Solution of Unit Commitment Problem Using Improved Shuffled Frog Leaping Algorithm

J.Mary Anita
Research Scholar, School of Electrical Sciences, Noorul Islam University
Kumaracoil, India.(e-mail: anitajayaseelan@rediffmail.com)

Dr. I. Jacob Raglend
Professor, School of Electrical Sciences, Noorul Islam University
Kumaracoil, India (e-mail: jacobraglend@rediffmail.com)

D.P.Kothari
General Director, Vindhya Group of Institutions
Indore, India (e-mail: dpk0710@yahoo.com)

Abstract: *This paper presents a new approach for solving Unit commitment (UC) problem of thermal units based on a new evolutionary algorithm known as Shuffled Frog Leaping Algorithm (SFLA). The integer coded algorithm is developed based on the behavior of group of frogs searching for a location that has the maximum amount of available food. This algorithm involves local search and shuffling process and these are repeated until a required convergence is reached. The efficiency and effectiveness of the SFLA is improved by introducing a cognition component which allows the frog to adjust its position according to the thinking of the frog itself along with global and local best of the population. In this proposed method of SFLA for the UC problem, the scheduling variables are coded as integers, so that the minimum up/down time constraints can be handled directly without using any penalty functions. To verify the performance of the proposed algorithm it is applied to the IEEE 14, 30, 56,118 bus systems and 10, 20 unit systems for a one day scheduling period. The results of 10 and 20 units systems are compared with the existing methods available in the literature. The results obtained are quite encouraging for the Unit Commitment problem when compared with the existing methods. The algorithm and simulation are carried out using Matlab software.*

Key Words: *Shuffled Frog Leaping Algorithm, Frogs, Unit commitment, Economic Dispatch, Local Search, Integer Coded Unit Commitment*

Abbreviations

T -Time horizon of Unit commitment (hrs)
P_D^t- Real power demand at tth hour (MW)
P_i- Real power generation of ith Unit (MW)
X_i- Commitment Status of ith Unit
P_{imax}- Maximum real power generation of ith Unit (MW)
P_{imin}- Minimum real power generation of ith Unit (MW)
SUC_i- Start- up cost of ith unit (\$)
RD_i- Allowable change in real power of ith unit
SDT_i -Shut down duration of ith unit(hrs)
MDT_i- Minimum down time of ith unit (hrs)
MUT_i- Minimum up time of ith unit (hrs)
HSC_i-Hot startup cost of ith unit (\$)
CSC_i- Cold start up cost of ith unit (\$)
CSH_i- Cold start hour of ith unit(hrs)
N-No., of generating units
C- No., of cycles
T_i^c- Duration of cth cycle of ith unit (hrs)

1. Introduction

Operating under the present competitive environment, Unit Commitment (UC) is essential since a significant amount of savings can be obtained by a sound UC decision. UC is a process of determining the minimal production cost generator turn ON/turn OFF schedule and real power outputs of committed units while meeting the forecasted demand over a scheduling horizon of usually between 24hrs to 168hrs (1 day- 7 days). The obtained UC schedule should also satisfy the global constraints (power balance, spinning reserve and environmental) and local constraints like operational and physical constraints of every unit [1] [2] [3]. Since it has to satisfy more number of constraints the UC Problem is a complex, non-linear, mixed integer optimization problem. UCP's combinatorial nature curtails any attempt to develop a rigorous mathematical optimization method.

Mathematical solution to UCP involves simultaneous solution of two sub problems. (i) The mixed integer non-linear programming problem of determining the generating units to be committed each hour of the planning horizon, while considering system capacity requirements. (ii)The quadratic programming problem of economic dispatch among the committed units during every specific hour of operation.

Complete enumeration can give an optimal solution but its excessive computational complexity and computational reserve requirements have made it not suitable for large scale real time systems. The need for practical, cost-effective UC solutions led to the development of UC algorithms.

In Literature there exist more no of methods to solve UCP [4]. The available approaches may be (i) numerical solution techniques such as Priory List (PL), Dynamic programming (DP) [5], Lagrangian relaxation, Branch and bound, and MIP. The PL is simple and fast but always led to a solution of higher operating cost. DP suffers with the problem of sub optimal solution while truncating some of the non-feasible solution, to reduce its computational time. The computational requirements increases with system size which limits DP to very small sized systems.

In LR [6] method the feasible solution is generated by an appropriate co-ordination technique while minimizing the (duality gap) difference between the primal and dual objective functions. LR is modifiable to the model characteristics of specific utilities of power system. The constraints can be easily added to the main objective function but the major drawbacks are its sensitivity and the availability of dual optimal solution. (ii) The stochastic search methods such as Genetic Algorithm (GA), [7], [8], [9], Particle Swarm Optimization (PSO) [11] [25], Ant colony optimization, and Bacterial foraging (BF) [13]. These methods are capable of handling complex nonlinear constraints to provide a high quality solution. Several GA approaches are reported in literature [18], [19], [20], [21] &[27].

Usually GA approaches use genetic guided search for the primal UC problem variables. GA suffers with long computation time due its random selection of GA operator. Integer coded GA for UC is more efficient than binary coded when accompanied by a suitable GA operator. Evolutionary programming (EP) differs from GA from the method of solution coding and selection of candidates for reproduction.

Bacterial foraging algorithm is based on the foraging behavior of E.coli bacteria present in the human intestine. An integer coded UC using BFA is reported in [13]. In [17] straight forward (SF) is presented which decomposes the UC into three sub problems. This method is based on linearization of quadratic cost functions of the generating units.

Decomposition and co-ordination of constraints are discussed in[23] [24]. Semi definite programming methods have also been used in solving UCP.[26].

The combination of EA's with local search was named memetic algorithms (MA's). MA's have been found more successful and efficient and more effective than traditional EA's for same problem domains. SFLA is one among the available memetic algorithm. Eusuff and Lansey first introduced SFLA [14], [15], [16] in 2003.

This method is based on the behavior of frogs search for the location that has the maximum amount of available food. Possible solutions are randomly generated to create the initial population of frogs. And these frogs are grouped into memeplexes. Memetic evolution step (local search) is carried out within every memeplex and a shuffling is done between the memeplexes. This process is repeated till a required convergence is reached. This algorithm has been successfully applied for several engineering optimization problems.

The integer coded UC [10] is used. The minimum up/down constraints are directly coded hence no need for any penalty function for these constraints. The performance of this algorithm is tested for various IEEE systems such as 14, 30, 56, 118 bus systems and 10, 20 generating unit systems for one day scheduling.

The organization of this paper is as follows. In section 2 the mathematical model of the UCP is presented. Section 3 details the idea of shuffled frog leaping algorithm and the introduction of cognitive component to modify the leaping rule. In section 4 the implementation of improved SFLA along with their mathematical equations is discussed. In section 5 & 6 the simulation results and conclusion are discussed respectively.

2. Mathematical modeling of UC

The total operating cost of electrical energy includes fuel cost, start up cost and shut down cost. The fuel costs are calculated using the data of unit heat rate & fuel price information which is normally a quadratic equation of power output of each generator at each hour determined by Economic Dispatch(ED).

$$F_c(P_i) = A_i + B_iP_i + C_iP_i^2 \quad (1)$$

Where, A_i, B_i, C_i are coefficients of cost matrix. The total fuel cost for the entire scheduling horizon 'T' is given by

$$TFC = \sum_{t-1}^{T} \sum_{i=1}^{N} F_cP_i * X_i(t) \quad (2)$$

Where, Xi (t) is the status of i^th unit at t^th hour.

Startup cost is the cost involved in bringing the thermal unit online. Start up costs is expressed as a function of the number of hours the units has been shut down, (Exponential when cooling and linear when banking). Shut down costs are defined as a fixed amount for each unit/shutdown. However it is not taken into account in this paper.

A simplified start up cost model is used as follows.

$$SUC_i = \begin{cases} HSC_i, if\ MDT_i \le DT_i < MDT_i + CSH_i \\ CSC_i, if\ DT_i > MDT_i + CSH_i \end{cases} \quad (3)$$

There are several constraints that must be satisfied by the UCP.

i) *System power balance*

The sum of generation of all the committed units at t^{th} hour must be greater than or equal to the demand at a particular hour 't'.

$$\sum_{i=1}^{N} X_i(t)P_i(t) \le P_D(t), \ t = 1,2,3 \ldots \ldots .T \quad (4)$$

ii) *System spinning reserve requirements*

In order to maintain certain degree of reliability an excess capacity of generation is essential to immediately take over when a running unit fails, or unexpected load occurs. A fixed reserve policy is used in this paper and the mathematical equation is given by

$$\sum_{i=1}^{N} X_i(t)P_i(t) \le P_D(t) + P_R(t), \quad t = 1,2,3 \ldots \ldots .T \quad (5)$$

iii) *Min up/down time*

A committed unit can be turned off only after it satisfies its minimum up time values, at the same time, a reserved unit can be turned on only after it satisfies, its min down time. This is due to the fact that the temperature of a thermal unit can be increased or decreased only gradually.

$$\begin{cases} T_i^c \ge MUT_i \ if \ T_i^c > 0 \\ -T_i^c \ge MDT_i \ if \ T_i^c < 0 \end{cases} \quad (6)$$

Where, T_i^c is a signed integer representing ON/OFF status duration of c^{th} operating cycle of the i^{th} unit.

iv) *Initial Operating status of generating units*

The initial operating status of every unit should take the last day's previous schedule into account, so that every unit satisfies it's minimum up/down time.

v) *Maximum/Minimum power limits*

Every unit has its own maximum/minimum power level of generation, beyond and below which it cannot generate

$$P_{imin} \le P_i^t \le P_{imax} \quad (7)$$

vi) *Ramp rate constraints*

Since, the temperature of a thermal unit can only be increased or decreased gradually; the output also can be increased or decreased within a limit. The response rate constraints of the unit limits the power generation and is given by

$$P_{imax}(t) = \min(P_{imax}, P_i^{t-1} + \tau RD_i)$$

$$P_{imin}(t) = \max(P_{imin}, P_i^{t-1} + \tau RD_i) \quad (8)$$

Where τ=60 min.

3. Improved shuffled frog leaping algorithm

(A). Original Shuffled Frog Leaping Algorithm

SFLA is a metaheuristic optimization method which combines the GA's memetic evolution and PSO's social behavior. It is a combination of deterministic and random strategies. The deterministic approach allows the algorithm to use the search space effectively to guide its heuristic search and the random approach ensures flexibility and robustness of the search process.

SFLA mainly based on the behavior of group of frogs searching for the location that has the maximum amount of available food. This algorithm is capable of solving discrete and continuous optimization problems. It is also capable of solving non-linear non-differentiable, multi modal optimization problems.

This algorithm has been successfully applied for several engineering applications like bridge deck repair, water source distribution, determination of optimal discrete pipe size for new pipe networks, data clustering, job shop scheduling etc, The most promising benefit of this algorithm is its faster convergence speed.

The SFLA involves a population of possible solutions defined by a set of virtual frogs. This set of virtual frogs is partitioned into subsets know as memeplexes. The memeplexes can be perceived as a set of parallel frog cultures attempting to reach some goal. Frog leaping improves an individual frog and enhances its performance towards the goal. Within each memeplex each frog holds different ideas and the idea of each frog can be used to infect the ideas of other frogs.

The process of passing information between the frogs of a memeplex is known as local search or memetic evolution step. After a defined number of memetic evolution step the virtual frogs are shuffled and reorganized so that the quality of memeplex is improved. Shuffling enhances the meme quality after infection and ensures the cultural evolution towards any particular interest.

The process of memetic evolution and shuffling are repeated unit a required convergence is reached. This is given graphically in Fig.1. The following steps are involved in SFLA.

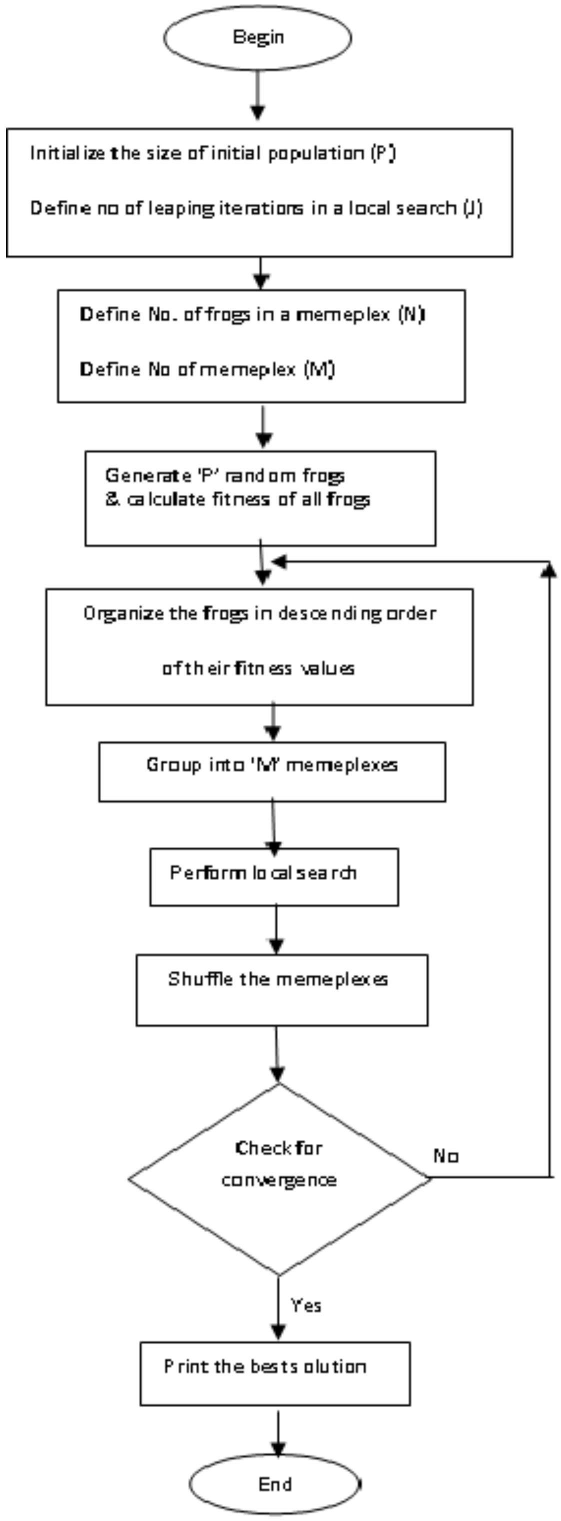
Begin
Initialize the size of initial population (P)
Define no of leaping iterations in a local search (J)
Define No. of frogs in a memeplex (N)
Define No of memeplex (M)
Generate 'P' random frogs
& calculate fitness of all frogs
Organize the frogs in descending order
of their fitness values
Group into 'M' memeplexes
Perform local search
Shuffle the memeplexes
Check for convergence
No
Yes
Print the best solution
End

Fig.1. Flow chart of SFLA

Step: I. Formation of Initial population

1) Population size (no. of frogs) P is chosen

2) P no. of frogs is generated randomly within the search space.

3) The position of every frog is defined as $X_i = X_i^1, X_i^2, \ldots\ldots X_i^D$, Where D is the no. of variables

4) The fitness of search frog is calculated as

$$fitness = \begin{cases} \frac{1}{f(x)} + C, & for\ minimization \\ f(x) + C, & for\ maximization \end{cases} \quad (9)$$

Where, f(x) is the objective function and c is a constant to ensure the fitness a positive value.

Step: II. Grouping of Frogs into Memeplexes:

1) The frogs are sorted in descending order according to their fitness values.
2) The entire population of ‘P’ frogs are grouped into ‘M’ memeplexes, and each memeplex is formed so that each memeplex consists of ‘N’ no of frogs (P=MXN).
3) The partitioning of memeplexes is done so that each memeplex have frogs with lower and higher fitness values. For this the first frog goes to 1st memeplex, the second frog goes to 2nd memeplex, the mth frog to mth memeplex and m+1th frog goes to 1st memeplex. This procedure is illustrated in Fig.2.

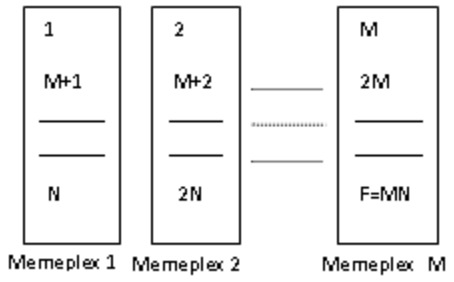

Fig. 2. Formation of Memeplexes

Step: III Local search process: (Memetic evolution step)

1) Within each memeplex, the frogs with worst (X_w) & best (X_b) fitness values are identified. Also the frog with global fitness X_g is also identified.
2) The frog with worst fitness is leaped towards the best frog by a random vector.
$D_i = rand(1) * (X_b - X_w)$ (10)
$X_w = X_w + D_i \quad D_{imin} < |D_i| < D_{imax}$
3) The fitness of the new leaped worst frog is calculated. If there is no improvement in fitness, the leaping vector is calculated with
$D_i = rand(1) * (X_g - X_w)$ (11)
$X_w = X_w + D_i \quad D_{imin} < |D_i| < D_{imax}$
4) The fitness of the new leaped worst frog is calculated. If there is no improvement, then Xw is replaced with a new random frog. This is illustrated in Fig. 3.
4) The steps 1, 2, 3, & 4 are repeated for some specific number of iterations.

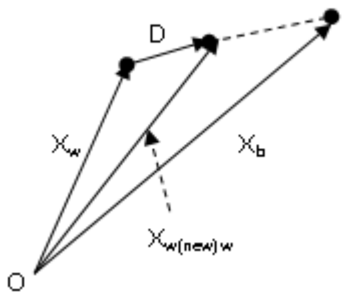

Fig.3. Frog Leaping Rule

Step: IV Shuffling Process:

After local search in every memeplex is completed shuffling of memeplex is done, and the frogs are reorganized in descending order of fitness values and again grouped into memeplex and local search process is carried out.

Step: V. Steps I, II, III, IV are repeated until any one of the following conditions is satisfied.

i) The relative change in the fitness of the global frog within a number of consecutive shuffling iterations is less than a pre-specified tolerance.

ii) The maximum predefined number of shuffling iterations have been reached.

(B). Modified Frog Leaping Rule

In the original SFL algorithm, every frog updates its position according to the best solution because of the influence of the local best solution. According to the frog leaping rule used in SFLA the leaping of the frog to its new position is restricted between the worst and the best frog position, and not beyond the best frog position. It may restrict the search space of the local search algorithm. This may lead to slower convergence and convergence within local optimal point. The above mentioned problem is overcome by the introduction of the cognitive (12) component. The ability and stability of the algorithm is improved by the introduction of the cognition component [15]. Introduction of this component allows the frog to adjust its position according to the thinking of the frog itself along with best frog within the memeplex or the global best frog of the population. The coordinates of current position of each frog is entered into the formulas for the measure of error of the estimate of target values, and it is moved towards the new position. This is repeated for a defined number of times. While moving towards the multivariate space, the individuals compare their current error value with the best error value they have attained at any point up to that iteration. The lowest error value is termed as the best error value *Pbest*$_{j}$., and the position where the *Pbest*$_{j}$ is evaluated is termed as P_j. The difference P_i - X_i indicates the distance between the individual's previous and current position. Each element

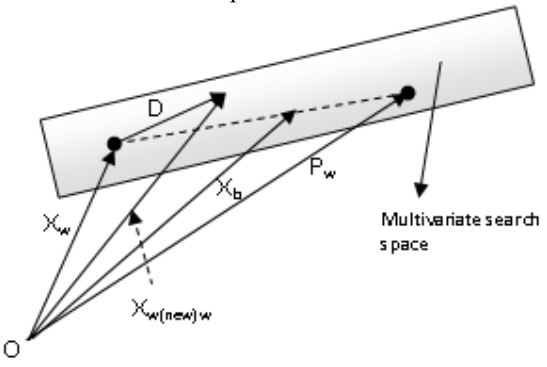

Fig.4. Improved Frog Leaping Rule

of the above distance vector is weighted by a positive random number in the range [0 1].Because of the introduction of this component the frog is not restricted to move along the line segment. Now, the leaping of the frog takes place in a widened search space avoiding premature convergence (i.e) definitely beyond the best frog, the global optimal point. The mathematical equation of the new modified frog leaping rule is given by the equation (12). Figure (4) represents the modified frog leaping rule.

$$D_i = rand(1) * (P_w - X_w) + rand(1) * (X_b - X_w)$$
$$X_w = X_w + D_i \quad D_{imin} < |D_i| < D_{imax} \tag{12}$$

The modified local search process of improved SFLA is illustrated pictorially in Fig.5.

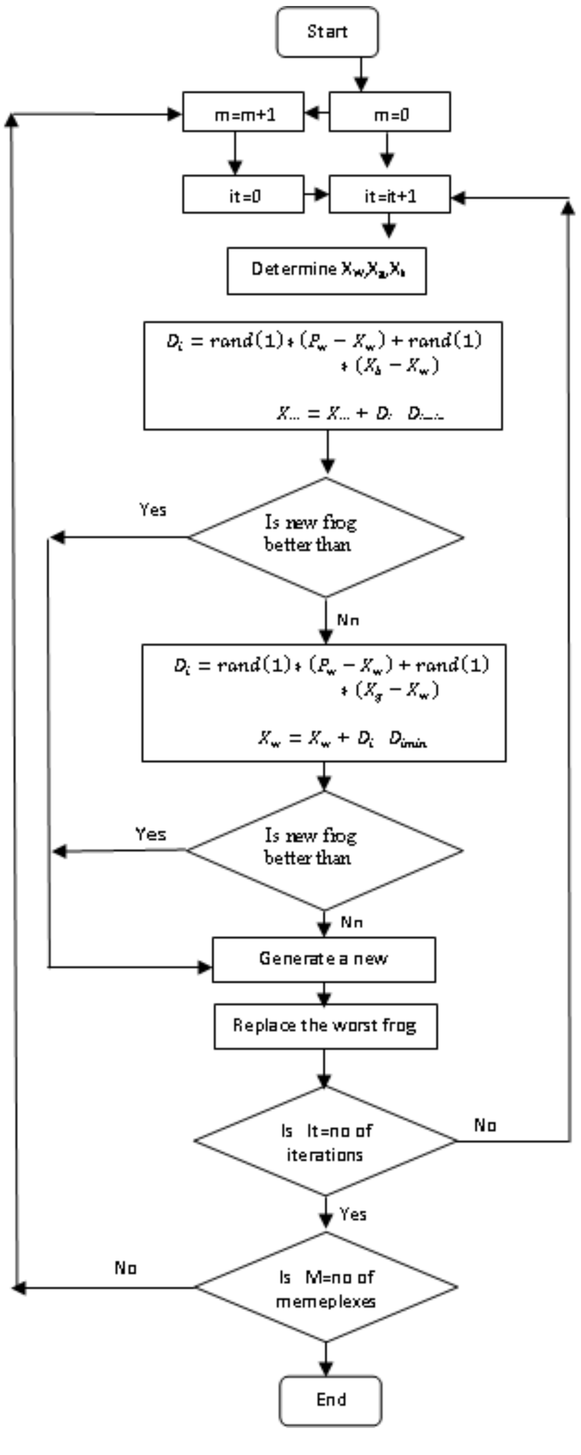
Start
m=m+1
m=0
it=0
it=it+1
Determine X_w,X_g,X_b
$D_i = rand(1) * (P_w - X_w) + rand(1) * (X_b - X_w)$
$X_w = X_w + D_i \quad D_{imin}$
Yes
Is new frog better than
No
$D_i = rand(1) * (P_w - X_w) + rand(1) * (X_g - X_w)$
$X_w = X_w + D_i \quad D_{imin}$
Yes
Is new frog better than
No
Generate a new
Replace the worst frog
Is It=no of iterations
No
Yes
Is M=no of memeplexes
No

Fig.5.Local Search Process of Improved Shuffled Frog Leaping Algorithm

4. Implementation of SFLA to UCP

Definition of frog position

The position of a frog in integer coded SFLA for UCP consists of a sequence of alternatively signed integers representing the duration of ON/OFF cycles of units during the scheduling horizon. A positive integer in the frog vector represents the duration of continuous ON state of a unit whereas the negative integer represents the duration of continuous OFF state of a unit.

The size of a frog is decided by the no of units (N) and no of cycles(C). No of cycles(C) is determined by the load peaks and minimum up and down time of units.In this paper the load profile with two load peaks as given in Fig. 6 is considered. From Fig. 6 it is understood that the peak units will have 5 cycles and the base units and intermediate units will have 1&3 respectively.

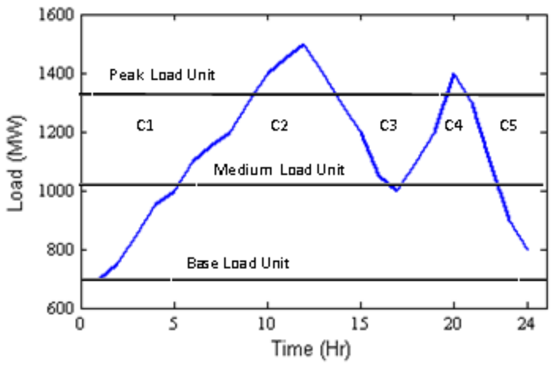

Fig.6. Load Curve for 24Hrs

So, the no of cycles vary between 1 and 5. But for simplicity and to increase the search space, the peaking unit cycles are taken for base and intermediate units and their remaining cycles are assumed to be zero. For a 10 unit, 5 cycle system the size of the frog for a one day scheduling is 1×10×5. Definition of frog from ON/OFF cycle duration of units and the UC schedule is illustrated in Table. 1.

Creating Initial Population

A part of a frog representing the operating schedule of a particular unit during the scheduling horizon should be formed such that $\sum_{c=1}^{C}|T_i^c| = T$. The values of T_i^c are randomly generated to form the initial population. The procedure is as follows.

Formation of first cycle: (T_i^1)

T_i^1 is selected so that the unit continues the operating mode(ON/OFF) of the last cycle of the previous day scheduling (T_i^0) for at least as many hours required so that no units are violating its minimum up/down time values.

$$T_i^1 = \begin{cases} +rand\left(\max\left(0, MUT_i - T_i^0\right), T\right) \; if \; T_i^0 > 0 \\ -rand\left(\max\left(0, MDT_i + T_i^0\right), T\right) \; if \; T_i^0 < 0 \end{cases} \qquad (13)$$

Formation of in between cycles (T_i^c, $1<c<C$)

The cycles between the initial and last cycles are generated considering the units minimum up/down time, the scheduling horizon (T) and the duration of previous cycles (i.e.) duration of *(c-1)* prior cycles.

If $T_i^{c-1} < 0$, indicates cycle 'c' is positive and it represents an ON status of ith unit.

$$T_i^c = \begin{cases} +rand(MUT_i, RT_i^{c-1}), if \; (RT_i^{c-1} > MUT_i) \\ +RT_i^{c-1}, otherwise \end{cases} \qquad (14)$$

If $T_i^{c-1} > 0$, indicates cycle 'c' is negative and it represents an OFF status of ith unit.

$$T_i^c = \begin{cases} -rand(MDT_i, RT_i^{c-1}), if \; (RT_i^{c-1} > MDT_i) \\ -RT_i^{c-1}, otherwise \end{cases} \qquad (15)$$

Where, RT_i^{c-1} indicates the remaining scheduling period after allocating the first (c-1) cycles.

$$RT_i^{c-1} = T - \sum_{p=1}^{c-1}\left|T_i^p\right| \qquad (16)$$

Formation of last cycle (T_i^C)

The duration of last cycle 'C' is decided by the duration of C-1 prior cycles (i.e.) $T_i^C = RT_i^{C-1}$ If due to random generation of cycle duration the entire scheduling interval (T) is completed within the first few cycles 'c' < C then the remaining $c+1$..... C cycles are assigned to Zero. From this type of representation it is well known that the minimum up / down time constraint is satisfied in the coding stage itself and hence there is no need for any penalty function for this constraint in the objective function.

Leaping of worst solution

After formation of memeplex, the local search process is carried out in each memeplex. Leaping of worst frog towards the best frog is done by the random vector $D_i = rand(1) * (P_w - X_w) + rand(1) * (X_b - X_w)$ or by the random vector $D_i = rand(1) * (P_w - X_w) + rand(1) * (X_b - X_w)$ Addition of this vector to the X_w may lead to change in X_w and it needs the following modifications.

i) Sum of all T_i^c of unit '*i*' will not be equal to '*T*'. To adjust the following correction is done. $(T_i^1, T_i^2, \ldots.. T_i^C) = \frac{T.*(T_i^1, T_i^2, \ldots.. T_i^C), i=1,2,\ldots.N}{\sum_{k=1}^{C} |T_i^k|}$ (17)

(ii) The rand (1) function generates a random number between 0 and 1 the parameter which is a non-integer number and this may lead the parameter of X_w to a non-integer values. But X_w should be an integer vector. Hence to convert the non integer parameters of X_w to integer the following correction is done by

$$X_w^1 = Round(X_w) \qquad (18)$$

Table: 1
Sample frog for 10 units 5 cycle system

Unit	1	2	3	4	5
1	T_1^1	T_1^2	T_1^3	T_1^4	T_1^5
	24	0	0	0	0
2	T_2^1	T_2^2	T_2^3	T_2^4	T_2^5
	24	0	0	0	0
3	T_3^1	T_3^2	T_3^3	T_3^4	T_3^5
	-4	19	-1	0	0
4	T_4^1	T_4^2	T_4^3	T_4^4	T_4^5
	-5	17	-2	0	0
5	T_5^1	T_5^2	T_5^3	T_5^4	T_5^5
	15	-3	3	-3	0
6	T_6^1	T_6^2	T_6^3	T_6^4	T_6^5
	-8	6	-3	4	-3
7	T_7^1	T_7^2	T_7^3	T_7^4	T_7^5
	-8	6	-5	3	-2
8	T_8^1	T_8^2	T_8^3	T_8^4	T_8^5
	-9	4	-6	1	-4
9	T_9^1	T_9^2	T_9^3	T_9^4	T_9^5
	-10	2	-12	0	0
10	T_{10}^1	T_{10}^2	T_{10}^3	T_{10}^4	T_{10}^5
	-11	1	-12	0	0

(iii) The above round of correction may again lead to the sum not equal to '*T*' Hence to adjust the values of T_i^c, the last non-zero cycle is adjusted as follows,

$$T_i^l = T - \sum_{k=1}^{l-1} \left|T_i^k\right|, i = 1,2,\ldots.N \qquad (19)$$

iv) After generation of new X_w, the minimum up / down time should be adjusted so that there is no violation in this constraint. The correction in '*c*' cycle should be followed by correction in '*c+1*' cycle for adjusting the sum of T_i^c to '*T*'

For $Ti^1>0$, if $Ti^1<max\ (0,MUT_i - T_i^0)$, then the duration of cycle 1& cycle 2 of unit '*i*' are changed as

$$\begin{cases} T_i^2 = T_i^2 - T_i^1 + \max(0, MUT_i - T_i^0) \\ \quad T_i^1 = \max(0, MUT_i - T_i^0) \end{cases} \qquad (20)$$

For $Ti^1<0$ *if* $-Ti^1< max\ (0,MDT_i + T_i^0)$, then the duration of cycle 1&cycle 2 of unit '*i*' are changed as

$$\begin{cases} \quad T_i^2 = T_i^2 - T_i^1 + \max(0, MDT_i + T_i^0) \\ T_i^1 = -\max(0, MDT_i + T_i^0) \end{cases} \qquad (21)$$

For $T_i^c>0$ if $T_i^c<MUT_i$ for $c=2, \ldots..C-1$, the cycles '*c*' and $c+1$ of unit '*i*' are changed

$$\begin{cases} T_i^{c+1} = T_i^{c+1} - T_i^c + MUT_i \\ T_i^c = MUT_i \end{cases} \qquad (22)$$

For $T_i^c<0$ if $-T_i^c<MDT_i$ for $c=2\ldots..\ C-1$ the cycles '*c*' and $c+1$ of unit '*i*' are changed

$$\begin{cases} T_i^{c+1} = T_i^{c+1} - T_i^c - MDT_i \\ T_i^c = MDT_i \end{cases} \qquad (23)$$

After all the above corrections are carried out, on X_w, the Economic Dispatch (ED) should be carried out for each hour of scheduling horizon for all committed units. Then the fitness value is calculated. The sample frog is given in Table.1

Computation of fitness function

The objective function of UC using SFLA has two terms, and they are the total operation cost and the penalty functions for violating system constraints (spinning reserve & power balance).

$$TC = \sum_{t=1}^{T} \sum_{i=1}^{N} FC_i\,(P_i^t) * X_i(t) + SU_T + SD_T \quad (24)$$

The penalty function has two terms. The first term for spinning reserve violation and is given by

$$\prod_{res} = \omega \sum_{t=1}^{T} \frac{1}{D^t}\ R((D^t + R^t) - \ \sum_{i=1}^{N} X_i(t) P_{imax}^t \qquad (25)$$

The second term for excessive capacity is given by

$$\prod_{cap} = \omega \sum_{t=1}^{T} \frac{1}{D^t}\ R(\sum_{i=1}^{N} X_i(t) P_{imin}^t - D^t) \qquad (26)$$

Where '*ω*' depends on maximum operating cost of the system over a scheduling period '*T*'. $\omega = \alpha\, \mathrm{T} \sum_{i=1}^{N} FC_i(P_{imax})$ (27) where α is a constant.

Now the objective is to minimize the fitness function

$$Fitness = A/(TC + \square_{res} + \square_{cap}) \qquad (28)$$

$A= 10^8$. '*A*' is a system dependent constant added for avoiding the fitness value from obtaining too small values. This should be of the order of the system maximum operating cost.

4. Simulation results

The proposed Improved SFLA to UC has been tested for various IEEE test bust systems such as IEEE 14, 30, 56, 118 buses and also for 10, 20 unit system for a scheduling period of 24hrs. The load and system data for 10 unit system is listed in appendix (1) & (2). The system data for 20 unit system is obtained by duplicating the 10 unit system data and the load is doubled. The proposed algorithm was also tested for IEEE56bus, IEEE118bus systems and their load and generator data are taken from www.motor.ece.iit.edu/data/IEEE118

Table: 2 .
Generator schedule of 30 bus system (6 units)
For 24 hrs

Hour	Power Generations of Units(MW)					
	1	2	3	4	5	6
1	87.3	33.7	15.0	10.0	10.0	10.0
2	111.2	38.4	16.4	10.0	10.0	10.0
3	138.5	42.9	17.6	10.0	10.0	10.0
4	170.9	47.3	18.8	10.0	10.0	10.0
5	185.1	49.0	19.3	10.0	10.0	10.0
6	175.2	47.8	19.0	10.0	10.0	10.0
7	162.5	45.2	18.3	0	10.0	10.0
8	135.2	40.8	17.0	0	10.0	10.0
9	127.8	38.0	16.2	0	10.0	0
10	103.1	32.9	15.0	0	10.0	0
11	91.5	30.5	15.0	0	10.0	0
12	102.3	32.7	15.0	0	10.0	0
13	110.4	34.4	15.2	0	10.0	0
14	122.2	36.9	15.9	0	10.0	0
15	140.7	40.4	16.9	0	10.0	0
16	160.1	44.0	17.9	0	10.0	0
17	171.4	46.1	18.5	0	10.0	0
18	167.4	45.3	18.3	0	10.0	0
19	163.3	44.6	18.1	0	10.0	0
20	154.5	42.9	17.6	0	10.0	0
21	137.5	39.8	16.7	0	10.0	0
22	129.8	36.4	15.8	0	0	0
23	113.1	32.9	15.0	0	0	0
24	88.2	27.8	15.0	0	0	0

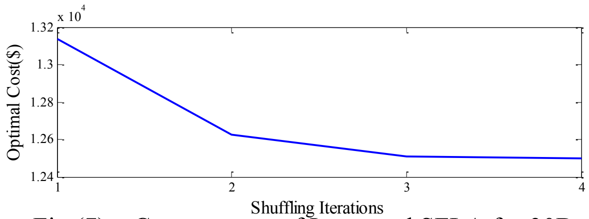

Fig (7) Convergence of Improved SFLA for 30Bus (6 units) system

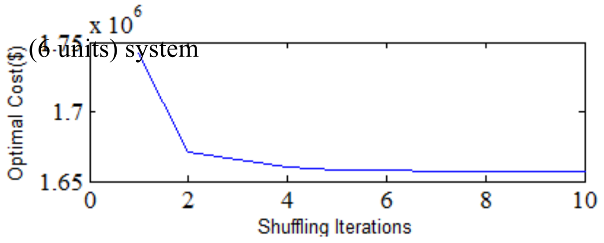

Fig (8) Convergence of Improved SFLA for 118Bus (54 units) system

The reserve requirement was 10% of the hourly load in all cases. The main parameters of SFLA have been taken from paper [22]. The initial population size for improved SFLA has been taken as 200 frogs. Grouping of 200 frogs is done between 20 memeplexes each with 10 frogs. Memetic evolution step is done for 10 iterations before each shuffling process. The improved SFLA program is developed and executed in MATLAB 2011.

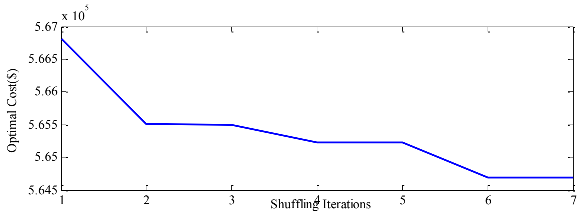

Fig (9) Convergence of Improved SFLA for a 10 unit system

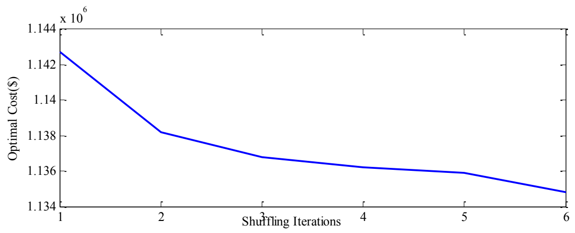

Fig (10) Convergence of Improved SFLA for 20 Units system

The results of generation scheduling along with their real power generation of the best solution for 30Bus (6 Units), 118Bus (54Units), 20 Units system are tabulated in Table 2 to 4. Table 5 gives the comparison of optimal cost between the original and improved SFLA for all test systems.

The shuffling iterations taken by original SFLA and improved SFLA are listed in Table 6. The optimal solution for all test systems is obtained between 4 to 7 shuffling iterations.

Table: 3 Generator schedule of 118 bus system (54 units) for 24 hrs

Hour	Power Generations of Units(MW)																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	0	201.5	201.5	0	0	0	0	201.5	350	0	0	0	0	0	0	0
2	0	0	0	179.2	179.2	0	0	0	0	179.2	350	0	0	0	0	0	0	0
3	0	0	0	150	135.7	0	0	0	0	135.7	350	0	0	0	0	0	0	0
4	0	0	0	150	100	0	0	0	0	100	105.1	0	0	0	0	0	0	0
5	0	0	0	150	100	0	0	0	0	100	278.1	0	0	0	0	0	0	0
6	0	0	0	150	146.6	0	0	0	0	146.6	350	0	0	0	0	0	0	0
7	0	0	0	201.3	201.3	0	0	0	0	201.3	350	0	0	0	0	0	0	0
8	0	0	0	150	236.2	0	25	0	0	261.2	350	0	0	25	0	25	0	0
9	0	0	0	150	100	0	25	0	0	291.2	350	0	0	25	0	25	0	0
10	0	0	0	265	300	0	100	0	0	300	350	0	0	100	0	25	0	0
11	0	0	0	150	300	0	100	0	0	300	350	0	0	100	0	100	0	0
12	0	0	0	175	300	0	25	0	0	300	350	0	0	25	0	25	0	0
13	0	0	0	150	100	0	25	0	0	276.3	350	0	0	25	0	25	0	0
14	0	0	0	150	100	0	25	0	0	171.2	350	0	0	25	0	25	0	0
15	0	0	0	265	300	0	100	0	0	300	350	0	0	100	0	25	0	0
16	0	0	0	235	300	0	100	0	0	300	350	0	0	100	0	100	0	0
17	0	0	0	300	300	0	25	0	0	300	350	0	0	25	0	25	0	0
18	0	0	0	300	300	0	100	0	0	300	350	0	0	100	0	25	0	0
19	5	0	0	150	300	0	100	0	0	300	350	0	0	100	0	100	0	0
20	5	0	0	300	300	0	100	0	30	300	350	0	0	100	0	100	0	0
21	0	5	30	300	300	0	100	0	0	300	350	0	0	100	0	100	0	0
22	0	5	0	150	300	0	100	0	0	300	350	0	30	100	30	100	8	8
23	0	0	0	175	300	0	100	0	0	300	350	30	0	100	0	25	0	0
24	0	0	0	150	300	0	100	0	0	300	350	0	30	25	8	25	8	8

Table 3 contd.,
Generator schedule of 118 bus system (54 units) for 24 hrs

Hour	Power Generations of Units(MW)																	
	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	0	250	250	0	0	200	200		420	420	201.3	0	0	0	0	0	0	201.3
2	0	250	250	0	0	200	200		398.7	398.7	179.2	0	0	0	0	0	0	179.2
3	0	250	250	0	0	200	200		354.2	354.2	135.7	0	0	0	0	0	0	150
4	0	207	207	0	0	95.2	95.2		27303	273.3	80	0	0	0	0	0	0	150
5	0	244.5	244.5	0	0	187.9	187.9		311.8	311.8	94.5	0	0	0	0	0	0	150
6	0	250	250	0	0	200	196.6		365.3	365.3	146.6	0	0	0	0	0	0	150
7	0	250	250	0	0	200	200		420	420	201.3	0	0	0	0	0	0	201.3
8	0	250	250	0	0	200	200	0	420	420	261.3	0	0	0	0	0	0	261.3
9	0	250	250	25	25	200	200	25	420	420	291.3	0	0	0	0	25	25	291.3
10	0	250	250	25	25	200	200	25	420	420	300	0	0	0	0	25	25	300
11	25	250	250	25	25	200	200	25	420	420	300	0	0	0	0	25	25	300
12	25	250	250	25	25	200	200	25	420	420	300	0	0	0	0	25	25	300
13	25	250	250	25	25	200	200	25	420	420	276.3	0	0	0	0	25	25	276.3
14	25	250	250	25	25	200	200	25	420	420	246.3	0	0	0	0	25	25	246.3
15	25	250	250	25	25	200	200	25	420	420	300	0	0	0	0	25	25	300
16	100	250	250	25	25	200	200	25	420	420	300	0	0	0	0	25	25	300
17	25	250	250	25	25	200	200	25	420	420	300	80	0	0	0	25	25	300
18	25	250	250	25	25	200	200	25	420	420	300	80	0	0	0	25	25	300
19	100	250	250	100	25	200	200	25	420	420	300	80	0	0	0	25	25	300
20	100	250	250	10	100	200	200	100	420	420	300	80	0	0	0	25	25	300
21	100	250	250	100	100	200	200	100	420	420	300	80	0	0	0	25	25	300
22	25	250	250	25	25	200	200	25	420	420	300	0	0	5	0	25	25	300
23	25	250	250	25	25	200	200	25	420	420	300	0	0	0	5	25	25	300
24	25	250	250	25	25	200	200	25	420	420	300	0	10	0	0	25	25	300

Table 3 contd.,
Generator schedule of 118 bus system (54 units) for 24 hrs

Hour	Power Generations of Units(MW)																	
	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
1	0	0	300	200	0	0	201.3	201.3	201.3	0	0	0	0	0	0	0	0	0
2	0	0	300	179.2	0	0	179.2	179.2	179.2	0	0	0	0	0	0	0	0	0
3	0	0	285.7	135.7	0	0	135.7	135.7	135.7	0	0	0	0	0	0	0	0	0
4	0	0	207	57	0	0	100	100	100	0	0	0	0	0	0	0	0	0
5	0	0	244.5	94.5	0	0	100	100	100	0	0	0	0	0	0	0	0	0
6	0	0	296.6	146.6	0	0	146.6	146.6	146.6	0	0	0	0	0	0	0	0	0
7	0	0	300	200	0	0	201.3	201.3	201.3	0	0	0	0	0	0	0	0	0
8	0	0	300	161.3	0	0	261.3	261.3	261.3	0	0	25	0	0	25	25	0	0
9	25	0	291.3	166.3	0	0	291.3	291.3	291.3	0	25	25	0	0	25	25	25	0
10	25	0	300	200	0	0	300	300	100	0	25	25	0	0	25	25	25	0
11	25	0	300	200	0	0	300	300	200	0	25	25	0	0	25	25	25	0
12	25	0	300	200	0	0	300	300	100	0	25	25	0	0	25	25	25	0
13	25	0	276.3	151.3	0	0	276.3	276.3	276.3	0	25	25	0	0	25	25	25	0
14	25	0	271.3	171.3	0	0	246.3	246.3	246.3	0	25	25	0	0	25	25	25	0
15	25	0	300	200	0	0	300	300	100	0	25	25	0	0	25	25	25	0
16	25	0	300	200	0	0	300	300	100	0	25	25	0	0	25	25	25	0
17	25	0	300	200	0	0	300	300	100	0	25	25	0	0	25	25	25	0
18	25	0	300	200	0	0	300	300	195	0	25	25	0	0	25	25	25	0
19	25	0	300	200	0	0	300	300	115	0	25	25	0	0	25	25	25	0
20	25	0	300	200	0	0	300	300	175	0	25	25	0	0	25	25	25	0
21	25	0	300	200	0	0	300	300	220	0	25	25	0	0	25	25	25	0
22	25	0	300	200	0	0	300	300	100	0	25	25	0	0	25	25	25	0
23	25	0	300	200	8	0	300	300	104	8	25	25	0	0	25	25	25	0
24	25	10	300	0	0	20	300	228	100	0	25	25	8	25	0	0	25	25

Table: 4 Generator schedule of 20 units system for 24 hrs

Hour	Power Generations of Units(MW)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	455	455	0	0	25	0	0	0	0	0	290	150	0	0	25	0	0	0	0	0
2	455	455	0	0	25	0	0	0	0	0	390	150	0	0	25	0	0	0	0	0
3	455	345	0	0	25	0	0	0	0	0	455	395	0	0	25	0	0	0	0	0
4	455	455	0	0	40	0	0	0	0	0	455	455	0	0	25	0	0	0	0	0
5	455	455	130	0	55	0	0	0	0	0	455	455	0	0	25	0	0	0	0	0
6	455	400	130	130	25	0	0	0	0	0	455	450	130	0	25	0	0	0	0	0
7	455	455	130	130	162	0	0	0	0	0	455	228	130	130	25	0	0	0	0	0
8	455	455	130	130	35	0	0	0	0	0	455	455	130	130	25	0	0	0	0	0
9	378	455	130	130	162	80	25	0	0	0	455	455	130	130	25	20	25	0	0	0
10	453	455	130	130	162	80	85	55	0	0	455	455	130	130	25	20	25	10	0	0
11	455	455	130	130	162	80	85	55	10	0	455	455	130	130	103	20	25	10	10	0
12	455	455	130	130	162	80	85	55	55	55	455	455	130	130	93	20	25	10	10	10
13	453	455	130	130	162	80	85	55	0	0	455	455	130	130	25	20	25	10	0	0
14	378	455	130	130	162	80	25	0	0	0	455	455	130	130	25	20	25	0	0	0
15	455	455	130	130	35	0	0	0	0	0	455	455	130	130	25	0	0	0	0	0
16	455	455	130	130	0	0	0	0	0	0	455	215	130	130	0	0	0	0	0	0
17	455	455	130	130	0	0	0	0	0	0	455	150	130	95	0	0	0	0	0	0
18	455	455	130	130	0	80	0	0	0	0	455	215	130	130	0	20	0	0	0	0
19	450	455	130	130	25	20	0	0	0	0	455	455	130	130	0	20	0	0	0	0
20	401	455	130	130	162	80	25	10	10	10	455	455	130	130	162	20	25	10	0	0
21	393	455	130	130	162	80	25	10	0	0	455	455	130	130	0	20	25	0	0	0
22	265	455	130	130	0	0	25	0	0	0	455	455	130	130	0	0	25	0	0	0
23	455	455	130	0	0	0	0	0	0	455	175	130	0	0	0	0	0	0	0	0
24	455	455	0	0	0	0	0	0	0	455	235	0	0	0	0	0	0	0	0	0

The results of improved SFLA for 10, 20 units systems are compared with the results of LRGA [7], ICGA [10] & original SFLA [12] and are listed in Table 16. It is obvious that SFLA has satisfactory results in comparison with other method. Fig. 7 to 10 shows the convergence rate of improved SFLA for the various systems considered in this work.

Table 5
Comparison of optimal cost of SFLA with improved SFLA

Sl.No	System	No.of generating Units	Optimal Cost($) SFLA	Optimal Cost($) Improved SFLA
1	IEEE14BUS	5	11171	10910
2	IEEE 30BUS	6	12768	12491
3	IEEE56BUS	7	51645	48875
4	IEEE 118BUS	54	1665800	1656700
5	10 UNIT	10	564769	564690
6	20 UNIT	20	1135800	1135800

It is better when compared to the 15 to 16 shuffling iterations taken by ordinary SFLA [12] to reach almost the same optimal solution. From Table (15) it can be concluded that the convergence rate of SFLA is improved by the introduction of cognition component.

Table 6
Comparison of shuffling iterations of SFLA with improved SFLA

	System	No. of generating Units	No. of Shuffling Iterations SFLA	No. of Shuffling Iterations Improved SFLA
1	IEEE14BUS	5	12	4
2	IEEE30BUS	6	16	4
3	IEEE56BUS	7	14	5
4	IEEE118BUS	54	14	7
5	10 UNIT	10	16	6
6	20 UNIT	20	16	5

Table: 7 Comparison of operation cost of various methods

No. of Units	Operational Cost ($)			
	LRGA [7]	IGCA [10]	SFLA [12]	Improved SFLA
10	565825	566404	564769	564690
20	1130660	1124892	1135800	1135800

6. Conclusion

The solution of UCP actually means, physically feasible and financially viable scheduling of generators. The existing methods for solving UCP have their inherent limitations of relaxation and computational efficiency. In this paper, a new evolutionary algorithm known as improved SFLA for UC problem was presented. The integer coding is used to code the parameters of UCP. This type of coding directly satisfies the min up/down time constraints, and no need for any penalty function for this constraint.

The performance of the proposed algorithm is tested for a one day scheduling for various test systems with 5 to 54 units. The results of 10 and 20 unit systems are compared with LR & ICGA and original SFLA method. The results of other systems are compared with original SFLA method. The simulation results shows that the production cost of SFLA is less than the other methods such as LR & ICGA. Also the test result shows that the introduction of cognition component improves the convergence rate of SFLA.

Our future work is directed towards the inclusion of emission and valve point loading effect along with the existing operational constraints.

7. References

[1]. A. J. Wood and B. F. Wollenberg, *Power Generation Operation and Control,* New York: Wiley, 1984

[2]. D.P. Kothari and J.S.Dhillon, *Power System Optimization*, Prentice Hall of India Pvt. Ltd., New Delhi, 2011.

[3]. D.P.Kothari and I.J.Nagrath, *Modern Power System Analysis*, 4th Edition, McGraw Hill, New York, 2011.
[4]. N.P. Padhy, *Unit commitment- A Bibliography Survey*, IEEE Trans. Power Systems, vol.19,no. 2, pp 1196-1205, May 2004.
[5]. W.L. Snyder, H.D. Powell, and J.C. Rayburn, *Dynamic Programming Approach to Unit Commitment*, IEEE Trans. Power Systems, vol.2, no. 2,pp.339-347, May 1987.
[6]. S.Virmani, E.C. Adrian, K.Imhof, *Implementation of a Lagrangian based Unit Commitment Problem*, IEEE Trans. Power Systems,vol.4, no.4,pp 1373-1380, Nov.1989.
[7]. S.A.Kazarlis, A.G.Bakitris, and V.Petridis, *A Genetic Algorithm Solution to the Unit Commitment Problem* , IEEE Trans. Power Systems, vol.11,no. 1, pp 83-92, Feb,1996.
[8]. K.S.Swarup and S.Yamashiro, *Unit Commitment Solution Methodology using Genetic Algorithm* , IEEE Trans. Power Systems, vol.17,no. 1, pp.87-91., Feb 2002.
[9]. J.M.Arroyo and A.J.Conejo, *A Parallel Repair Genetic Algorithm to Solve Unit Commitment Problem* , IEEE Trans. Power Systems, vol.17, no. 4, pp.1216-1224, Nov 2002.
[10]. I.G. Damousis, A.G. Bakirtzis,and P.S. Dokopolous , *A Solution to Unit Commitment Problem using Integer Coded Genetic Algorithm* , IEEE Trans. Power Systems, vol.19,no. 2, pp.1165-1172, May 2004.
[11]. W.Xiong , M.J.Li,and Y.Cheng, *An Improved Particle Swarm Optimization Algorithm for Unit Commitment Problem* , in Proc. ICICTA, 2008.
[12]. Javad Ebrahimi, Seyed Hossein Hosseinain, Gevorg B. Harehpatian, *Unit Commitment Problem Solution Using Shuffled Frog Leaping Algorithm* , IEEE Trans. Power Systems, vol.26, no.2, pp.573-581,May 2011.
[13]. M.Eslamian, as.ah.ahosseinian, B.vahidi, *Bacterial Foraging based Solution to the Unit Commitment Problem*, IEEE Trans. Power Systems,vol.24, no. 3, pp.1478-1488, Aug 2009.
[14]. M.M.Eusuff,K.E.Lansey, F.Pasha, *Shuffled Frog leaping: A Memetic Meta-Heuristic for Discrete Optimization*, Eng Optimiz, vol.38,no.2,pp.129-154, 2006
[15]. X.Zhang, X.Hu,G.Cui, Y.Wang, Y.Niu, *An Improved Shuffled Frog Leaping Algorithm With Cognitive Behavior,* in Proc., 7th World Congress, Intelligent Control and Automation, 2008.
[16]. T.H. Huynh, *A Modified Shuffled Frog Leaping Algorithm for Optimal Tuning of Multivariable PID Controllers*, in proc., ICIT 2008, pp 1-6, 2008.
[17]. S.H.Hosseini A.Khodaei,and F.Aminifar, *A Novel Straightforward Unit Commitment Method For Large Scale Power Systems* , IEEE Trans. Power Systems, vol.22,no. 4, pp.2134-2143, Nov 2007.
[18]. G.B. Sheble and T.T.Maifeld, *Unit Commitment By Genetic Algorithm and Expert System* , Electrical Power Systems Research, vol.30, no.2, pp.115-121,July/Aug.1994.
[19]. X.Ma,A.A.El-keib, R.E.Smith, and H.Ma, *A Genetic Algorithm Based Approach to Thermal Unit Commitment*, Electrical Power Systems Research, vol.34, pp.29-36,1995.
[20]. A.Rudolf and R.Bayreleithner, *A Genetic Algorithm for Solving the Unit Commitment Problem of a Hydro-Thermal Power System* , IEEE Trans. Power Systems, vol.14, pp.1460-1468, Nov 1999.
[21]. W.G.Xing and F.F.Wu, Genetic Algorithm Based Unit Commitment With Energy contract ,Int. J. Elect. Power ,vol.24, no.5 pp.329-336,June 2002.
[22]. J.Kennedy and R.C. Eberhart, *Particle Swarm Optimization*, in Proc, IEEE Conf. Neural Networks, 1995, vol.4, pp.1942-1948.
[23]. Qing Xia, Y.H.Song, Boming Zhang, Congqng Kang, Niande Xiang, *Effective Decomposition and co-ordination Algorithms for unit Commitment and economic Dispatch with security constraints*, Electric Power System Research, vol.53, pp.39-45, 2000.
[24] Bo Lu, Mohammed Shahidephour, *Unit commitment with flexible Generating Units",* IEEE Trans. Power Systems, vol.20, no.2, pp.1022-1034, May 2005.
[25]. J.Kennedy and R.C. Eberhart, *Particle Swarm Optimization*, in Proc, IEEE Conf. Neural Networks, 1995, vol.4, pp.1942-1948.
[26].Bai. X & Wei.H, *Semi –definite programming – based method for security constrained unit commitment with operational and optimal power flow constraints*, IET Genrn, Trans, and Distr. 2009, Vol3 (2), pp 182-197.
[27].C.Christopher Columbus, Sishaj P. Simon, *Parallel hybrid enhanced inherited GA based SCUC in a distributed cluster*, Artificial Intelligence research, Sep 2012, Vol1. No.1 ,pp96-106.

APPENDIX: 1
LOAD DATA FOR ALL TEST SYSTEMS

Hour	1	2	3	4	5	6	7	8	9	10	11	12
Load (MW)	700	750	850	950	1000	1100	1150	1200	1300	1400	1450	1500
Hour	13	14	15	16	17	18	19	20	21	22	23	24
Load (MW)	1400	1300	1200	1050	1000	1100	1200	1400	1300	1100	900	800

APPENDIX:2
10 UNIT SYSTEM DATA

	Pmax	Pmin	A	B	C	MUi	MDi	Hcost	Ccost	Chour	IniState
Unit1	455	150	0.00048	16.19	1000	8	8	4500	9000	5	8
Unit2	455	150	0.00031	17.26	970	8	8	5000	10000	5	8
Unit3	130	20	0.002	16.60	700	5	5	550	1100	4	-5
Unit4	130	20	0.00211	16.5	680	5	5	560	1120	4	-5
Unit5	162	25	0.00398	19.70	450	6	6	900	1800	4	-6
Unit6	80	20	0.00712	22.26	370	3	3	170	340	2	-3
Unit7	85	25	0.00079	27.74	480	3	3	260	520	2	-3
Unit8	55	10	0.00413	25.92	660	1	1	30	60	0	-1
Unit9	55	10	0.00222	27.27	665	1	1	30	60	0	-1
Unit10	55	10	0.00173	27.79	670	1	1	30	60	0	-1