2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A heuristic to generate initial feasible solutions for the Unit Commitment Problem

Yi Sun
Department of Electrical and Electronic Engineering
The University of Hong Kong
Email: sunyimik@hku.hk

Albert Y.S. Lam
Department of Computer Science
Hong Kong Baptst University
Email: albertlam@ieee.org

Victor O.K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Email: vli@eee.hku.hk

***Abstract*—This paper presents a heuristic approach to generate initial feasible solutions for the Unit Commitment (UC) problem in electric power generation. The Chemical Reaction Optimization (CRO) algorithm is implemented to solve this problem. Multiple generator constraints and system constraints are considered. We also program the binary PSO and the Elite PSO (EPSO) for comparison. The proposed heuristic approach is combined with the three optimization algorithms to form H-CRO, H-PSO and H-EPSO. We test the performance of all algorithms on the standard 10-unit system. Simulation results show that the heuristic can improve the performance and CRO provides better convergence than the two PSO algorithms. H-CRO is also tested on a 20-unit and 100-unit system to show its capability. The results provided in this paper suggest that the proposed heuristic approach is a better alternative for solving the UC problem. CRO also has its advantage in optimizing UC problems.**

***Index Terms*—Chemical reaction optimization, heuristic, unit commitment, power grid.**

I. Introduction

Power supply and demand must be balanced in a power system. However, the demand varies. For example, the demand during the daytime is likely to be higher than that at midnight. Similarly, compared to Saturdays and Sundays, weekdays tend to require much more power [1]. Hence the power generating units are required to schedule their power outputs according to the demand.

Given one particular demand curve, there are many power generation strategies to accommodate the system constraints while satisfying the power demand. Of course it is preferable to select the strategy with the minimum cost. Unit commitment (UC) is the problem of finding the least cost schedule to satisfy the power balance. This scheduling problem can be formulated as an optimization problem with system operating constraints [2].

In the past few decades, the optimal UC problem has been addressed with different approaches, ranging from dedicated heuristics to general-purpose optimization algorithms.

The very early methods applied to solve the UC problem including Exhaustive Enumeration [3], Priority List [4] and Dynamic programming [5]. However these approaches are not satisfactory with the increasing demand on higher accuracy and shorter computational time.

Evolutionary approaches have shown their capabilities in dealing with complex problems. Genetic Algorithm (GA) is one such approach applied to the UC problem. It was tested on the single day as well as the one-week UC problem and on small and large size systems [6] [7]. Some other GA-based approaches have also been developed to provide better results [8] [9] [10]. However the main drawback of applying GA to today's real world problems is its long computational time. Other popular evolutionary approaches have also been applied to solve the UC problem, such as Evolutionary Programming (EP)[11] and Particle Swarm Optimization (PSO) [12] [13] [14]. The performance in accuracy and efficiency have all been improved compared to the early GA approaches.

Chemical Reaction Optimization (CRO) is a recently developed evolutionary algorithm [15]. CRO is inspired by the molecular behaviors in a chemical reaction to search for the most stable state with the minimum free energy. The correspondence between the free energy of a molecule and the objective function value of a solution makes the similarity between chemical reactions and optimization apparent. With the molecules representing the solutions, CRO performs optimization by mimicking the behaviors of molecules in a chemical reaction. CRO has shown its capability in solving optimization problems [16] [17].

By studying the previous efforts in solving the UC problem, three common steps can be identified: finding feasible schedules, allocating power output to calculate the total cost, and performing optimization to search for the best solution. By the first two steps, feasible solutions can be found. The third step searches for the best feasible solution. Depending on the optimization algorithm used, the first two steps may be repeated in the optimization process. For the second step, also known as Economic Dispatch (ED), many fast and accurate methods have been developed [18]. The third step can be solved by evolutionary algorithms.

However, the first step, also known as the unit schedule initialization, is mostly done by random generation and trial-and-error. Random search for a feasible schedule is not efficient, especially when dealing with large scale systems. Very few efforts focus on initialization. There are some heuristic algorithms aiming at changing a randomly generated schedule into a feasible one, thus improving the performance of the overall optimization results [19]. To further improve the performance of heuristic initialization, instead of changing a randomly gen-

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

TABLE I
NOTATIONS

Given variables	Description
N	Number of units
i	Index of unit
T	Number of time intervals
t	Index of time intervals
P_{imax}, P_{imin}	Max and min output power
MUT_i, MDT_i	Min up and down time
SUC_{ihot}, SUC_{icold}	Hot and cold start costs
SDC_i	Shut down cost
R_{up_i}, R_{down_i}	Ramp up and down rates
a_i, b_i, c_i	Economic coefficients
T_{icold}	Cold start hours
$Preduration_i$	Status of Unit i from last UC planning horizon
	positive for already online time
	negative for already offline time
Control variables	**Description**
U_i^t	Operation status of Unit i at time t
	0 for offline, 1 for online
Dependent variables	**Description**
P_i^t	Output power of Unit i at time t
Γ_i^t	Duration of Unit i being online/offline before time t
	Positive for online time
	Negative for offline time

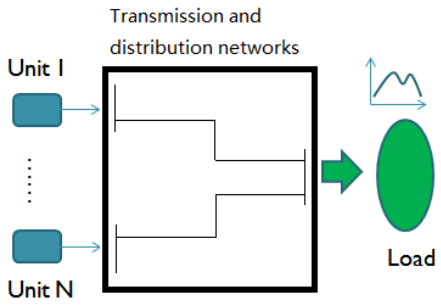

Fig. 1. Sample UC system

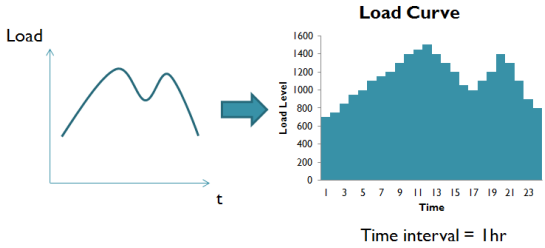

Fig. 2. 24-hour load curve

erated solution to make it feasible, one can directly construct a feasible solution. In this paper, we propose a constructive heuristic utilizing the charactistics of the UC problem to determine the initial population of feasible solutions. With this domain knowledge-based heuristic intialization, feasible unit status schedules can be constructed without resorting to trial-and-error. This constructive approach has been tested on the 10-unit, 20-unit and 100-unit systems. The rest of the paper is organized as follows. We introduce the UC problem formulation in Section II. Then the detailed design of the proposed algorithm is explained in Section III. The CRO and PSO algorithms used in our simulation are introduced in Section IV. The simulation results are stated and compared in Section V. Finally we conclude the paper in Section VI.

II. Problem Formulation

The UC problem aims to find an operation plan for generating units to achieve the least total cost without violating the unit operating constraints. As power systems expand, more accurate and faster UC algorithms are required.

We consider a power system with N generation units and a given load curve D divided into T time intervals. Let i and t be the index of the units and time intervals, respectively. We define all the variables in a UC problem in Table I. To give a better understanding on Γ_i^t, we use the following examples. $\Gamma_2^5 = 3$ means that Unit 2 has been online for three time intervals consecutively till the 5th time interval. Similarly $\Gamma_2^5 = -3$ means Unit 2 is said to be offline for three time intervals consecutively till the 5th time interval.

To better illustrate the relationship between the units and the load, an example of the system is given in Fig 1. Power is generated by the generating units, and transmitted through the transmission and distribution networks to the load. We aim to balance the power demand at the load side with the power supply from the generating units. The structures and constraints of the transmission and distribution networks are not considered.

The load curve is estimated from past experience. In the system model, the load curve is divided into T time intervals. The average value of each interval is used to represent a constant load level for the whole interval to simplify the calculation. An example of the load curve model is displayed in Fig 2, where the planning horizon is one day. One time interval is set to be one hour and there are two peaks in the curve, one at the 12th hour and the other at the 20th hour.

The units can be categorized into many types according to their power sources. In our paper, the units are all assumed to be the conventional type which can provide stable and fully controllable power output. There are three kinds of cost for Unit i: generation cost $F(P_i)$, start-up cost SUC_i, shut-down cost SDC_i.

The generation cost $F(P_i)$ of the ith unit is defined as:

$$F(P_i) = a + b \times P_i + c \times P_i^2 \tag{1}$$

where P_i is the output power of Unit i, and a, b and c are economic coefficients of Unit i.

The start-up cost is a function of the unit offline time. The start-up cost of unit i is defined as a piecewise function as follows [7]:

$$SUC_i(\Gamma_i) = \begin{cases} SUC_{ihot}, & if\ MDT_i < \Gamma_i < MDT_i + T_{icold} \\ SUC_{icold}, & if\ \Gamma_i \geq MDT_i + T_{icold} \end{cases} \tag{2}$$

where T_{icold} is the cold start hour of Unit i. The shut-down costs of the conventional units are normally neglected. Therefore we do not include the term of this cost in our unit models[7].

The objective is to minimize the sum of the generation cost and the start-up cost [2], i.e.,

$$\text{minimize} \sum_{t=1}^{T}\sum_{i=1}^{N} P_i^t U_i^t + SUC_i(1 - U_i^{t-1})U_i^t \tag{3}$$

The constraints include:

1. Power balance constraints. In each time interval, the sum of the supply power should balance the demand:

$$\sum_{i=1}^{N} P_i^t U_i^t = D^t,\ t = 1, 2, ..., T \tag{4}$$

2. Output power limits on Unit i at time t. The power output of every unit should be within its upper and lower bounds, i.e.,

$$P_{i\min} < P_i^t < P_{i\max}. \tag{5}$$

3. Spin reserve requirement at time t. For safety considerations, there should be some spare power to cover possible unexpected extra demand. The sum of the maximum output power of every online unit should be larger than the total demand plus the spin reserve (SR) which is normally set to 10% of the demand, i.e.,

$$\sum_{i=1}^{N} P_{i\max} U_i^t > D^t + SR \tag{6}$$

4. Minimum up and down time requirements for Unit i. Once a unit i is online, it has to be kept online for MUT_i time intervals. Similarly once a unit i is offline, it has to be offline for MDT_i time intervals.

$$\Gamma_i^t \geq MUT_i, if\ \Gamma_i^t > 0\ and\ U_i^t = 0 \tag{7}$$

$$-\Gamma_i^t \geq MDT_i, if\ \Gamma_i^t < 0\ and\ U_i^t = 1 \tag{8}$$

5. Ramp rate constraints for Unit i. The change rate of the output power for Unit i between two consecutive time intervals is limited by

$$P_i^{t-1} - Rdown_i \leq P_i^t \leq P_i^{t-1} + Rup_i \tag{9}$$

Hence the UC problem is defined as follows:

$$\text{minimize} \sum_{t=1}^{T}\sum_{i=1}^{N} F(P_i^t, U_i^t, SUC_i^t) \tag{10}$$

$$= \sum_{t=1}^{T}\sum_{i=1}^{N} P_i^t U_i^t + SUC_i(1 - U_i^{t-1})U_i^t \tag{11}$$

subject to (2), (4), (5), (6), (7), (8), and (9)

III. Algorithm Design

A. Background

Solving the UC problem mainly includes three steps: finding feasible unit status schedules $U = [U_i^t, i = 1, .., N, t = 1, ..., T]$, performing Economic Dispatch (ED), and searching for the optimized solution among the feasible ones. There are basically two ways to represent a feasible status schedule. One is to use an integer number sequence to represent the schedule for every unit. For example, sequence $[3, -2, 4]$ means that the unit is online for 3 time intervals then offline for 2 intervals and again online for 4 intervals. The other way is using an $N \times T$ binary matrix to represent the schedule for the whole system, where 0 means the unit is offline at that time interval and 1 otherwise. Our proposed heuristic initialization algorithm is based on the second approach.

First we give a detailed explanation about how our algorithm works and why it can lead to a feasible solution in an efficient manner. It is followed by the design of the CRO algorithm, focusing on the operator designs. Detailed information for CRO can found in [15].

B. Initialization Heuristic Design

The proposed heuristic basically produces in five steps a feasible unit status matrix scheduling solution. The UC problem knowledge and the corresponding empirical rules are applied to design the heuristic. We first define the $Unit\ Check\ Sequence\ (UCS)$ as

$$UCS = [u_1, u_2, ..., u_N] \tag{12}$$

Each integer u_k, where $k = 1, 2, ..., N$, represents the u_kth unit. UCS is used to help construct the initial solutions. The detailed steps are given as follows:

Step 1) Initialization of the status matrix. We set all components in the status matrix to zero, i.e. $U_{u_k}^t = 0, k = 1, 2, ..., N, t = 1, 2, ..., T$.

Step 2) Peak load time preset. Every demand curve has peak load periods. A peak load always requires the most online units to satisfy the reserve constraint. Hence we satisfy the reserve constraints at peak load times first. Suppose there are two peaks and they occur at times t_1 and t_2. At $t = t_1$, we start with $k = 1$ and check if the reserve constraint for time $t = t_1$ is satisfied:

$$\sum_{i=1}^{N} (P_{max_{u_k}} \times U_{u_k}^{t_1}) \geq D^{t_1} + SR$$

If not, set Unit u_1 to online, i.e. $U_{u_1}^{t_1} = 1$, then update $k = k+1$. Repeat the reserve constraint check until $k = N$. During

TABLE II
EXAMPLE DEMAND

Time	1	2	3	4	5	6
Demand	800	600	700	750	1000	800

this process, if the reserve constraint is satisfied or $k = N$, we continue to commit the same units in the time intervals surrounding t_1 to satisfy the MUT constraint. Suppose t_r is a random integer and let $t_r \in [0, MUT_{u_k}]$. If $U_{u_k}^{t_1} = 1$, according to the minimum up time constraint, we set all $U_{u_k}^{t_j} = 1, t_j \in [t_1 - t_{random}, t_1 - t_{random} + MUT_{u_k}]$. Then we change the time interval from $t = t_1$ to $t = t_2$ where t_2 is the next peak load time. Repeat this until all peak load time intervals satisfy the reserve and the minimum up time constraint.

Step 3) First time interval preset. Since every unit is already online or offline for several time intervals in the last UC planning horizon, we check the status of every unit i from the last UC planning horizon, $Preduration_i$ to see if any MUT/MDT constraints are not satisfied at the first time interval. That is, for any Unit i, if $0 < Preduration_i < MUT_i$, then commit $U_i^t = 1$ for $t = 1, ..., MUT_i - Preduration_i$. Else if $-MDT_i < Preduration_i < 0$, then set $U_i^t = 0$ for $t = 1, ..., MDT_i - Preduration_i$.

Step 4) Overall schedule check. Repeat Step 2 from $t = 1$ to $t = T$. If any U_i^t is already set online or offline by the previous steps, it will be kept unchanged. Otherwise, U_i^t will be set according to the rule used in Step 2.

Step 5) MDT check. Since we separately commit the peak and other time intervals, the statuses in between may violate the MDT constraints. For each Unit i at time interval t, if $U_i^t = 1, U_i^{t+1} = 0$, check if any U_i^{t+2} to $U_i^{t+MDT_i}$ is already set to 1 previously. If so, suppose $U_i^k = 1$ where $t + 1 < k < t + MDT_i$ then all U_i^{t+1} to U_i^k is set to 1 and set $t = k$. Otherwise, set $t = t + 1$ and repeat this until $t = T$.

After these 5 steps, a feasible status matrix is determined. With different UCS, we can generate different matrices. Hence the UCS can be used as the manipulative agents (main control factors) when we apply optimization algorithms to find the final solution for the UC problem. For better illustration, a flowchart of the initialization heuristic is displayed in Fig 3.

An example with 4 units and 6 time intervals is provided as follows for illustration. Demand is given in Table II. The output power limits for every unit and the MUT/MDT is given in Table III.

For this example:

$$UCS = \{1, 2, 4, 3\}$$

The steps are illustrated in Fig 4.

C. Economic Dispatch (ED)

Once the feasible unit status schedule is found by the heuristic, we use Lambda Iteration to perform the ED allocating the output power for every online unit.

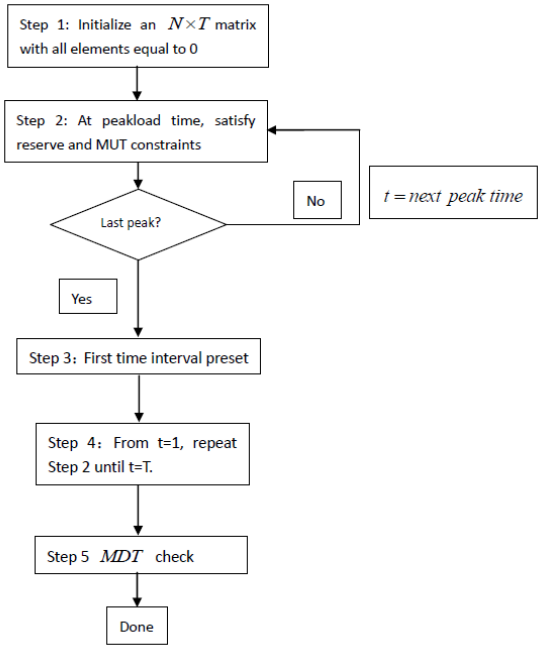

Fig. 3. Initialization heuristic algorithm

TABLE III
EXAMPLE UNIT LIMITS

Unit	1	2	3	4
P_{max}	550	250	150	150
P_{min}	200	100	50	50
MUT	4	3	1	2
MDT	4	3	1	2
$Preduration$	3	-3	-1	-2

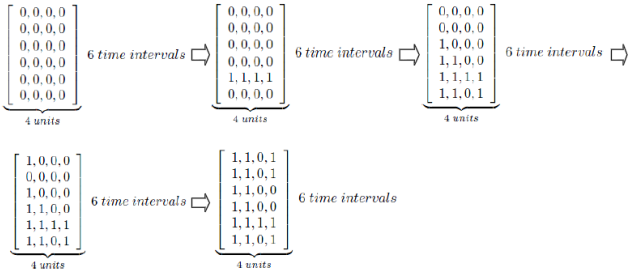

Fig. 4. Solution matrix update of the example

In every time interval t, the generation cost is given by $F(P_i^t) = \sum_{i=1}^{N} (a_i + b_i \times P_i^t + c_i \times P_i^t)$. Suppose

$$\lambda^t = \frac{\partial F}{\partial P} = b_i + 2c_i P_i^t \tag{13}$$

Hence P_i^t can be calculated by

$$P_i^t = \frac{\lambda^t - b_i}{2c_i} \tag{14}$$

To find the proper λ^t, we start from a λ^t_{low} and a λ^t_{high} and run the iteration to find the result.

In every iteration, if $\left|\lambda^t_{high} - \lambda^t_{low}\right| > \varepsilon$, where ε is a preset tolerance value, then λ^t_{mean} is calculated by

$$\lambda^t_{mean} = \frac{\lambda^t_{low} + \lambda^t_{high}}{2} \tag{15}$$

The output power for all units in time interval t is calculated by

$$P^t_i = \frac{\lambda^t_{mean} - b_i}{2c_i} \tag{16}$$

If the calculated $P^t_i > P_{imax}$ or $P^t_i < P_{imin}$, P^t_i is set to the corresponding boundary value. Then λ^t_{high} or λ^t_{min} is updated according to the difference between the sum of output power and the demand $\sum_{i=1}^{N}(P^t_i U^t_i) - D^t$.

$$\begin{aligned} &If \sum_{i=1}^{N}(P^t_i U^t_i) - D^t > 0 \\ &\lambda^t_{high} = \lambda^t_{mean} \\ &If \sum_{i=1}^{N}(P^t_i U^t_i) - D^t < 0 \\ &\lambda^t_{low} = \lambda^t_{mean} \end{aligned}$$

Repeat the iteration until $\left|\lambda^t_{high} - \lambda^t_{low}\right| \le \varepsilon$ and calculate the final λ^t_{mean}. The output power for all units in time interval t is determined by (16).

IV. Optimization Algorithms

In this section, a brief introduction of the Chemical Reaction Optimization and the Particle Swarm Optimization is given.

A. Chemical Reaction Optimization

1) Manipulated Agents: The basic manipulated agents of CRO are molecules. For discrete problems such as UC, there are three major operands for each molecule: the molecular structure (ω), potential energy (*PE*, also known as the objective function value of the molecule) and kinetic energy (*KE*). ω contains the control variables for a feasible solution.[1] The control variable in a molecular structure ω is UCS.

Since $\sum_{t=1}^{T}\sum_{i=1}^{N} F(P^t_i, U^t_i, SUC^t_i)$ is the objective function value of the UC problem, the *PE* of the molecule ω is defined as follows:

$$PE_\omega = \sum_{t=1}^{T}\sum_{i=1}^{N} F(P^t_i, U^t_i, SUC^t_i) = F(\omega) \tag{17}$$

where P^t_i, U^t_i are the components in the solution matrix obtained from the UCS in ω, SUC^t_i is the start up cost of Unit i.

KE is defined as the tolerance to generate a molecule with a worse structure (i.e. a solution with a higher objective function value).

[1]Theoretically, a candidate solution should be composed of both the control and the dependent variables. Since the latter can be calculated from the former in the process of solving the UC problem, the dependent variables are removed from ω for simplicity.

2) Elementary Reactions: There are four kinds of elementary reactions defined in CRO. We use examples to illustrate the four reactions as follows:

On-wall Ineffective Collision

An on-wall ineffective collision is a collision of a molecule with the wall of the container. It causes a small change in the molecular structure. An example of the on-wall collision where the molecular structure ω corresponds to 6 units, that is, $N = 6$, is given as:

$$\underbrace{[1, \underline{2}, \underline{4}, 3, 5, 6]}_{\omega} \rightarrow \underbrace{[1, \underline{4}, \underline{2}, 3, 5, 6]}_{\omega'}$$

Decomposition

A decomposition of one molecule results in two new molecules. It will bring a dramatic change and the resultant molecules will have significantly different structures and energies from the original one. The decomposition is illustrated by the following example:

$$\underbrace{[1, 2, \underline{4}, \underline{3, 5, 6}]}_{\omega} \rightarrow \underbrace{[3, 5, 6, 4, 2, 1]}_{\omega_1} + \underbrace{[6, 5, 3, 1, 2, 4]}_{\omega_2}$$

Inter-molecular Ineffective Collision

An inter-molecular ineffective collision is caused by two molecules hitting each other and bouncing back.

The following 6-unit example is given for illustration:

$$\underbrace{[1, \underline{2}, 4, \underline{3}, 5, 6]}_{\omega_1} \rightarrow \underbrace{[1, \underline{3}, 4, \underline{2}, 5, 6]}_{\omega'_1}, \underbrace{[\underline{1}, 2, 4, 3, \underline{5}, 6]}_{\omega_2} \rightarrow \underbrace{[\underline{5}, 2, 4, 3, \underline{1}, 6]}_{\omega'_2}$$

Synthesis

A synthesis of two molecules results in one combined molecule. The change is vigorous.

An example is given:

$$\underbrace{[1, 2, \underline{4}, 3, \underline{5}, 6]}_{\omega_1} + \underbrace{[5, 1, \underline{4}, 6, \underline{5}, 2]}_{\omega_2} \rightarrow \underbrace{[1, 3, \underline{4}, 2, \underline{5}, 6]}_{\omega'}$$

3) The Overall Algorithm: The overall algorithm includes three stages: initialization, iterations, and the final stage.

In initialization, we set the values of the CRO parameters, i.e., *PopSize*, *InitialKE*, *MoleColl* and *KELossRate*. We create *PopSize* number of molecules with randomly generated *UCS* with *KE* equal to *InitialKE* and *PE* calculated from the *UCS*. We go through a number of iterations until a preset number of function evaluations (FEs) is reached.

At the end of each iteration, any newly found better objective function value will be recorded. When the maximum FEs have been reached, the best overall result will be output. More details of implementing CRO can be found in [20].

B. Particle Swarm Optimization

1) Manipulated Agents: The basic manipulated agents of PSO are particles. Considering the UC problem with binary bit representing the unit on/off state, we use the binary version of PSO instead the original one. The control variables are formed as a binary matrix.

Let p stands for an individual particle at ith iteration, suppose the problem is a D-dimension problem, there are three vectors composed in each p: the position of the particle $X_p^i = (x_{p1}^i, x_{p2}^i, ..., x_{pD}^i)$, the best position found by this individual particle $B_p = (b_{p1}, b_{p2}, ..., b_{pD})$ and its velocity $V_p^i = (v_{p1}^i, v_{p2}^i, ..., v_{pD}^i)$. In the UC problem, $D = N \times T$ where N is the number of units and T is the number of time intervals. In a new iteration $i+1$, the position and velocity vectors are updated as follows:

$$v_{pd}^{i+1} = v_{pd}^i + c_1(b_{pd} - x_{pd}^i) + c_2(b_{gd} - x_{pd}^i)$$
$$x_{pd}^{i+1} = x_{pd}^i + v_{pd}^i$$
$$d = 1, 2, ..., D$$

where g_d is the dth variable of the global best solution ever found in the optimization process.

However in terms of binary variables, x_{pd}^i and b_{pd} are both integers in $[0, 1]$ and v_{pd}^i should also be constrained to the interval $[0.0, 1.0]$ representing the possibility that the value of x_{pd}^i is changed in the next iteration. The updating equation is defined as follow:

$$if\ (rand() < G(v_{pd}^i))\ then\ x_{pd}^{i+1} = 1$$
$$else\ x_{pd}^{i+1} = 0$$

Where $G(v_{pd}^i)$ is a function to restrict the velocity in the interval $[0.0, 1.0]$. A sigmoid limiting function is selected to do this restriction. The $rand()$ is a random number selected from a uniform distribution in $[0.0, 1.0]$.

For detailed instructions about the binary PSO, readers are suggested to refer to [21].

V. Simulation Results

The simulation is implemented in the Visual studio 2012 environment executed on a 2.4 GHz Intel i5-2430M dual core personal computer with 4GB RAM. To demonstrate the effectiveness of the proposed heuristic algorithm, 10-unit, 20-unit and 100-unit power system models are tested using CRO and CRO with heuristic (H-CRO). A standard binary PSO version and a EPSO version are programmed based on [21] and [19]. The proposed initialization heuristic is combined with these two PSOs to form PSO with heuristic (H-PSO) and EPSO with heuristic (H-EPSO). All the above six algorithms are tested on the three systems to show the capability of the heuristic initialization algorithm.

The CRO and PSO parameters are set as shown in Table IV, Table V and Table VI. E_p and E_g are the number of elite groups of the best positions for each particle and the swarm. c_1, c_2, $c_{p,1}$, $c_{p,2}$, $c_{g,1}$, $c_{g,2}$ are the acceleration coefficients.

For the 10-unit system, the parameters for the CRO and the H-CRO are the same. Since the UCS is used as the control variables, with an increasing number of units, the number of possible choices of the UCS increases dramatically. Hence we apply larger $MoleColl$ for bigger system so that the search is more efficient. The increasing $InitialKE$ is also due to the larger objective function value for the bigger system.

TABLE IV
CRO PARAMETER SETTINGS

Variables	10-unit	20-unit	100-unit
PopSize	3	3	3
InitialKE	100	10000	100000
KELossRate	0.5	0.5	0.5
MoleColl	0.001	0.1	0.5

TABLE V
PSO PARAMETER SETTINGS

Variables	10-unit	20-unit	100-unit
Numofparticles	20	20	20
Maxgen	500	2000	5000
w	Rand[0,1]	Rand[0,1]	Rand[0,1]
c_1 *and* c_2	c_1=2.8 c_2=1.2	c_1=2.8 c_2=1.2	c_1=2.8 c_2=1.2

TABLE VI
EPSO PARAMETER SETTINGS

Variables	10-unit	20-unit	100-unit
Numofparticles	20	20	20
Maxgen	500	2000	5000
Ep	1	1	2
Eg	2	2	2
c_p *and* c_g	c_p=2.0 $c_{g,1}$=1.2 $c_{g,2} = 0.8$	c_p=2.0 $c_{g,1}$=1.2 $c_{g,2} = 0.8$	$c_{p,1}$=1.2 $c_{p,2}$=0.8 $c_{g,1}$=1.2 $c_{g,2} = 0.8$

In the 10-unit system, the maximum, minimum power output, minimum up and down times of every unit, the status from the last UC planning horizon (Preduration, Pre) and the economic coefficients are given in Table VII and the demand curve of one day is given in Table VIII. The 20-unit and 100-unit system are the combination of 2 and 10 10-unit systems.

The detailed comparison is shown in Table IX. The run time and Function Evaluation (FE) used per run is shown in the Table IX. The best, worst, mean results are recorded.

The big differences occured in PSO and EPSO worst and mean results for 20-unit and 100-unit are due to the convergence of the two algorithms. There is a possibility that PSO and EPSO have not get converged within the designed FEs.

Next we compare the rates of convergence. In Fig 5 and Fig 6, the test system is the 10-unit system and we run up to 5000 FEs. It can be concluded that when the initialization heuristic is not used, CRO gains a faster convergence compared to PSO and EPSO. When the heuristic is used, H-CRO also converges faster than H-PSO and H-EPSO.

In Fig 7, Fig 8 and Fig 9, the major goal is to stress the

TABLE VII
UNIT PARAMETERS

Unit	P_{max}	P_{min}	MUT	MDT	SUC_{hot}	SUC_{cold}	Pre	T_{icold}	a	b	c
1	455	150	8	8	4500	9000	8	5	1000	16.19	0.00048
2	455	150	8	8	5000	10000	8	5	970	17.26	0.00031
3	130	20	5	5	550	1100	-5	4	700	16.6	0.002
4	130	20	5	5	550	1100	-5	4	680	16.5	0.00211
5	162	25	6	6	900	1800	-6	4	450	19.7	0.00398
6	80	20	3	3	170	340	-3	2	370	22.26	0.00712
7	85	25	3	3	260	520	-3	2	480	27.74	0.00079
8	55	10	1	1	30	60	-1	0	660	25.92	0.00413
9	55	10	1	1	30	60	-1	0	665	27.27	0.00222
10	55	10	1	1	30	60	-1	0	670	27.79	0.00173

TABLE VIII
DEMAND

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

TABLE IX
COMPARISONS AMONG CROS AND PSOS

System	Method	Best result ($)	Worst result ($)	Mean result ($)	FE per run	run
10-unit	CRO	596257	607011	602221	5000	50
	H-CRO	564748	565554	564941	5000	50
	PSO	582567	2588100	879988	10000	50
	H-PSO	582943	604201	592137	10000	50
	EPSO	585954	1624190	713682	10000	50
	H-EPSO	579135	597106	583172	10000	50
20-unit	CRO	1187890	1197800	1194200	10000	50
	H-CRO	1117960	1132020	1125710	10000	50
	PSO	1209950	4256330	2063720	20000	50
	H-PSO	1220000	1240000	1233000	20000	50
	EPSO	1200070	3267260	1954750	20000	50
	H-EPSO	1114000	1220000	1170000	20000	50
100-unit	CRO	6000000	6130000	6040000	20000	20
	H-CRO	5593150	5708840	5656900	20000	20
	PSO	-	-	-	100000	20
	H-PSO	6168000	7185000	6187000	100000	20
	EPSO	-	-	-	100000	20
	H-EPSO	6080000	7120000	6520000	100000	20

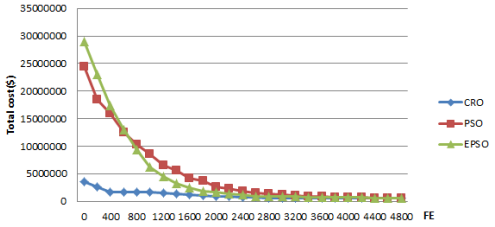

Fig. 5. Convergence without our proposed heuristic in 10-unit system

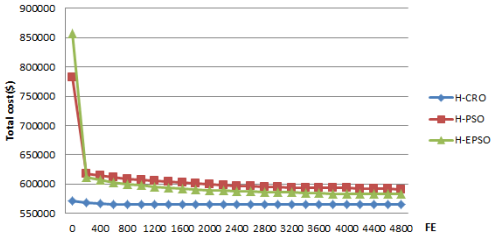

Fig. 6. Convergence with our proposed heuristic in 10-unit system

merits of the proposed heuristic initialization algorithm. In terms of the PSO and EPSO cases, the application of the heuristic leads to significant convergence rate improvements. In the CRO case, the heuristic also returns a better optimization result.

VI. CONCLUSION

The UC problem schedules the power generating units in order to minimize the cost and is one of the basic problems in power system engineering. There are many algorithms aiming at improving the optimization process of the UC problem. However the feasible solution initialization part is somehow ignored.

In this paper, we propose a constructive method to find feasible initial UC solutions. In previous work, a feasible solution is always obtained by changing a randomly generated

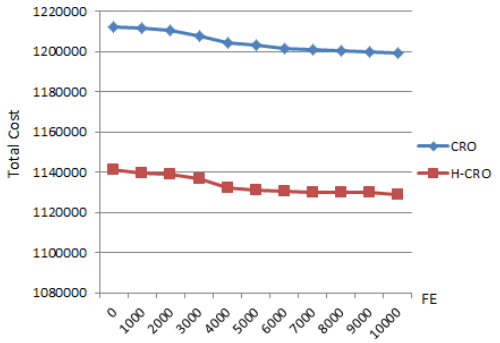

Fig. 7. Convergence comparision of CROs in 20-unit system

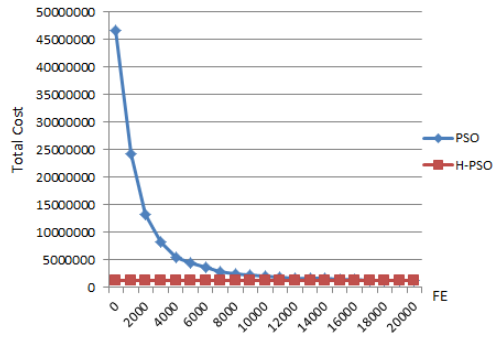

Fig. 8. Convergence comparision of PSOs in 20-unit system

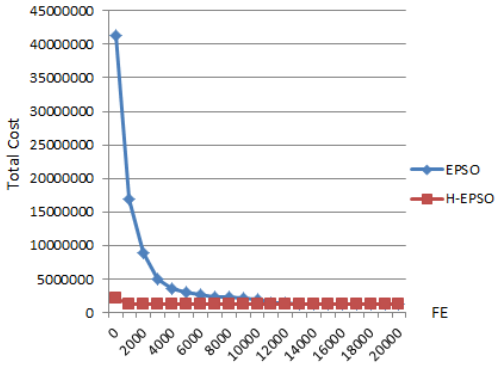

Fig. 9. Convergence comparision of EPSOs in 20-unit system

solution to make it feasible, which is very inefficient.

When comparing CRO, H-CRO, PSO, H-PSO, EPSO and H-EPSO, the simulation results show that with the proposed initialization scheme, both the overall results as well as the convergence rates are improved.

In the future, we plan to study larger systems to see the capability of the constructive initialization algorithm. The Security Constrained Unit Commitment (SCUC) is another problem we plan to study. By using our algorithm, we hope we can construct a faster and more efficient solution. Another direction is to generalize the initialization analysis. We want to find some principles in determining the initialization algorithms that will work best for different types of problems.

ACKNOWLEDGMENT

A.Y.S. Lam is supported in part by the Faculty Research Grant of Hong Kong Baptist University, under Grant No. FRG2/13-14/045. Y. Sun and V.O.K. Li are supported in part by the Collaborative Research Fund of the Research Grants Council of Hong Kong under Grant No. HKU10/CRF/10.

REFERENCES

[1] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control*, 2nd ed. New York, NY: John Wiley & Sons, 1996.

[2] N. Padhy, "Unit commitment-a bibliographical survey," *Power Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 1196 – 1205, may 2004.

[3] R. Kerr, J. Scheidt, A. Fontanna, and J. Wiley, "Unit commitment," *Power Apparatus and Systems, IEEE Transactions on*, vol. PAS-85, no. 5, pp. 417 –421, may 1966.

[4] F. Lee, "Short-term thermal unit commitment-a new method," *Power Systems, IEEE Transactions on*, vol. 3, no. 2, pp. 421 –428, may 1988.

[5] P. G. Lowery, "Generating unit commitment by dynamic programming," *Power Apparatus and Systems, IEEE Transactions on*, vol. PAS-85, no. 5, pp. 422–426, 1966.

[6] D. Dasgupta and D. McGregor, "Thermal unit commitment using genetic algorithms," *Generation, Transmission and Distribution, IEE Proceedings-*, vol. 141, no. 5, pp. 459 –465, sep 1994.

[7] S. Kazarlis, A. Bakirtzis, and V. Petridis, "A genetic algorithm solution to the unit commitment problem," *Power Systems, IEEE Transactions on*, vol. 11, no. 1, pp. 83 –92, feb 1996.

[8] K. Swarup and S. Yamashiro, "Unit commitment solution methodology using genetic algorithm," *Power Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 87 –91, feb 2002.

[9] C.-P. Cheng, C.-W. Liu, and C.-C. Liu, "Unit commitment by lagrangian relaxation and genetic algorithms," *Power Systems, IEEE Transactions on*, vol. 15, no. 2, pp. 707–714, 2000.

[10] I. Damousis, A. Bakirtzis, and P. Dokopoulos, "A solution to the unit-commitment problem using integer-coded genetic algorithm," *Power Systems, IEEE Transactions on*, vol. 19, no. 2, pp. 1165–1172, 2004.

[11] K. A. Juste, H. Kita, E. Tanaka, and J. Hasegawa, "An evolutionary programming solution to the unit commitment problem," *Power Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 1452–1459, 1999.

[12] Z.-L. Gaing, "Discrete particle swarm optimization algorithm for unit commitment," in *Power Engineering Society General Meeting, 2003, IEEE*, vol. 1, july 2003, p. 4 vol. 2666.

[13] ——, "Discrete particle swarm optimization algorithm for unit commitment," in *Power Engineering Society General Meeting, 2003, IEEE*, vol. 1, 2003, pp. –424 Vol. 1.

[14] T. Ting, M. V. C. Rao, and C. Loo, "A novel approach for unit commitment problem via an effective hybrid particle swarm optimization," *Power Systems, IEEE Transactions on*, vol. 21, no. 1, pp. 411–418, 2006.

[15] A. Y. S. Lam and V. O. K. Li, "Chemical-reaction-inspired metaheuristic for optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 381 –399, Jun. 2010.

[16] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li, "Evolutionary artificial neural network based on chemical reaction optimization," in *Proc. IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, LA, USA, Jun. 2011, pp. 2083–2090.

[17] Y. Sun, A. Y. S. Lam, V. O. K. Li, J. Xu, and J. J. Q. Yu, "Chemical reaction optimization for the optimal power flow problem," in *Proceedings of IEEE Congress on Evolutionary Computation (IEEE CEC 2012)*, Brisbane, Australia, Jun. 2012, pp. 1–8.

[18] W.-M. Lin, F.-S. Cheng, and M.-T. Tsay, "An improved tabu search for economic dispatch with multiple minima," *Power Systems, IEEE Transactions on*, vol. 17, no. 1, pp. 108 –112, feb 2002.

[19] P.-H. Chen, "Two-level hierarchical approach to unit commitment using expert system and elite pso," *Power Systems, IEEE Transactions on*, vol. 27, no. 2, pp. 780 –789, may 2012.

[20] A. Y. S. Lam and V. O. K. Li, "Chemical reaction optimization: A tutorial (invited paper)," *Memetic Computing*, vol. 4, no. 1, pp. 3–17, 2012. [Online]. Available: http://www.springerlink.com/content/y3774v13p68g6047/

[21] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, 1997, pp. 4104–4108 vol.5.