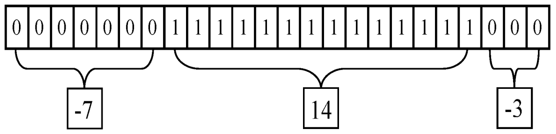

Fig. 2. Particle representation using integer programming.

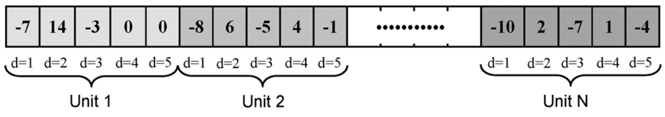

Fig. 3. Particle coding in fixed duty cycle approach for N units.

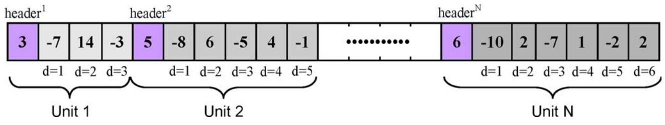

Fig. 4. Particle coding in variable-dimension approach.

The point of time representation in binary programming is converted to period of time representation in integer programming. Instead of representing the status of a unit at each hour, the operation of the unit over a period of time is described in duty cycles. The 24 bits in the binary particle in Fig. 1 can be represented by three duty cycles as shown in Fig. 2. Each integer represents the continuous on/off period of a generator. Positive/Negative sign indicates that the unit is up/down respectively. But in this modeling, the number of duty cycles or integers required by each unit should be defined before the start of the optimization process. This method is referred to as the fixed-duty cycle approach. The particle coding with this approach for N generators is as shown in Fig. 3. In this case the number of duty cycles for each unit is fixed to five. So there are five $\pm$ve integers describing the commitment of the each unit for the entire planning period. This may not be well defined in all situations because the number of duty cycles required by each unit depends not only on the technical and operational delay times of the units but also on the nature of demand profile over the planning period.

The proposed variable-dimension optimization approach has an adaptive particle size. It means the number of duty cycles required by each unit is not predefined but decided during the optimization process. For the UCP with N generators, the particle is as shown in Fig. 4.

The particle consists of a string of positive and negative integers. It has the information regarding the number of UC variables required for each unit referred to as the header and the operation periods describing the commitment of the unit at each hour of the planning period. For instance, the particle in Fig. 4 requires four integers to describe the scheduling of unit 1. The first integer represents header 1 (=3) and this describes the number of UC variables required for unit 1. The remaining three integers represent the duty cycles. Similarly,

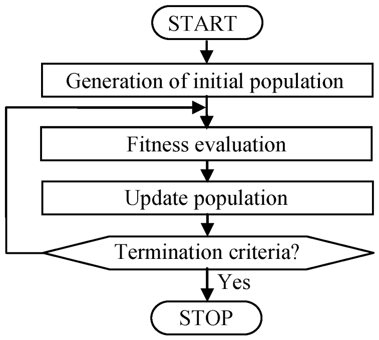

Fig. 5. Flowchart illustrating the basic PSO algorithm.

unit 2 is represented by six integers. The first integer or header 2 (=5) indicates that five integers are used for describing the scheduling of unit 2 and the next five integers represent the operation periods of unit 2.

V. Variable-Dimension Approach

For simplicity, the proposed approach is described using the standard PSO algorithm. The algorithm basically consists of three major processes: initialization, fitness evaluation, and population update as shown in Fig. 5.

A. Initialization

This process consists of generating initial conditions for each particle in the swarm. The header for each unit is generated randomly between lower and upper bound using (2). The minimum number of variables required for each unit is described by minimum up/down time and the maximum extends to the planning period. When header equals to the lower bound, the approach resembles fixed duty cycle method and to binary method when header equals the upper bound:

$$header^i = rand\left(\frac{\mathrm{PT}}{\mathrm{MUT^i}+1}, \mathrm{PT}\right). \tag{2}$$

The remaining dimensions of the particle are calculated using (3). The first dimension for each unit $(\mathrm{d}=1)$ is generated with consideration to the previous state, $previous$ of the unit. For (3) at the bottom of the page, $\mathrm{d}>1$, the dimensions are generated using MUT/MDT and unscheduled planning time, where $PT_{un}^{id} = PT - \sum_{k=1}^{d-1}|x_{ik}|$ and A,B,C are random integers which can either be $+1$ (ON) or -1 (OFF). The $rand$ function generates random integers between the specified lower and upper bounds. If the upper bound happens to be less than the lower bound, the function returns zero. This type of initialization will ensure that the sum of UC variables for each unit sum up to the planning period and always satisfy the MUT/MDT constraints. Apart from the UC variables, the particle also has economic dispatch, ED variables. These are continuous variables and are randomly generated between the maximum and minimum generation limits of the generators.

B. Fitness Evaluation

The procedure for fitness calculations is explained by the flowchart in Fig. 6. Let us consider a particle to consist of two sets of variables: UC variables, Y, and ED variables, Z. The flowchart takes the current position of the particle as the input and delivers a new corrected particle position, fitness value and the magnitudes of the constraint violations. Initially the integer based UC variables are converted to binary variables, $\mathrm{U^{old}}$. Before evaluating the fitness and validating the constraints, the particle traverses through the reserve manager and demand equalizer operations. The reserve manager ensures that the reserve requirement is always fulfilled. This is done by switching some of the economical units. This operator also takes care of the unwanted reserve by switching off some of the committed units. This operator therefore generates a new set of UC variables, $\mathrm{U^{new}}$. The demand equalizer ensures that the economical units share the maximum load. This operator maintains the balance between generation and load. This operation leads to a new set of ED variables, $\mathrm{Y^{new}}$. Before terminating the fitness evaluation algorithm, the new binary UC variables, $\mathrm{U^{new}}$ are converted back to integer variables, $\mathrm{Z^{new}}$.

C. Population Update

The particles in the swarm explore and exploit the search space by traveling at a certain velocity. The current velocity, $\mathrm{v_i^{k+1}}$, is guided by its previous velocity, $\mathrm{v_i^k}$, the distance of the particle to its local best, $\mathrm{x_p^k}$, and the distance to the global best, $\mathrm{x_g^k}$ as shown in (4). The current position, $\mathrm{x_i^{k+1}}$, is obtained by adding the current velocity to the previous position, $\mathrm{x_i^k}$. In variable-dimension optimization approach, the vectors $\mathrm{v_i^k}$, $\mathrm{x_p^k}$, $\mathrm{x_g^k}$, and $\mathrm{x_i^k}$ have different dimensions. So information has to be shared among different dimensional particles. The association between $\mathrm{x_i^k}$ and $\mathrm{x_p^k}$ to generate $\mathrm{x_i^{k+1}}$ is illustrated in Fig. 7. Unit 1 in $\mathrm{x_i^k}$ is three dimension where as $\mathrm{x_p^k}$ is two dimensions. The first two dimensions of $\mathrm{x_i^k}$ and $\mathrm{x_p^k}$ can communicate to result in $\mathrm{x_i^{k+1}}$ for the corresponding dimension. The third dimension in $\mathrm{x_i^k}$ is an alien space for $\mathrm{x_p^k}$, so there is no contribution of $\mathrm{x_p^k}$ to this dimension. If the other vectors $\mathrm{v_i^k}$ and $\mathrm{x_g^k}$ also do not contribute to this dimension, then a random velocity is added to this dimension. For unit 2, $\mathrm{x_i^k}$ has five dimensions where as $\mathrm{x_p^k}$ has

$$x_{id} = \begin{cases} \begin{cases} A * rand(MUT - |previous|, PT), & \text{for } previous < MUT \text{ and } d=1 \\ A * rand(MUT, PT), & \text{for } previous > MUT \text{ and } d=1 \end{cases} \\ B * rand\left(MUT, PT_{un}^{id}\right), & \text{for } d = 2,3,\ldots, header^i - 1 \\ C * PT_{un}^{id}, & \text{for } d = header^i \end{cases} \tag{3}$$

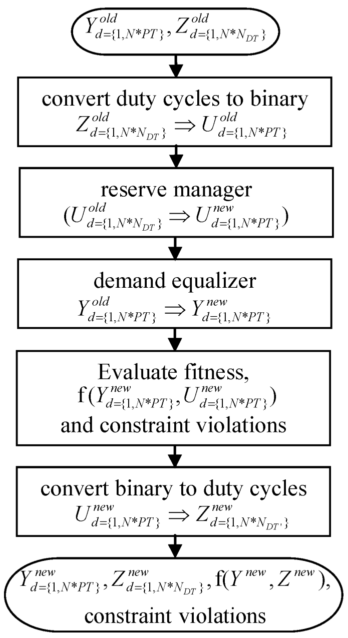
$Y^{old}_{d=\{1,N*PT\}}, Z^{old}_{d=\{1,N*N_{DT}\}}$
convert duty cycles to binary $Z^{old}_{d=\{1,N*N_{DT}\}} \Rightarrow U^{old}_{d=\{1,N*PT\}}$
reserve manager $(U^{old}_{d=\{1,N*N_{DT}\}} \Rightarrow U^{new}_{d=\{1,N*PT\}})$
demand equalizer $Y^{old}_{d=\{1,N*PT\}} \Rightarrow Y^{new}_{d=\{1,N*PT\}}$
Evaluate fitness, $f(Y^{new}_{d=\{1,N*PT\}}, U^{new}_{d=\{1,N*PT\}})$ and constraint violations
convert binary to duty cycles $U^{new}_{d=\{1,N*PT\}} \Rightarrow Z^{new}_{d=\{1,N*N_{DT'}\}}$
$Y^{new}_{d=\{1,N*PT\}}, Z^{new}_{d=\{1,N*N_{DT}\}}, f(Y^{new}, Z^{new})$, constraint violations

Fig. 6. Flowchart illustrating the fitness evaluation process.

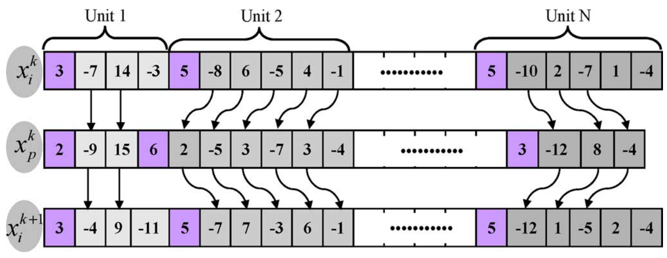

Fig. 7. Information exchange between particle k and its local best.

six dimension. Since $\mathrm{x}_{\mathrm{i}}^{\mathrm{k}+1}$ has the same dimension as the $\mathrm{x}_{\mathrm{i}}^{\mathrm{k}}$, the first five dimension of $\mathrm{x}_{\mathrm{i}}^{\mathrm{k}}$ can correspond to their

$$\begin{aligned} v_i^{k+1} &= \chi\left(wv_i^k + c_1r_1\left(x_p^k - x_i^k\right) + c_2r_2\left(x_g^k - x_i^k\right)\right) \\ x_i^{k+1} &= x_i^k + v_i^{k+1} \end{aligned} \tag{4}$$

respective dimensions in $\mathrm{x}_{\mathrm{p}}^{\mathrm{k}}$ to contribute to $\mathrm{x}_{\mathrm{i}}^{\mathrm{k}+1}$. Since the sixth dimension is missing in $\mathrm{x}_{\mathrm{i}}^{\mathrm{k}}$, no information is required in this dimensional space and therefore omitted from this update process. Similar procedures are utilized in communicating with $\mathrm{v}_{\mathrm{i}}^{\mathrm{k}}$ and $\mathrm{x}_{\mathrm{g}}^{\mathrm{k}}$. After a particle is updated, it is ensured that the sum of the absolute values of UC variables for each unit sum up to the planning period.

VI. Simulation Algorithm

The variable-dimension optimization approach illustrated above is implemented in a new version of PSO algorithm. The motive behind the new version of PSO is to develop a black box optimization tool. The algorithm should involve minimum human assistance and should provide good solutions to a wide variety of real-time applications. PSO should be used like a built in function call. The function should only have inputs like problem dimension, lower and upper bounds of the decision variables, number of equality and inequality constraints and a stopping criteria. This function call should be associated with an external function which can take an instance of decision variables and return the corresponding fitness value and constraint violations. No other information about the problem can be exchanged. The particles should be able to adjust their flight and the swarm should maintain a suitable population diversity to automatically self tune their search process and discover good solutions.

The above requirements are fulfilled by the new version of PSO called Self-adaptive particle swarm optimization, here after referred to as APSO [15]. It is a parameter free optimization tool. The particles of this algorithm have the capability to modify their search strategy based on their personal performances. The particles and the swarm adapt to the situations to find the global optimal solution. It is free from the burden of selecting the most appropriate swarm size. The algorithm is inspired from the nomad community. Nomads are groups of people who move from place to place following the seasonal availability in search for a better living. This algorithm simulates the moving strategy of different sized groups of nomads called "Tribes". The basic structure of the algorithm is derived from the TRIBE-PSO introduced by Clerc [16].

A. Swarm Evolution

The search process is ignited by minimal set of $\mathrm{N_T}$ tribes. Each tribe consists of a fixed set of particles, $\mathrm{N_P}$. Each particle is associated with a certain velocity and fitness. The particles try to memorize its previous two performances and also its best performance. At the end of $\mathrm{N_G}$ generations, the tribes are evaluated. The particle is judged based on its two previous performances. The performances can be an improvement $(+)$, status quo $(=)$, or a deterioration $(-)$. A bad particle is one which deteriorates or shows no progress $(--, -=, ==, =-, +-)$. On the other hand a good particle is one whose performances are improvements$(=+, ++, +=)$. The TRIBE is also labelled as good or bad based on the majority of its good or bad particles. At the first iteration, the previous two performances of the particles are initialized to their current position. If the TRIBE happens to be a bad performer, it indicates that its current information about the search space is not enough to find good solution. At this instant, this tribe will add more information by generating a new tribe with $\mathrm{N_P}$ particles. Two-third of the new particles are randomly generated while the remaining one-third particles are generated in the close proximity of the best particle in the current TRIBE. The second bad TRIBE will add another $\mathrm{N_P}$ particles to the newly generated TRIBE, whereas the good TRIBE has majority of good particles. It means that the TRIBE has enough information about the good solution. If this TRIBE has more than one particle, the worst performing particle is identified. This particle may also be good but its close associates in the same TRIBE are much better and this particle has the same or less information as the rest of its associates. So there is no risk in deleting this particle. The good TRIBE therefore will eliminate one of its least performing particle. The updated swarm is again allowed to explore for $\mathrm{N_G}$ iterations. This process continues until the stopping criteria is reached. The process of evolution indicates that new particles will be born only when they are required. Particles which do not contribute to the search process are eliminated. The swarm always has potential particles enthusiastic enough to search for an optimal solution. This substantially helps the algorithm to find the solution within few iterations.

B. Flying Strategies

Different particles in the TRIBEs have different levels of performances. Before each TRIBE evaluation, the particles are given enough time to explore. During the evaluation, the particles are compared based on their performance and least performing particles may be eliminated. But, are all the particles given a fair chance to improve? Both good and bad particles are allowed to explore for the same amount of time. Since all particles have the same flying strategies, the good particles will always perform better and the bad particle will never reach the standards of the good particle. In order to remove this bias, different particles have different flying strategies. Based on their previous performances, the particles automatically judge the right flying strategy. The particles are categorized into three groups. A worst particle is one whose performances are deterioration $(--, = \ -)$. Bad particles and good particles comprise the following combinations $(- \ =, ==, +-)$, $(+=, =+, -+, ++)$, respectively. The worst particles follow a random search strategy, the bad particle prefers pivot strategy and the good particles follow Gaussian update strategy. These strategies are explained in [17].

C. Neighborhood Topologies

The standard PSO version employs a star topology where each particle is directly connected to the global best performer of the swarm. When the global best performer catches a local minima, all particles are naturally attracted to it. Since each particle is directly connected to the global best, information is rapidly propagated and the whole swarm may prematurely converge on this local minima. In order to avoid such untoward convergence, several information or neighborhood topologies are suggested. In APSO algorithm, each particle has a fixed neighborhood. The flight of the particle is consistently monitored and guided by these neighbors. The global best performer is no more common to all the particles. Each particle selects its own global best performers from its neighborhood and not from the whole swarm.

D. Description of the Proposed Approach

The optimization process starts with $\mathrm{N_T}$ tribes and eventually evolves to explore the entire problem space. Each particle in a tribe is assisted by a set of associates in its neighborhood. Each tribe will try to locate a minimum and in the process also communicate with the other tribes to discover the global solution. The algorithm consists of two iterative loops. One loop controls the ultimate termination of the search process. The second loop allows the swarm to explore and exchange information among its neighbors before they are finally evaluated. So this loop controls the evolutionary process. A fixed set of generation, $\mathrm{N_G}(=10)$, is set as the termination criteria for this loop. The algorithm is explained in the following steps:

```
1) Swarm initialization
    FOR i = 1 to $\mathrm{N_T}$
        FOR j = 1 to TOTAL_PARTICLES_TRIBE[i]
          a) Randomly generate a particle with position,
             X and velocity, V
          b) Assign a fitness value, f (X) for each particle
          c) Initialize the local best $(\mathrm{X_p})$ and previous
             two performances $(\mathrm{X_{p1}}, \mathrm{X_{p2}})$ to the current
             position, X.
          d) Generate a neighborhood for the particle.
             The neighborhood list of a particle consists
             of all the particles in the current tribe and a
             random parent from the other tribe.
        END
    END
2) WHILE $(\mathrm{CYCLE} < \mathrm{N_G})$
    DO
            FOR i = 1 to TOTAL_TRIBES
                FOR j = 1 to
                TOTAL_PARTICLES_TRIBE[i]
                  1) choose the right flying strategy
                     and update velocity and position
                  2) correct the BOUND violations
                  3) evaluate the fitness
                  4) memorize $\mathrm{X_{p1}}, \mathrm{X_{p2}}$
                  5) update $\mathrm{X_p}$, $\mathrm{X_g}$
                  6) COUNT++
                  7) IF $(\mathrm{COUNT} >$
                     $\mathrm{MAX\_FUNCTION\_}$
                     $\mathrm{EVALUATIONS})$ GOTO
                     STEP 7
                END
            END
            CYCLE++
    END WHILE
4) Evaluate the TRIBES and perform the necessary adaptations
    FOR i = 1 to TOTAL_TRIBES
      a) Evaluate the TRIBE (GOOD or BAD)
      b) IF $(\mathrm{TRIBE} == \mathrm{BAD})$
             i) Identify the best performer in the TRIBE
            ii) $\mathrm{TRIBE\_COUNT} =$
                $\mathrm{TOTAL\_TRIBES} + 1$, for the first
                BAD TRIBE only
           iii) Generate a new TRIBE with $\mathrm{N_P}$ particles
                • Two-third particles are randomly
                  generated
                • One-third particles are generated close
                  to best performer of TRIBE[i]
            iv) Generate the neighborhood for these
                particles. The neighbors of a particle
                include all the particles in the new TRIBE
```

TABLE I
PROBLEM DIMENSION AND OPERATION COSTS FOR THE THREE METHODS

	UC variables	ED variables	Total Costs
Binary Approach	240	240	€ 567,062.57
Fixed Duty Cycles	50	240	€ 560,673.00
Variable-Dimension	30	240	€ 560,338.63

```
            and the best performer of the current BAD
            TRIBE
    c) IF (TRIBE == GOOD)
        i) Identify the worst particle and remove it.
        ii) Update the neighborhood topologies by
            replacing the deleted particle with its
            global best performer.
  END
  CYCLE = 0, TOTAL_TRIBES = TRIBE_COUNT
6) GOTO STEP 2
7) END
```

VII. Numerical Results

The proposed variable-dimension optimization approach was tested on a bench set UCP adapted from [18]. The test case has ten generators supplying the load for a scheduling period of 24 h. The spinning reserve is assumed to be 5% of the load. In order to illustrate the effectiveness of the proposed approach, the results are compared with binary programming approach and fixed duty cycle approach. All the approaches are implemented using the adaptive PSO algorithm. Since this PSO version is a black box optimization tool, no algorithm related parameter tuning is required.

In binary programming approach, the particle consists of 10 $\times$ 24 $(\text{units} = 10, \text{planning time} = 24\text{ h})$ binary UC variables and 10 $*$ 24 ED variables. A total of 480 variables are required for this approach, whereas in fixed duty cycle approach only 5 $*$ 10 UC variables are required. There is a reduction of 80% in the number of UC variables. In variable-dimension approach only 30 UC variables were needed (Table III). Unit 1, 2, and 10 required one UC variable, unit 3, 4, 8, and 9 required three variables and the remaining used five UC variables. The reduction of UC variables in this case is 88%. The numbers of ED variables required are 10 $*$ 24 and are the same for all the methods. This information is shown in Table I.

The total operation cost using the binary approach amounts to €567 062. The fixed duty cycle approach was able to find a solution which is €6389.57 cheaper than the solution obtained from the binary approach. The variable-dimension approach produced the best results among the three methods. The operation costs were €6723.94 and €334.37 less compared to binary and fixed duty cycle approaches, respectively. The total fuel and start-up costs for the global best particle in the swarm are shown in Figs. 8 and 9, respectively. The final fuel costs for fixed duty cycle and variable-dimension approach are almost equal. But the variable-dimension approach requires more function evaluations to converge on the final solution. This is because the dimension of the search space is not fixed until 7500 function evaluations. The ambiguity in the search space dimension results in extra exploration of the swarm. The binary approach had fewer start-up costs compared to fixed duty cycle method. But the variable-dimension approach was the best among the three methods at any point of time during the search process. The changes in the search space dimension with respect to the best particle in the swarm are illustrated in Fig. 10. The best particle starts with an initial search dimension of 278 (240 ED variables +38 UC variables), gradually reduces to 274 (240 + 34) and finally settles at 270 (240 + 30). The cost curves closely follow the dimension change pattern. When ever there is a reduction in the search space dimension, the particles were able to efficiently exploit the reduced search space to produce considerable reduction in the costs. This implies that with reduced dimensionality, the algorithm was able to generate better UC schedule. The results obtained by duty cycle and variable dimension PSO approach are better than the results obtained by various other evolutionary algorithms presented in [19].

The UC schedule generated by the binary method is presented in Table II. The economical units 3–6 are under utilized and the expensive units 7–9 had longer operation periods. This was the

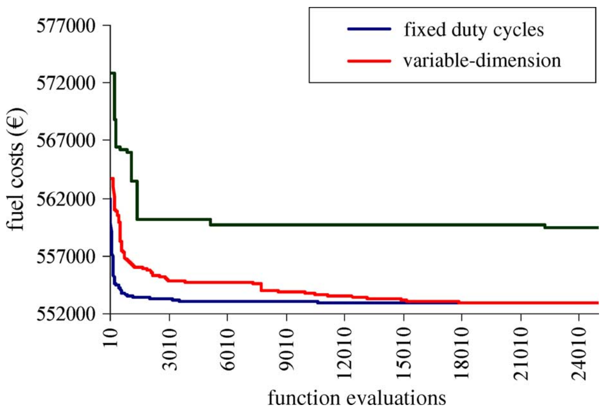

Fig. 8. Improvement in the total fuel costs with the search progress.

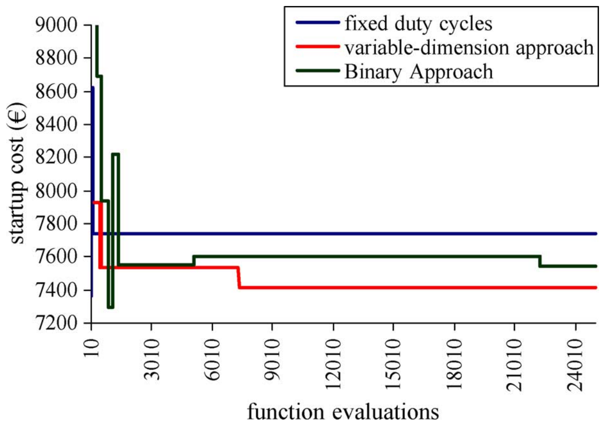

Fig. 9. Improvement in the total start-up costs with the search progress.

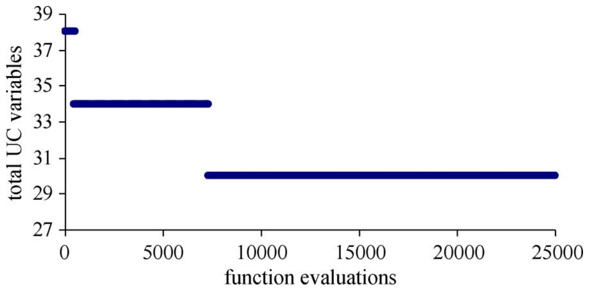

Fig. 10. Evolution of the search dimension with search progress.

TABLE II
BEST SOLUTION OBTAINED FROM THE BINARY METHOD

Unit	Unit schedule
1	111111111111111111111111
2	111111111111111111111111
3	000001111111111000111100
4	000000111111111001111000
5	000111111111111111111100
6	000000000111110000000000
7	000000001111100000010000
8	000000000011000000000000
9	000000000001000100011010
10	000000000000000000000000

TABLE III
BEST SOLUTION OBTAINED FROM THE (A) FIXED DUTY CYCLE AND (B) VARIABLE-DIMENSION APPROACH

(A)

	Duty cycles				
	1	2	3	4	5
1	24	0	0	0	0
2	24	0	0	0	0
3	-6	8	-5	3	-2
4	-5	10	-2	6	-1
5	-3	18	-3	0	0
6	-10	6	-2	3	-3
7	-9	4	-6	1	-4
8	-11	1	-12	0	0
9	-8	4	-12	0	0
10	-24	0	0	0	0

(B)

	Duty cycles				
	1	2	3	4	5
1	24				
2	24				
3	-6	16	-2		
4	-5	18	-1		
5	-3	12	-3	3	-3
6	-8	6	-5	2	-3
7	-9	4	-6	1	-4
8	-11	1	-12		
9	-10	2	-12		
10	-24				

TABLE IV
OPTIMAL SOLUTION OBTAINED FROM THE VARIABLE-DIMENSION METHOD

Hour	Production Cost (€)	Startup Cost (€)	Generation Schedule (MW)									
1	13882.84	0.0	245.0	455.0	0	0	0	0	0	0	0	0
2	14567.76	0	439.0	311.0	0	0	0	0	0	0	0	0
3	16310.57	0.0	445.2	404.8	0	0	0	0	0	0	0	0
4	18657.51	1790.0	437.7	455.0	0	0	57.3	0	0	0	0	0
5	19666.56	0.0	440.0	455.0	0	0	105.0	0	0	0	0	0
6	21862.86	1116.2	454.3	455.0	0	130.0	60.7	0	0	0	0	0
7	23303.51	1097.7	410.0	455.0	130.0	130.0	25.0	0	0	0	0	0
8	24650.71	0.00	354.0	455.0	130.0	99.0	162.0	0	0	0	0	0
9	26600.99	339.3	452.0	455.0	130.0	130.0	113.0	20.0	0	0	0	0
10	29404.65	519.3	451.0	455.0	130.0	130.0	155.3	53.7	25.0	0	0	0
11	31330.83	60.0	449.1	455.0	130.0	130.0	162.0	75.1	38.8	0	10	0
12	33224.74	60.0	453.0	455.0	130.0	130.0	162.0	80.0	25.0	55	10	0
13	29606.00	0.0	420.8	455.0	127.3	130.0	162.0	80.0	25.0	0	0	0
14	26806.36	0.0	403.0	455.0	130.0	130.0	162.0	20.0	0	0	0	0
15	24163.62	0.0	451.0	455.0	130.0	130.0	34.0	0	0	0	0	0
16	21089.72	0.0	445.0	455.0	130.0	20.0	0	0	0	0	0	0
17	20240.25	0.0	440.2	455.0	84.8	20.0	0	0	0	0	0	0
18	21924.93	0.0	441.2	455.0	130.0	73.8	0	0	0	0	0	0
19	24156.15	1599.2	453.3	455.0	130.0	130.0	31.7	0	0	0	0	0
20	29600.92	833.1	431.1	455.0	116.9	130.0	162.0	80.0	25.0	0	0	0
21	26606.86	0.0	450.5	455.0	130.0	130.0	114.5	20.0	0	0	0	0
22	21918.89	0.0	447.8	455.0	111.3	85.9	0	0	0	0	0	0
23	17895.61	0.0	315.0	455.0	0	130.0	0	0	0	0	0	0
24	15450.86	0.0	428.0	372.0	0	0	0	0	0	0	0	0
Total	**552923.68**	**7414.9**										

reason for higher overall operation cost. The operation schedules obtained using fixed duty cycle and variable-dimension approaches are presented in Table III(a) and (b), respectively. Both the methods had completely different scheduling schemes for most of the generators. In common, both the methods had allocated longer operation schedules for low cost units and the expensive units are scarcely used. The high cost units are used only during the peak load periods and are shut down during the off-peak periods. Variable-dimension approach proved to be the most efficient among the two methods. The operation periods of the cost ineffective units are further reduced in the proposed approach. More over the schedules of the cost effective units (especially units 3 and 4) are arranged so that the frequency of the start-ups are reduced to the minimum. This is the reason for the reduction in the start-up costs as shown in Fig. 9. The final solution using the variable-dimension approach is presented in Table IV.

The effect of reserve manager operator on the quality of the solution can be seen from Fig. 11. The amount of reserve allocated by various methods is shown in this figure. In binary method without reserve operator, there was huge unwanted reserve allocated during the low demand hours. When there was a sudden reduction in demand between hours 15:00 to 19:00, the algorithm was unable to reschedule the units efficiently. The units that are in operation during the first peak demand continue to operate even during this sudden dip. The excess unwanted reserve allocated during this period comes at a price and accounts for the additional operation costs. All the methods with reserve operators provided optimal reserve allocation which was very close to the required reserve capacity.

The convergence properties of the three methods are shown in Fig. 12. The binary method had the worst convergence. The

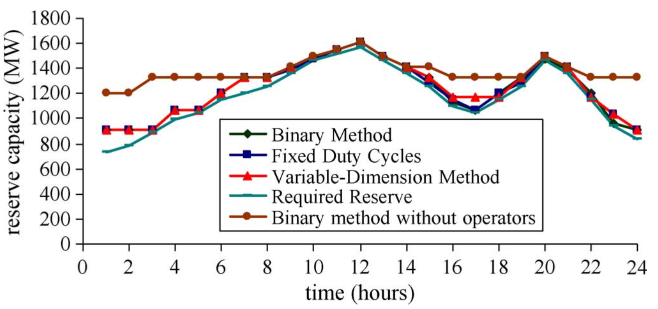

Fig. 11. Reserve allocated by different methods with and without reserve operator.

fixed duty cycle method provided better convergence than the variable-dimension approach. In the proposed approach, lot of information is either lost or ignored during the early stages of

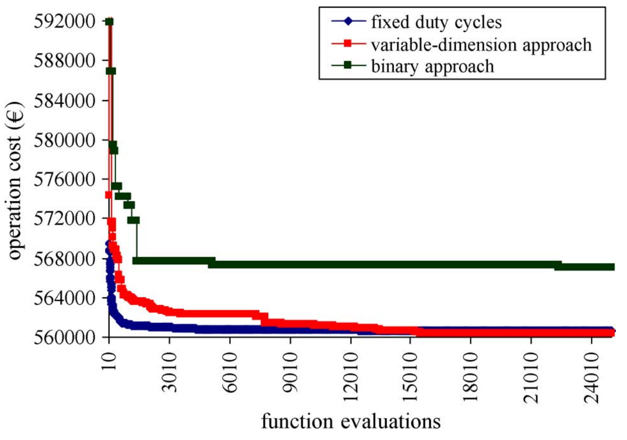

Fig. 12. Convergence properties of the three UC methods.

TABLE V
BEST SOLUTION FOR LARGE SYSTEMS

Problem size	ICGA	PSO-LR	EP	MA	Variable Dimension
20	1,127,244	1,128,281	1,125,494	1,128,192	1,123,792
40	2,254,123	2,252,330	2,249,093	2,249,589	2,247,213
60	3,378,108	3,377,718	3,371,611	3,370,820	3,372,006
80	4,498,943	4,499,347	4,498,479	4,494,214	4,492,633
100	5,630,838	5,623,607	5,623,885	5,616,314	5,617,204

the optimization process due to the differences in the dimensionality among the particles in the swarm. But once the search process identified the right problem dimension, the convergence property was better than the fixed dimension approach.

The proposed variable-dimension approach is further tested on 20-,40-,60-,80-, and 100-unit system. The larger systems are obtained by duplicating the ten-unit system in [4]. The demand and reserve are scaled accordingly. The simulation results obtained for these systems are compared with other evolutionary algorithms such as ICGA [4], hybrid PSO-Langrangian relaxation [20], Evolutionary Programming [21], and Memetic Algorithm [22] and listed in Table V. The computational time required to solve the UCP ranged from 50 s for ten-unit system to 60 min for the 100-unit system. Although the proposed approach required more time for the larger systems, the quality of the end results are much better than the results obtained by other binary and integer based evolutionary algorithms.

VIII. CONCLUSION

A new innovative UC variable modeling using variable-dimension optimization approach is presented in this paper. The simulation results on a ten generator test problem imply that nearly 88% reduction in problem dimension is possible. The idle duty cycles present in the fixed duty cycle can be completely eliminated. The advantage of reduced search dimension is reflected in the improved quality of the final solution. Due to few decision making UC variables, the swarm was able to effectively optimize the scheduling of various generation units. This research has proved that it is possible to define an optimization problem even without declaring the problem dimension.

REFERENCES

[1] W. L. Snyder, Jr., H. D. Powell, Jr., and J. C. Rayburn, "Dynamic programming approach to unit commitment," *IEEE Trans. Power Syst.*, vol. 2, no. 2, pp. 339–347, May 1987.
[2] A. I. Cohen and M. Yoshimura, "A branch-and-bound algorithm for unit commitment," *IEEE Trans. Power App. Syst.*, vol. PAS-102, pp. 444–451, Feb. 1983.
[3] N. R. Jimenez and A. J. Conejo, "Short-term hydro thermal coordination by Lagrangian Relaxation: Solution to the dual problem," in *Proc. IEEE Power Eng. Soc. Winter Meeting*, 1998.
[4] S. A. Kazarlis, A. G. Bakirtzis, and V. Petridis, "A genetic algorithm solution to the unit commitment problem," *IEEE Trans. Power Syst.*, vol. 11, no. 1, pp. 83–92, Feb. 1996.
[5] T. O. Ting, M. V. C. Rao, and C. K. Loo, "A novel approach for unit commitment problem via an effective hybrid particle swarm optimization," *IEEE Trans. Power Syst.*, vol. 21, no. 1, pp. 411–418, Feb. 2006.
[6] A. H. Mantawy, Y. L. Abdel-Magid, and S. Z. Selim, "Unit commitment by Tabu search," *Proc. Inst. Elect. Eng., Gen., Transm., Distrib.*, vol. 145, no. 1, pp. 56–64, Jan. 1998.
[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, Nov. 27–Dec. 1 1995, vol. 4, pp. 1942–1948.
[8] C. Li, R. B. Johnson, A. J. Svoboda, C. Tseng, and E. Hsu, "A robust unit commitment algorithm for hydro-thermal optimization," *IEEE Trans. Power Syst.*, vol. 13, no. 3, pp. 1051–1056, Aug. 1998.
[9] A. H. Mantawy, Y. L. Abdel-Magid, and S. Z. Selim, "Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem," *IEEE Trans. Power Syst.*, vol. 14, no. 3, pp. 829–836, Aug. 1999.
[10] N. P. Padhy, "Unit commitment using hybrid models: A comparative study for dynamic programming, expert system, fuzzy system and genetic algorithms," *Elect. Power Energy Syst.*, vol. 23, pp. 827–836, 2000.
[11] S.-J. Huang and C.-L. Huang, "Application of genetic-based neural network to thermal unit commitment," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 654–660, May 1997.
[12] B. Tessema and G. G. Yen, "A self adaptive penalty function based algorithm for constrained optimization," in *Proc. IEEE Congr. Evolutionary Computation*, 2006.
[13] I. G. Damousis, A. G. Bakirtzis, and P. S. Dokopoulos, "A solution to the unit-commitment problem using integer-coded genetic algorithm," *IEEE Trans. Power Syst.*, vol. 19, no. 2, pp. 1165–1172, May 2004.
[14] V. S. Pappala and I. Erlich, "A new approach for solving the unit commitment problem by adaptive particle swarm optimization," in *Proc. IEEE Power and Energy Soc. General Meeting—Conversion and Delivery of Electrical Energy in the 21st Century*, Jul. 20–24, 2008, pp. 1–6.
[15] V. S. Pappala, M. Wilch, S. N. Singh, and I. Erlich, "Reactive power management in offshore wind farms by adaptive PSO," in *Proc. Int. Conf. Intelligent System Application to Power System, 2007*.
[16] M. Clerc, "TRIBES, a parameter free particle swarm optimizer," presented at the OEP'03, Paris, France, in French.
[17] V. S. Pappala, I. Erlich, K. Rohrig, and J. Dobschinski, "A stochastic model for the optimal operation of a wind-thermal power system," *IEEE Trans. Power Syst.*, vol. 24, no. 2, pp. 940–950, May 2009.
[18] Z.-L. Gaing, "Discrete particle swarm optimization algorithm for unit commitment," in *Proc. IEEE Power Eng. Soc. General Meeting*, Jul. 13–17, 2003, vol. 1.
[19] A. Ghasemi, M. M. Farsangi, and H. Nezamabadi-Pour, "Unit commitment scheduling using binary differential evolution algorithm," *OPSEARCH*, vol. 46, no. 1, pp. 108–122, Mar. 2009.
[20] H. H. Balci and J. F. Valenzuela, "Scheduling electric power generators using particle swarm optimization combined with the Lagrangian relaxation method," *Int. J. Appl. Math. Comput. Sci.*, vol. 14, no. 3, pp. 411–421, 2004.
[21] K. A. Juste, H. Kitu, E. Tunaka, and J. Hasegawa, "An evolutionary programming solution to the unit commitment problem," *IEEE Trans. Power Syst.*, vol. 14, no. 4, pp. 1452–1459, Nov. 1999.
[22] J. Valenzuela and A. Smith, "A seeded memetic for large unit commitment problems," *J. Heurist., Alg.*, vol. 8, no. 2, pp. 173–195, 2002.

Venkata Swaroop Pappala was born in 1981. He received the B.E. degree in electrical engineering from the Faculty of Electrical Engineering, Nagarjuna University, Nagarjuna, India, in 2002, and the M.Sc. degree in electrical engineering with emphasis on power and automation from University Duisburg-Essen, Duisburg, Germany, in 2005, where he is currently pursuing the Ph.D. degree.

His research interests include stochastic optimization under uncertainty using evolutionary algorithms.

Istvan Erlich (SM'06) was born in 1953. He received the Dipl.-Ing. degree in electrical engineering and the Ph.D. degree from the University of Dresden, Dresden, Germany, in 1976 and 1983, respectively.

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. From 1991 to 1998, he worked with the consulting company EAB, Berlin, Germany, and the Fraunhofer Institute IITB Dresden, respectively. During this time, he also had a teaching assignment at the University of Dresden. Since 1998, he has been a Professor and Head of the Institute of Electrical Power Systems at the University of Duisburg-Essen, Duisburg, Germany. His major scientific interest is focused on power system stability and control, modeling and simulation of power system dynamics including intelligent system applications.

Dr. Erlich is a member of VDE.