



ELSEVIER

Computers and Chemistry 25 (2001) 251–259

**Computers  
& Chemistry**

www.elsevier.com/locate/compchem

# Modeling and prediction for discharge lifetime of battery systems using hybrid evolutionary algorithms

Hongqing Cao <sup>a,\*</sup>, Jingxian Yu <sup>b</sup>, Lishan Kang <sup>a</sup>, Hanxi Yang <sup>b</sup>, Xinping Ai <sup>b</sup><sup>a</sup> State Key Laboratory of Software Engineering, Wuhan University, Wuhan, 430072, People's Republic of China<sup>b</sup> Institute of Electrochemistry, Department of Chemistry, Wuhan University, Wuhan, 430072, People's Republic of China

Received 15 March 2000; received in revised form 26 June 2000; accepted 14 September 2000

## Abstract

A hybrid evolutionary modeling algorithm (HEMA) is proposed to build the discharge lifetime models with multiple impact factors for battery systems as well as make predictions. The main idea of the HEMA is to embed a genetic algorithm (GA) into genetic programming (GP), where GP is employed to optimize the structure of a model, while a GA is employed to optimize its parameters. The experimental results on lithium-ion batteries show that the HEMA works effectively, automatically and quickly in modeling the discharge lifetime of battery systems. The algorithm has some advantages compared with most existing modeling methods and can be applied widely to solving the automatic modeling problems in many fields. © 2001 Elsevier Science Ltd. All rights reserved.

**Keywords:** Discharge lifetime of battery systems; Lithium-ion battery; Hybrid evolutionary modeling; Genetic programming; Genetic algorithm

## 1. Introduction

Regardless of the primary and the secondary battery systems, the discharge lifetime is one of the most important indicators concerned by users. Especially with the development of electric vehicles, it presses for an instantaneous, non-invasive and remote method to measure and predict the discharge lifetime of sealed batteries. One answer to this is to build the discharge lifetime model of battery systems based on observed data.

The traditional modeling methods include data fitting, regression analysis and some other approximation methods (He, 1995). Some discharge lifetime models have been obtained by using these methods, such as the Lifetime Equation of Fast Detection for the button type Li/MnO<sub>2</sub> cell (Liu and Gui, 1993, 1996) and the

Peukert Equation for the lead-acid battery (Bode, 1977). The usual steps of these methods are to choose a model structure for the system initially, to determine the parameters contained in the model subsequently, and finally, to test the validity of the model (Xiang et al., 1988; Gan, 1991). However, due to the complexity and nonlinearity of electrochemical reactions of battery systems, it is usually difficult for people to choose a suitable model structure without having sufficient domain details and human expertise, and the determination of parameters also requires the modeler to have a rich mathematical knowledge and professional skills. In Qi et al. (1995), Qi and his coworkers applied the back propagation (BP) algorithm and the multilayer perceptron in neural networks to build 'black box' model for battery systems. But the performance of neural networks depends largely upon the topology and weight values, and it is usually an arduous task to design a neural network, including the topology and connection weights, for a specific problem. Recently Salkind et al.

\* Corresponding author. Tel.: + 86-27-87682681; fax: + 86-27-87882661.

E-mail address: jxyu@whu.edu.cn (H. Cao).

(1999) used the fuzzy logic methodology to predict the states of charge and health of battery systems, but their predicted values did not coincide with the measured values very well.

To overcome the drawbacks of the available methods mentioned above, we propose a hybrid evolutionary modeling algorithm (HEMA) to implement the automatic modeling of discharge lifetime with multiple impact factors for battery systems as well as make predictions. The experimental results on lithium-ion batteries testify the effectiveness of the algorithm.

## 2. Hybrid evolutionary modeling method

Genetic Programming (GP) (Banzhaf et al., 1997) is a new branch of Evolutionary Computation (Eiben and Michalewicz, 1999) which was introduced by John Koza and co-workers (Koza, 1992; Koza et al., 1994, 1999) in the 1990s. GP is an extension of John Holland's Genetic Algorithm (GA) (Holland, 1975) in which the genetic population consists of computer programs of varying sizes and shapes. In standard GP, computer programs are represented as parse trees rather than bit strings, where a branch node represents an element from a function set  $F$ . This function set usually contains arithmetic operations and elementary functions of at least one argument. A leaf node represents an element from a terminal set  $T$ . This terminal set usually contains variables and constants. These symbolic programs are subsequently evaluated by running them on a set of 'fitness cases'. Fitter programs are selected for recombination to create the next generation using crossover and mutation. This step is iterated for some number of generations until the termination criterion is satisfied.

As GP uses parse trees to represent the individuals in a population whose sizes and types are not fixed, it is well suited for the task of the evolutionary modeling of some complex systems. First of all, as for the representation of a model, any model can be represented as a tree structure of GP once the  $T$  and the  $F$  are chosen appropriately. Secondly, as for the diversity of a model, the model structure can be conducted and modified flexibly by using standard GP mutation operator and crossover operator. This enables the modeling algorithm to find suitable models ultimately. Finally, as for the evaluation of a model, the criteria to evaluate the quality of a model can be easily converted into a fitness function in GP.

However, when we apply standard GP to build the discharge lifetime model of battery systems, one major problem arises. Since the fitness value of a model depends largely upon the values of its parameters, a model with a favorable structure will have a great probability of being eliminated from the population

during the evolution if the randomly generated parameters are inappropriate. Consequently it is unlikely for us to obtain a highly accurate model for the system. Moreover the evolutionary process can be slow due to the large number of generations needed, as well as suffering from the premature convergence phenomenon which means that the population quickly converges to a suboptimal solution. To reduce the impact of these problems, we consider embedding the parameter optimization process into the evolutionary modeling process.

There are a great many traditional methods for solving nonlinear parameter estimation problems, such as direct search methods, Hooke–Jeeves methods, Nelder–Mead methods, gradient methods and variable metric methods (Nash and Walker-Smith, 1987), but they are only applicable to specific types of problems and require restricted conditions to be imposed upon the model, such as being continuous, differentiable or single-peaked. While the models randomly generated by the modeling algorithm are various and each of them may possess multiple parameters, we have no way of optimizing those parameters by using the traditional methods mentioned above. Here we propose a GA with real-valued encoding to implement the parameter optimization of arbitrary models. Different from the genetic operators used by other researchers (Moros et al., 1996; Xiong et al., 1997; VanderNoot and Abrahams, 1998), we used a novel crossover operator based on the non-convex linear combination of multiple parents during the recombination of the population, which proves to work stably and effectively in the parameter optimization process.

The two main processes, namely the structure optimization process based on GP and the parameter optimization process based on a GA, accompanied by the simplification and normalization of models, the sharing of the same kind of models, constitute the framework of the evolutionary modeling.

## 3. Hybrid evolutionary modeling algorithm

Taking the evolutionary modeling of discharge lifetime of battery systems as an example, we give the detailed description of the hybrid evolutionary modeling algorithm in this section.

### 3.1. Initialization of model population

Suppose there are  $n$  factors denoted as  $x_1, x_2, \dots, x_n$  which affect the discharge lifetime of battery systems. When initializing the model population, the algorithm generates POPSIZE individuals randomly having the form of  $y = f(x_1, x_2, \dots, x_n)$  and each individual can be expressed as a Lisp statement where operators precede

their arguments. For example, the Lisp expression of a discharge lifetime model including two impact factors

$$f(x_1, x_2) = (x_1 + \sin x_2 - \exp(3x_2)) / (x_1(x_2 + 3.5)) \quad (1)$$

can be written as

$$f(x_1, x_2) = (/ (- (+ x_1 \sin(x_2)) \exp (* (3x_2))) (* x_1 (+ x_2 3.5))) \quad (2)$$

which can be easily represented as a binary tree as shown in Fig. 1. The maximum depth of a tree is restricted by a constant  $D$  and the complexity of a model is measured by the number of nodes  $N_{nodes}$  contained in its tree.

### 3.2. Simplification and normalization of models

The simplification of a model means simplifying the tree structure of each individual in the model population by replacing those subtrees which consist of arithmetic operations between constants by their calculated values. It affects the number of parameters to optimize and the number of nodes contained in the tree.

The normalization of a model means adjusting the structures of its subtrees whose roots are ‘+’ (plus) or ‘\*’ (multiplication), and whose left branches or right branches are a constant, to ensure that the constant always lies on the right of ‘+’ or ‘\*’ in the S-expression of the model. This operation is helpful so that ‘ $a + x$ ’ and ‘ $x + a$ ’ or ‘ $a * x$ ’ and ‘ $x * a$ ’ will not be regarded as different structures, thus eliminating redundant work in the optimization process.

### 3.3. Fitness evaluation of models

Modelers often regard how well the model fits the historic data as the only criterion when evaluating a model’s quality without consideration of the model’s parsimony. To balance the accuracy and parsimony of a model, we define the sum of minimal square error (MSE) and the number of nodes as the fitness function, namely,

$$fitness = \sum_{i=1}^m (\hat{y}_i - y_i)^2 + C \times N_{nodes} \quad (3)$$

where  $m$  is the number of data points to be fitted,  $\hat{y}_i$  and  $y_i$  are the fitted value of the model and the observed value at the  $i$ th data point respectively,  $C$  is a constant coefficient which can be used to control the complexity of a model. The MSE and the number of nodes are comparable in magnitude for our example, we set  $C = 1$  to emphasize that the accuracy and the parsimony play the same important roles in evaluation of a model. Obviously, here the lower the fitness value, the better the model.

### 3.4. Recombination of model population

The model population is recombined into the next generation by selecting the individuals to do some genetic operations including crossover, mutation and reproduction. To choose better individuals to generate the offspring, we use tournament selection with sample size of 4 in the algorithm. That is, every time we randomly select four individuals in the population and compare their fitness values. The best one is chosen into the next generation to breed the offspring. An elitist strategy is also adopted, which places the best individual of the population into the next generation.

In each generation, we use the deterministic dynamic adaptation method (Hinterding et al., 1997) to alter the mutation rate of individuals according to fitness so that the fitter individuals change only in a small range (with a small mutation rate) but the less fit individuals in a wide range (with a large mutation rate). Given parent  $i$ , its mutation rate  $p_m(i)$  is defined as

$$p_m(i) = 0.3 * (1 - f_{min}/f_i) \quad (4)$$

where  $f_{min}$  is the fitness value of the best individual in the current generation and  $f_i$  is the fitness value of  $i$ .

Crossover is performed by choosing a random point in each parent, exchanging the subtrees beneath those points to produce two new trees and using either of them as the offspring on the condition that its depth does not exceed  $D$ .

Mutation is performed by choosing a random point in the parent and replacing the subtree beneath that point by a randomly generated subtree to produce the offspring.

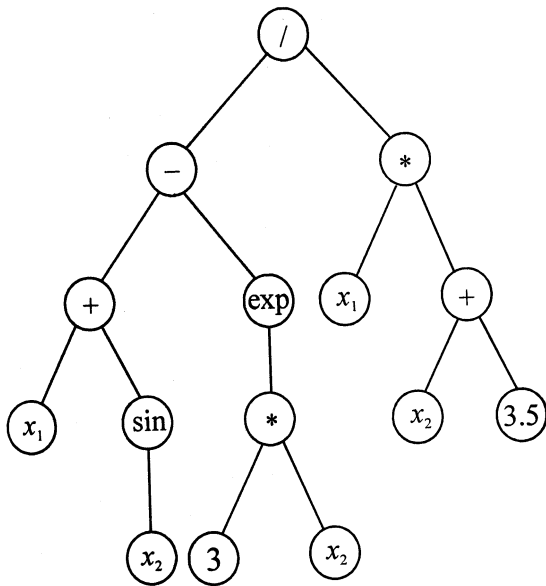


Fig. 1. The binary representation of the model  $f(x_1, x_2)$ .

### 3.5. Parameter optimization of models

At the beginning of this process, we first examine whether the model structure has been optimized in the current generation. If so, we do nothing with it, otherwise we use the GA described below to optimize its parameters.

#### 3.5.1. Initialization of parameter population

We first check all the constants in the model, including counting the number of constants  $l$  and recording their positions. When initializing the parameter population, the GA generates  $N$  individuals. Each of them can be represented as an  $l$ -dimensional row vector  $(c_1, c_2, \dots, c_l)$ . Each component  $c_i$  for  $i = 1, 2, \dots, l$  is encoded as a floating number and generated randomly in a symmetrical interval  $[-MAX_c, MAX_c]$ . The value of  $MAX_c$  can be set as an arbitrary positive integer, but it usually needs not be too big due to the fact that by performing the crossover operation described below, those parameters can adjust their values adaptively and may skip out of this range if necessary. In our algorithm, we set  $MAX_c = 20$ .

#### 3.5.2. Fitness evaluation of parameter individuals

Before evaluating the fitness of an individual in the parameter population, we return to the original model and replace all constants with the corresponding components of the row vector (i.e. the individual). We then follow the same procedure as in Section 3.3 to calculate the fitness.

#### 3.5.3. Production of a new parameter individual

We use a novel crossover operator to create a new individual in the parameter population in the following way. Randomly select  $M$  different individuals from the old population ( $M > 2$ ) denoted as  $X_1, X_2, \dots, X_M$  where  $X_k = (c_1^{(k)}, c_2^{(k)}, \dots, c_l^{(k)}) (k: 1 \sim M)$ . Produce  $M$  random coefficients  $\alpha_k$ , where  $\alpha_k$  ranges from  $a$  to  $b$  ( $a < 0, b > 1$ ) and satisfies  $\sum_{k=1}^M \alpha_k = 1$ . Generate a new individual  $X = (c_1^*, c_2^*, \dots, c_l^*)$  by the nonconvex linear combination of these  $M$  individuals as follows

$$X = \sum_{k=1}^M \alpha_k X_k \quad (5)$$

If any parameter  $c_i^* (1 \leq i \leq l)$  of  $X$  is out of the pre-designated range  $[-MAX_{para}, MAX_{para}]$ , abandon  $X$  and reproduce another one as above.

#### 3.5.4. Replacement of parameter population

Evaluate the fitness of the new individual  $X$  following the same procedure as Section 3.5.2. If the fitness value of  $X$  is lower than that of the worst individual in the current population, then replace the worst one with  $X$ .

The whole process of parameter optimization is iterated between Section 3.5.3 and Section 3.5.4 until a

predetermined maximum number of crossover  $MAX_{ox}$  is reached.

There are four adjustable control parameters  $N, M, a, b$  in the GA. Setting their optimal values depends upon the properties of the specific problem. It is worth noting that the essence of the algorithm lies in the random search of the subspace determined by multiple parents. One advantage of the multi-parent crossover is that the crossover search space is determined by multiple points rather than only two points used in a pairwise crossover. Moreover the interval of weighted coefficients,  $(a, b)$ , is not in the range of  $(0, 1)$  as commonly used, but satisfies  $a < 0, b > 1$ , which makes the crossover search possible to skip out of the space determined by the parents.

### 3.6. Sharing of the same kind of models

To ensure the diversity of the model population, as for the same kind of models which have an identical structure, we only keep the one whose parameters are optimized best into the next generation while perform mutation operations on others.

The whole evolutionary modeling process is iterated between Section 3.2 and Section 3.6 until the termination criterion is satisfied whereby a maximum generation,  $MAX_{genos}$ , is reached.

### 3.7. Prediction of model

Once the best-evolved model is obtained in one run, we then take some unseen validation data to predict the corresponding discharge lifetime of battery systems based on the model.

## 4. Experimental

### 4.1. Preparation of batteries and collection of sample data

The UR2025 lithium-ion battery had a sandwich structure using  $LiCoO_2$  and modified petroleum coke as positive and negative materials respectively. The separator was a Celgard 2400 microporous polyethylene membrane. The electrolyte was 1 M  $LiClO_4$  dissolved in a 1:1 mixture of propylene carbonate (PC) and dimethoxyethane (DME). We chose nine UR2025 lithium-ion batteries randomly as experimental batteries. Every battery was charged to 4.2 V at the current of 1.0 mA, then discharged under different working temperature and load resistance. The voltage of the battery was detected automatically every 3 min. Once its values is less than 2.75 V, the discharge circuit would be cut off and we recorded the discharge time of the battery between 4.2 and 2.75 V. As for each pair of tempera-

Table 1  
The measured values of discharge lifetime of UR2025 lithium-ion battery under different working temperature and load resistance

| Serial No. $i$ | Temperature (°C) $x_1$ | Resistance ( $k\Omega$ ) $x_2$ | Measured discharge lifetime ( $h$ ) $y_i$ |
|----------------|------------------------|--------------------------------|---|
| 1              | -20                    | 4.7                            | 20.6                                      |
| 2              | -20                    | 10                             | 73.0                                      |
| 3              | -20                    | 15                             | 118.3                                     |
| 4              | 0                      | 4.7                            | 39.2                                      |
| 5              | 0                      | 10                             | 83.7                                      |
| 6              | 0                      | 15                             | 128.5                                     |
| 7              | 20                     | 4.7                            | 43.5                                      |
| 8              | 20                     | 10                             | 87.8                                      |
| 9              | 20                     | 15                             | 130.2                                     |
| 10             | 40                     | 4.7                            | 42.7                                      |
| 11             | 40                     | 10                             | 86.2                                      |
| 12             | 40                     | 15                             | 127.6                                     |
| 13             | -10                    | 12                             | 98.8                                      |
| 14             | 30                     | 8                              | 68.6                                      |
| 15             | 40                     | 5                              | 44.5                                      |
| 16             | 15                     | 14                             | 120.3                                     |

ture and resistance, we calculated the mean value of nine batteries as its measured discharge lifetime which is listed in Table 1.

4.2. Modeling experiments

The function set  $F$  and the maximum depth  $D$  are two important control parameters that affect both the search space and the complexity of models in the algorithm. To determine the optimal  $F$  and  $D$  which are most suitable for searching the discharge lifetime models of batteries, we have experimented with  $D = 4, 5, 6, 7$  on  $F = F_1, F_2, F_3, F_4$  respectively, where  $F_1 = \{+, -, *, /\}$ ,  $F_2 = \{+, -, *, /\}$ ,  $\sin, \cos\}$ ,  $F_3 = \{+, -, *, /\}$ ,  $\exp, \ln\}$ ,  $F_4 = \{+, -, *, /\}$ ,  $\sin, \cos, \exp, \ln\}$ . In every modeling experiment, we take the first twelve measured data points in Table 1 as modeling samples to build the relationship model between the discharge lifetime and its two impact factors (i.e. working temperature and load resistance), and make predictions for the last four unseen data points based on the model.

All the experiments are performed on a Pentium III (500 MHz) computer using Visual C++ 5.0 Compilers. The parameter settings are shown in Table 2.

In addition, to evaluate the accuracy of a model, we define fitting error (FE) and prediction error (PE) as

$$FE = \sum_{i=1}^{12} (\hat{y}_i - y_i)^2 \quad PE = \sum_{i=13}^{16} (\hat{y}_i - y_i)^2 \quad (6)$$

where  $y_i$  is the measured value of  $y$  at the  $i$ th data point,  $\hat{y}_i$  is the fitted value and the predicted value in FE and PE respectively. Forty runs are conducted independently for each experiment ( $D, F$ ) and we compute their average values including average fitting error (AFE), average prediction error (APE) and mean run-

ning time (MRT). In one run, as for the final best-evolved model, if there is at least one data point  $y_i (13 \leq i \leq 16)$ , whose relative prediction error  $|\hat{y}_i - y_i|/y_i$  is greater than 1, we declare this run a failure; otherwise a success. We only take into account the successful runs ( $N_{succ}$ ) when calculating the average values.

5. Results and discussion

The average fitting errors and prediction errors of 40 runs on different ( $D, F$ ) are listed in Table 3. From Table 3 we can see that as far as the AFE is concerned, on the same function set, an increase in  $D$  leads to a lower AFE, but with a diminishing gradient. For example, on  $F_1$ , the AFE of  $D = 4$  is more than double that of  $D = 5$ , while the difference between the AFEs of  $D = 6, 7$  is trivial. For the same  $D$ , the disparity in the AFEs on different function sets is not obvious. As far as the APE is concerned, on the same function set, the APEs varies inconsistently with the growth of  $D$ , which get smaller or larger now and then. While in most

Table 2  
Parameter settings of the hybrid evolutionary modeling algorithm

|                             |   |
|-----------------------------|---|
| Structure optimization (GP) | $F = F_1, F_2, F_3, F_4$ $T = \{x_1, x_2, c\}$ ( $c$ is a random constant) $D = 4, 5, 6, 7$<br>POPSIZE = 200 $MAX_{geno} = 150$ |
| Parameter optimization (GA) | $N = 30$ $M = 8$ $a = -0.5$ $b = 1.5$<br>$MAX_{ox} = 500$ $MAX_{para} = 500$  |

Table 3  
The average fitting errors and prediction errors on different ( $D$ ,  $F$ ) (40 runs)

| $F$   | $D = 4$ |        | $D = 5$ |        | $D = 6$ |        | $D = 7$ |        |
|-------|---------|--------|---------|--------|---------|--------|---------|--------|
|       | AFE     | APE    | AFE     | APE    | AFE     | APE    | AFE     | APE    |
| $F_1$ | 45.95   | 56.36  | 20.66   | 96.18  | 14.59   | 47.61  | 14.47   | 143.69 |
| $F_2$ | 60.92   | 136.85 | 30.21   | 137.14 | 22.30   | 140.90 | 13.33   | 123.21 |
| $F_3$ | 43.44   | 42.39  | 31.21   | 176.09 | 17.85   | 46.88  | 16.84   | 239.18 |
| $F_4$ | 55.74   | 106.23 | 27.82   | 179.66 | 13.38   | 144.03 | 12.90   | 521.75 |

Table 4  
The scoring results of different function sets in terms of AFE and APE

| $F$   | $D = 4$ |     | $D = 5$ |     | $D = 6$ |     | $D = 7$ |     | Total |
|-------|---------|-----|---------|-----|---------|-----|---------|-----|-------|
|       | AFE     | APE | AFE     | APE | AFE     | APE | AFE     | APE |       |
| $F_1$ | 3       | 3   | 4       | 4   | 3       | 3   | 2       | 3   | 25    |
| $F_2$ | 1       | 1   | 2       | 3   | 1       | 2   | 3       | 4   | 17    |
| $F_3$ | 4       | 4   | 1       | 2   | 2       | 4   | 1       | 2   | 20    |
| $F_4$ | 2       | 2   | 3       | 1   | 4       | 1   | 4       | 1   | 18    |

Table 5  
The number of successful runs and average running time on different ( $D$ ,  $F$ ) (40 runs)

| $F$   | $D = 4$           |         | $D = 5$           |         | $D = 6$           |         | $D = 7$           |         |
|-------|-------------------|---------|-------------------|---------|-------------------|---------|-------------------|---------|
|       | $N_{\text{succ}}$ | MRT (s) | $N_{\text{succ}}$ | MRT (s) | $N_{\text{succ}}$ | MRT (s) | $N_{\text{succ}}$ | MRT (s) |
| $F_1$ | 38                | 378     | 35                | 582     | 37                | 841     | 37                | 878     |
| $F_2$ | 36                | 381     | 37                | 622     | 37                | 806     | 37                | 954     |
| $F_3$ | 37                | 361     | 38                | 581     | 35                | 891     | 33                | 933     |
| $F_4$ | 37                | 355     | 39                | 595     | 36                | 858     | 38                | 894     |

cases, when  $D$  is 4 or 6, the APE is relatively low. For the same  $D$ , the APEs differ remarkably when choosing different function sets, but it is difficult to find the most appropriate function set whose APEs are consistently small for all tree depths.

To find out the optimal function set with consideration both the AFE and the APE, we score the four function sets according to their AFEs and APEs. Namely, for the same  $D$ , we rank the four function sets by their AFEs (APEs) and give them the score from 1 to 4 in sequence. The smallest is marked the highest score. The final scoring results are listed in Table 4.

As shown in Table 4,  $F_1$  earns the highest score and thus is determined as the optimal function set. On the  $F_1$ , the AFEs for  $D = 6, 7$  are both small, while the APE for  $D = 7$  is much larger than that of  $D = 6$ . Obviously, it is preferable to choose 6 as the optimal tree depth.

The number of successful runs and average running time of 40 runs on different ( $D$ ,  $F$ ) are listed in Table 5.

From Table 5, we can see that for each function set, the running time increases significantly with the growth of  $D$ . But for the same  $D$ , the time cost on different function sets are almost the same. These facts can be explained as follows. As the complexity of a model is related to the  $D$  closely, usually the larger the  $D$ , the longer the expression of a model. It is certainly more time-consuming to deal with a long expression when calculating the fitness value of the model. While the enlarging of a function set can only affect the search space of a model, but has little effect on its complexity. In addition, from the fact that the number of successful runs is over 35 (85% of success rate) for each ( $D$ ,  $F$ ), it demonstrates the effectiveness of the algorithm in modeling the discharge lifetime of lithium-ion batteries.

As a great variety of models can be obtained in different runs, it is an important task to pick the superior models which can best describe the practical system. There are two criteria when we judge a model is better or not. One is that the model should be both

accurate and simple. In other words, its FE, PE and  $N_{\text{nodes}}$  are relatively small. The other is that the model should be reasonable in the whole definition fields of all independent variables. To achieve this, one simple way is to draw out the three dimensional (3D) graph of  $f(x_1, x_2)$  (lifetime) versus  $x_1$  (temperature) and  $x_2$  (resistance) which can represent the predicted value surface of the model clearly, then to judge its reasonability based on the features of the landscape and some background knowledge.

By drawing out the 3D graphs of 38 models obtained in successful runs when  $(D, F) = (6, F_1)$ , we find that there are a few models which have small FEs or PEs but whose landscapes conflict with the actual discharge law of batteries. For example, we obtain a model having the form of

$$\begin{aligned} f^*(x_1, x_2) &= 8.4034x_2 + 3.8571/(x_1 - 18.713) \\ &+ x_1/(12.457(x_1 + 18.644)(x_1 + x_2)) \end{aligned} \quad (7)$$

whose FE and PE are relatively small, 15.2285 and 11.6024, respectively. However, by observing its 3D graph as illustrated in Fig. 2, we see that the curved surface is not continuous and the discharge lifetime at some points are even negative, which are impossible in the actual discharge process of batteries.

In the 38 runs, what occurs most frequently (10 runs) is the model having the form of

$$f_1(x_1, x_2) = 8.483x_2 + 8.483x_1/(2x_1 + x_2 + 44.136) \quad (8)$$

whose 3D graph is a smooth, continuous and monotonic curved surface as depicted in Fig. 3. We also

find some other superior models. Despite the different structures from  $f_1$ , their FEs and PEs are also small and their landscapes are rather similar to  $f_1$ . We list two of them as follows

$$\begin{aligned} f_2(x_1, x_2) &= 8.564x_2 + 26.851(x_1 - x_2)/(x_1x_2 + 26.851x_2) \\ &(\text{FE} = 14.5805 \quad \text{PE} = 3.9550) \end{aligned} \quad (9)$$

$$\begin{aligned} f_3(x_1, x_2) &= 8.507x_2 + x_1(x_2 + 19.282)/(x_2(x_1 + 25.127)) \\ &(\text{FE} = 12.8700 \quad \text{PE} = 5.3008) \end{aligned} \quad (10)$$

The fitting and prediction results of the three models  $f_1 \sim f_3$  are listed in Table 6. Their evolution curves are depicted in Fig. 4.

From Table 6, we can see that both the fitting and prediction accuracy of the three models are high, with the smallest relative error 0.001 and the largest one 0.046. It shows that multiple superior models of discharge lifetime can be obtained by running the HEMA. Moreover, from Fig. 4, we can see that the computer can usually search the superior models in less than 40 generations which further demonstrates the efficiency of the algorithm.

## 6. Conclusions

A hybrid evolutionary modeling algorithm HEMA is proposed in this paper. Its main idea is to embed a genetic algorithm GA into genetic programming GP, where GP is employed to optimize the structure of a model, while a GA is employed to optimize its parameters. The algorithm proves to work quickly and effectively in building the discharge lifetime models with

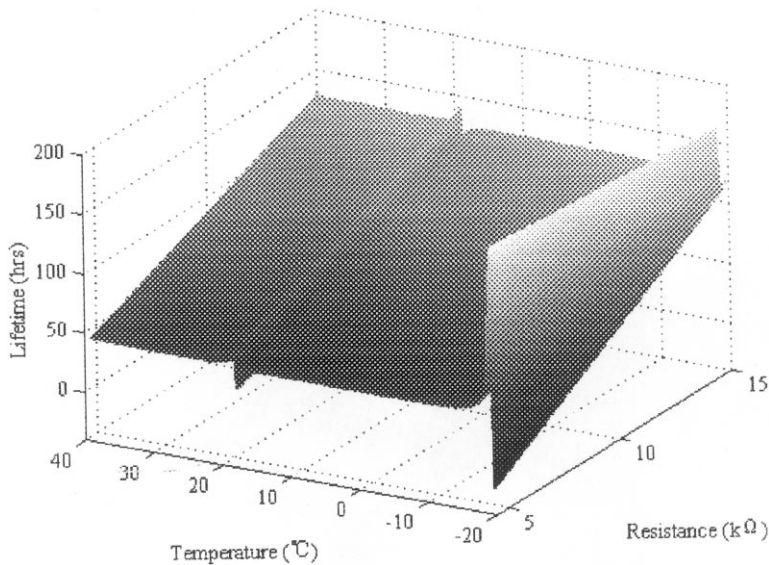
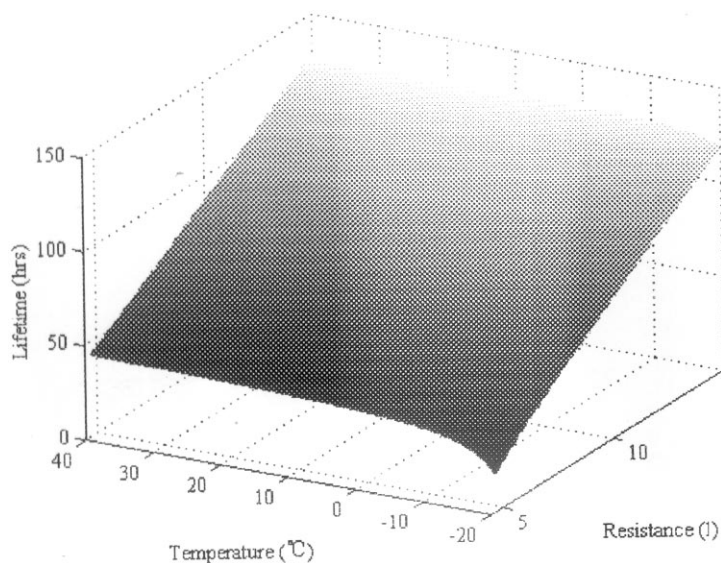


Fig. 2. The 3D graph of  $f^*(x_1, x_2)$ .

Fig. 3. The 3D graph of  $f_1(x_1, x_2)$ .Table 6  
The fitting and prediction results of the three best-evolved models

| Serial No. $i$ | Fitted/predicted value |        |        | Relative error $ (y_i - f(x_i))/y_i $ |       |       |
|----------------|------------------------|--------|--------|---------------------------------------|-------|-------|
|                | $f_1$                  | $f_2$  | $f_3$  | $f_1$                                 | $f_2$ | $f_3$ |
| 1              | 20.67                  | 19.65  | 20.08  | 0.003                                 | 0.046 | 0.025 |
| 2              | 72.83                  | 73.88  | 73.65  | 0.002                                 | 0.012 | 0.009 |
| 3              | 118.38                 | 119.31 | 118.69 | 0.001                                 | 0.009 | 0.003 |
| 4              | 39.87                  | 39.25  | 139.98 | 0.017                                 | 0.001 | 0.020 |
| 5              | 84.83                  | 84.64  | 85.07  | 0.013                                 | 0.011 | 0.016 |
| 6              | 127.24                 | 127.46 | 127.61 | 0.010                                 | 0.008 | 0.007 |
| 7              | 41.78                  | 42.12  | 42.25  | 0.040                                 | 0.032 | 0.029 |
| 8              | 86.63                  | 86.21  | 86.37  | 0.013                                 | 0.018 | 0.016 |
| 9              | 128.95                 | 128.65 | 128.62 | 0.010                                 | 0.012 | 0.012 |
| 10             | 42.50                  | 43.27  | 43.12  | 0.005                                 | 0.013 | 0.010 |
| 11             | 87.36                  | 86.84  | 86.87  | 0.013                                 | 0.007 | 0.008 |
| 12             | 129.68                 | 129.12 | 129.01 | 0.016                                 | 0.012 | 0.011 |
| 13             | 99.45                  | 99.84  | 100.36 | 0.007                                 | 0.011 | 0.016 |
| 14             | 70.13                  | 69.81  | 69.91  | 0.022                                 | 0.018 | 0.019 |
| 15             | 45.04                  | 45.63  | 45.52  | 0.012                                 | 0.025 | 0.023 |
| 16             | 120.20                 | 119.94 | 119.99 | 0.001                                 | 0.003 | 0.003 |

multiple impact factors for lithium-ion batteries. It has the following advantages compared with most existing modeling methods:

1. The structure optimization and the parameter optimization of models can be performed simultaneously by using the two main processes in the algorithm. The modelers need not know the exact structure of a model and the number of parameters it contained in advance.
2. The whole modeling process is carried on automati-

cally. The modelers need not have rich professional knowledge and abundant domain details.

3. Multiple superior models with different structures can be obtained by running the algorithm. The fitting and prediction accuracy of these models are usually high.
4. The algorithm has a strong generality. It can be applied to the prediction of discharge lifetime of any other batteries, pattern recognition, quantitative structure-activity relationship of pharmaceuticals,

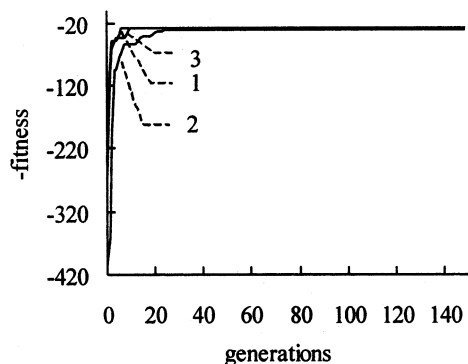


Fig. 4. Evolution curves of three best-evolved models: (1)  $f_1(x_1, x_2)$ ; (2)  $f_2(x_1, x_2)$ ; (3)  $f_3(x_1, x_2)$ .

chemical clustering analysis and in many other fields.

Additionally, in our experiments, we only take into account two impact factors when studying the discharge lifetime models of batteries, namely, the working temperature and the loaded resistance. If using the electrochemical parameters measured by current pulse response and impedance technique as independent variables, the models built by running the HEMA should describe the law of the discharge lifetime of battery systems more reasonably.

#### Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 70071042, No. 60073043 and No. 29833090) and National Laboratory for Parallel and Distributed Processing. The authors would like to thank the anonymous referees for their helpful comments on the paper.

#### References

- Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D., 1997. Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann.
- Bode, H., 1977. Lead-Acid Batteries. Wiley, New York.
- Eiben, A.E., Michalewicz, Z., 1999. Evolutionary Computation. IOS Press, Amsterdam.
- Gan, R.-C., 1991. The Statistical Analysis of Dynamic Data. Beijing University of Science and Technology Press, Beijing (in Chinese).
- He, J.-X., 1995. System Modelling and Mathematical Models. Fujian Science & Technology Press, Xiamen (in Chinese).
- Hinterding, R., Michalewicz, Z., Eiben, A., 1997. Proceedings of the 4th International Conference on Evolutionary Computation, IEEE Press, 65.
- Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press.
- Koza, J.R., 1992. Genetic Programming: on the Programming of Computers by means of Natural Selection. MIT Press, Cambridge.
- Koza, J.R., 1994. Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge, MA.
- Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A., 1999. Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann, San Francisco, CA.
- Liu, R.-H., Gui, C.-Q., 1993. Chinese J. Power Sources 17 (2), 25 (in Chinese).
- Liu, R.-H., Gui, C.-Q., 1996. Battery Bimonthly 26, 166 (in Chinese).
- Moros, R., Kalies, H., Rex, H.G., Schaffarczyk, St., 1996. Comput. Chem. Eng. 20, 1257.
- Nash, J.C., Walker-Smith, M., 1987. Nonlinear Parameter Estimation. Marcel Dekker, New York, Basel.
- Qi, G.-G., Qian, J., Ding, D.-H., 1995. Journal of Tsinghua University, Sci. Tech, 35 (5), 38 (in Chinese).
- Salkind, A.J., Fennie, C., Singh, P., Atwater, T., Reisner, D.E., 1999. J. Power Sources 80, 293.
- VanderNoot, T.J., Abrahams, I., 1998. J. Electroanal. Chem 448, 17.
- Xiang, J.-T., Du, J.-G., Shi, J.-E., 1988. Dynamic Data Processing: Time Series Analysis. Meteorology Press, Beijing (in Chinese).
- Xiong, Y., Pan, Z.-J., Wang, H., Wu, D.-Q., Kang, L.-S., Qu, S.-I., 1997. Acta Phys. Chim. Sinica 13, 503 (in Chinese).