Journal of Mechatronics, Electrical Power, and Vehicular Technology 8 (2017) 40–49

Journal of Mechatronics, Electrical Power, and Vehicular Technology

e-ISSN: 2088-6985
p-ISSN: 2087-3379

www.mevjournal.com

Comparison between RLS-GA and RLS-PSO for Li-ion battery SOC and SOH estimation: A simulation study

Latif Rozaqi [a, *], Estiko Rijanto [a], Stratis Kanarachos [b]

[a] *Research Center for Electrical Power and Mechatronics, Indonesian Institute of Sciences (LIPI)*
Kampus LIPI, Jalan Sangkuriang, Gd.20, Bandung 40135, Indonesia
[b] *Centre for Mobility & Transport, Coventry University*
Priory Street, Coventry, CV1 5FB, United Kingdom

Received 22 March 2017; received in revised form 31 May 2017; accepted 03 July 2017
Published online 31 July 2017

Abstract

This paper proposes a new method of concurrent SOC and SOH estimation using a combination of recursive least square (RLS) algorithm and particle swarm optimization (PSO). The RLS algorithm is equipped with multiple fixed forgetting factors (MFFF) which are optimized by PSO. The performance of the hybrid RLS-PSO is compared with the similar RLS which is optimized by single objective genetic algorithms (SOGA) as well as multi-objectives genetic algorithm (MOGA). Open circuit voltage (OCV) is treated as a parameter to be estimated at the same time with internal resistance. Urban Dynamometer Driving Schedule (UDDS) is used as the input data. Simulation results show that the hybrid RLS-PSO algorithm provides little better performance than the hybrid RLS-SOGA algorithm in terms of mean square error (MSE) and a number of iteration. On the other hand, MOGA provides Pareto front containing optimum solutions where a specific solution can be selected to have OCV MSE performance as good as PSO.

Keywords: Li-Ion; battery; state of charge (SOC); state of health (SOH); recursive least square (RLS); particle swarm optimization (PSO); genetic algorithm (GA)

I. Introduction

Battery states of charge (SOC) and state of health (SOH) have to be estimated properly in order to build a good battery management system (BMS) for electric vehicles. It is known that Lithium battery has time varying nonlinear dynamics where the speed of parameter values change is different on each parameter.

There have been many SOC estimation methods proposed by other researchers. A mixed coulomb-counting and model-based algorithm was proposed for SOC estimation of $LiFePO_4$ battery [1, 2, 3]. Current and terminal voltages are measured, and an integral feedback controller is used to compensate terminal voltage and SOC estimation errors. A PI observer was proposed for SOC estimation of Li-Ion battery where the SOC and polarization voltage are used as state variables [4]. More robust and advanced methods such as Kalman filter [5, 6] and Sliding Mode Observer [7] have also been used. However, the above methods assumed that the battery parameter values are constant or constant at some specified region, and treated the parameter values variance as a disturbance. A deeper investigation is required to evaluate the stability and estimation performance when the parameter values vary largely.

Recursive Least Square (RLS) has also been applied for battery SOC estimation. It was applied to a single RC Thevenin model of Lithium-Ion battery whose open circuit voltage (OCV) was depicted by Nernst equation [8]. It was applied to a double polarization RC Thevenin model of a $LiFePO_4$ battery of which the SOC is estimated by online identification of OCV and the predetermined OCV-SOC look up table [9]. Moving window least square (MWLS) method was developed and applied to single RC

* Corresponding Author. Tel: +62 22 250 3055
E-mail address: latiefrozaqie@gmail.com

doi: https://dx.doi.org/10.14203/j.mev.2017.v8.40-49

Accreditation Number: (LIPI) 633/AU/P2MI-LIPI/03/2015 and (RISTEKDIKTI) 1/E/KPT/2015.

Thevenin models of Li-Ion and Li-Polymer batteries [10]. The SOC and battery parameters are co-estimated using a combination of MWLS and linear observer. All the above RLS based SOC estimation methods use single forgetting factor. RLS with multiple fixed forgetting factors (MFFF) has been used to estimate SOC of a Li-Ion battery. The forgetting factors were optimized using Genetic Algorithm (GA), and it was proved that the algorithm provided better performance than RLS with single forgetting factor [11]. An interesting result has been reported on the estimation of battery SOH using RLS without forgetting factor. Estimation speed and reliability have been compared between internal ohmic resistance based estimation and capacity based estimation. It can be concluded that SOH estimation based on internal resistance is faster and more reliable [12].

Many researchers have used PSO algorithm for estimating battery SOC in different ways. Support Vector Regression (SVR) was used to estimate SOC of a Lead-acid battery in which hyperparameters of the SVR are determined using PSO [13]. A hybrid model which combined multivariate adaptive regression splines (MARS) and PSO was used to estimate SOC of a $LiFeMnPO_4$ battery cell. PSO was used to find the optimal parameters of the MARS model. As a result, SOC is represented by 29 pairs of basis functions and their coefficients [14]. Stepwise method considering multicollinearity was used to predict battery SOC. PSO was used to find optimum coefficient values, and the SOC can be expressed using 9 variables [15].

Some methods for concurrent estimation of battery SOC and SOH have been proposed. Dual Kalman Filter (DKF) was used for adaptive state and parameter estimation of Lithium-Ion batteries. Diffusion voltage, state of charge, and internal resistance are selected as state variables, while cell capacity, diffusion resistance, and diffusion capacitance are chosen as parameters. One Kalman filter is used for state estimation and the other Kalman filter is used for parameter values [16]. A hybrid battery model was proposed which consists of an enhanced Coulomb counting algorithm and an electrical circuit model. The Coulomb counting algorithm is used for SOC estimation while the electrical circuit model is used for electrical impedance estimation. Five parameters are used in the electrical model those are internal resistance, one pair of resistance and capacitance which governs short-term dynamics, and one pair of resistance and capacitance which governs long-term dynamics. A set of nonlinear discrete time dynamic equations are formulated using battery terminal voltage and current as measured signals as well as six unknown parameters. The unknown parameters include internal resistance, open circuit voltage, two parameters as a function of short-term dynamical resistance and capacitance, and two parameters as a function of long-term dynamical resistance and capacitance. PSO is used to find a set of values of the unknown parameters which minimizes the selected fitness function. The OCV is then used for SOC estimation using the enhanced Coulomb counting method [17].

The DKF involves extended Kalman filter for parameter identification which adds computational burden. The use of PSO in the hybrid model requires execution of the PSO iteration independently to the SOC calculation routine which may rise a problem since there is no guarantee that the stopping criterion is fulfilled in the sampling period of SOC calculation.

An adaptive algorithm which can estimate SOC and SOH concurrently and can work under single sampling time and less computing burden is necessary. In this paper, such requirement is answered by proposing a new algorithm named hybrid Recursive Least Square – Particle Swarm Optimization (RLS-PSO). RLS is equipped with multiple fixed forgetting factors whose the values are tuned by PSO. PSO is simple and inexpensive computational effort compared to other artificial intelligence (AI) methods. The PSO is used to find the optimum values of these forgetting factors in an offline manner using AI to avoid the tedious effort instead of trial and error. Once optimum forgetting factor λ is obtained, the RLS will run online with these determined optimum forgetting factor. SOC is predicted based on Open Circuit Voltage (OCV) while SOH is predicted based on internal resistance. Moreover, in order to evaluate the performance of hybrid RLS-PSO, a hybrid RLS-GA (Single objective GA (SOGA)) which is a more common method and had already used by the author on previous paper is employed [11]. Furthermore, hybrid RLS with multi-objectives GA (MOGA) is also introduced.

In Section II, battery dynamical model, RLS, and problem formulation described. Section III presents optimization methods to calculate values of forgetting factors using PSO, SOGA, and MOGA. Simulation results and discussion are reported in Section IV. Finally, conclusion is drawn in Section V.

II. Modeling and problem formulation

Figure 1 shows an equivalent circuit model using single RC [3]. V_t and I represent the battery terminal voltage and current, respectively. R_0 is the battery internal resistance, R_p is diffusion resistance, and C_p is diffusion capacitance. U_d denotes the voltage drop in the diffusion resistance.

By using a convention that the current is positive when it flows into the battery, the dynamics of the battery model can be expressed in the following discrete time equations.

$$U_d(k) = -a_1 U_d(k-1) + b_0 I(k) + b_1 I(k-1) \quad (1)$$

$$V_t(k) = U_d(k) + OCV(k) \quad (2)$$

where:

$$R_0 = b_0;\ R_p = \left(\frac{b_1 - a_1 b_0}{1 + a_1}\right);\ C_p = \left(\frac{T}{b_1 - a_1 b_0}\right)$$

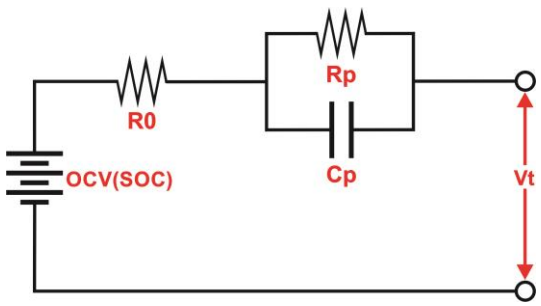


Figure 1. Single RC equivalent circuit model

Terminal voltage and current are measurable, but $U_d(k)$ and $OCV(k)$ can not be measured in real time manner. OCV of the battery is known to be a nonlinear function of its SOC [8]. The internal battery parameters are dependent on SOC and they are time varying in nature.

Terminal voltage estimate $\hat{V}_t(k)$ can be expressed in the following linear equation.

$$\hat{y}_k = \hat{V}_t(k) = \hat{\theta}_k^T x_k \tag{3}$$

where the regressor x_k and the parameter estimates $\hat{\theta}_k$ are given below.

$$x_k = [U_d(k-1); I(k);\ I(k-1);\ 1]$$

$$\hat{\theta}_k = [-a_1(k);\ b_0(k);\ b_1(k);\ OCV(k)]$$

The measured terminal voltage is assumed to follow the following formula.

$$y_k = V_t(k) = \hat{V}_t(k) + e_k \tag{4}$$

The parameter estimates are calculated using RLS with multiple fixed forgetting factors (MFFF-RLS) as follows [18, 19].

$$e_k = y_k - x_k^T \hat{\theta}_{k-1} \tag{5}$$

$$\mathrm{K_{i_k}} = \frac{\mathrm{P_{i_{k-1}} x_{i_k}}}{\lambda_i + \mathrm{x_{i_k}^T P_{i_{k-1}} x_{i_k}}} \tag{6}$$

$$\mathrm{P_{i_k}} = \left(1 - \mathrm{K_{i_k} x_{i_k}^T}\right)\mathrm{P_{i_{k-1}}} \tag{7}$$

$$\mathrm{L_k} = \frac{1}{1 + \frac{\mathrm{P_{1_{k-1}} x_{1_{k-1}}^2}}{\lambda_1} + \cdots + \frac{\mathrm{P_{i_{k-1}} x_{i_{k-1}}^2}}{\lambda_i}} \begin{bmatrix} \frac{\mathrm{P_{1_{k-1}} x_{1_{k-1}}}}{\lambda_1} \\ \vdots \\ \frac{\mathrm{P_{i_{k-1}} x_{i_{k-1}}}}{\lambda_i} \end{bmatrix} \tag{8}$$

$$\hat{\theta}_k = \hat{\theta}_{k-1} + \mathrm{L_k e_k} \tag{9}$$

where subscript i indicates the scalar components $i = 1, 2 \ldots n$. For the battery model addressed in this paper $n = 4$. λ_i denotes forgetting factor. By assuming that OCV changes faster than the internal parameters, it is reasonable to select different values of forgetting factors among them.

A computer script code (m file in Matlab®) has been built to realize the MFFF-RLS algorithm according to the above description and formulae. The following performance index is used to evaluate the MFFF-RLS algorithm.

$$J_0 = \frac{1}{N_s}\sum_{k=1}^{N_s}\left\{V_t(k) - \hat{V}_t(k)\right\}^2 \tag{10}$$

SOC estimation is optimized using performance index J_1, while SOH estimation is optimized by performance index J_2 as follows.

$$J_1 = \frac{1}{N_s}\sum_{k=1}^{N_s}(OCV^*(k) - OCV(k))^2 \tag{11}$$

$$J_2 = \frac{1}{N_s}\sum_{k=1}^{N_s}(R_0^*(k) - R_0(k))^2 \tag{12}$$

OCV^* and R_0^* represent true values of OCV and internal resistance, respectively.

The problem of determining optimum forgetting factor values is formulated as follows.

$$\left.\begin{array}{c} Minimize\colon J_1(\lambda_i) \\ Minimize\colon J_2(\lambda_i) \\ Where\colon \\ 0 < \lambda_i < 1 \\ I(k)\ is\ generated\ by\ UDDS \end{array}\right\} \tag{13}$$

III. Optimization methods using PSO and GA

The optimization problem is solved using particle swarm optimization (PSO) and genetic algorithm (GA). Figure 2 shows the block diagram of the optimization method proposed in this paper. Three methods are elaborated i.e. optimization based on PSO (method 1), optimization based on SOGA (method 2), and optimization based on MOGA (method 3). Their results are analyzed and compared.

PSO is a kind of evolutionary computation techniques which resembles the social behaviour of fish schooling or bird flocking. Its basic conceptual framework was originally proposed in 1995 for optimization of continuous nonlinear functions [20]. The term swarm was selected because it articulated well five basic principles of swarm intelligence in artificial life, those are the proximity principle, the quality principle, the principle of diverse response, the principle of stability, and the principle of adaptability. It involves cooperation and competition among individuals throughout generations. Each individual remembers the best position which had found, and the information of the global best position that an individual had found was shared to all members. Since then it has been experiencing various developments [21, 22].

In PSO, a particle represents a solution, and a swarm of particles is referred to as population of solutions. Each particle is characterized by its velocity and position. Every time a new position is achieved the best positions and velocities are updated. Each particle adjusts its velocity based on its experiences. The following equations are used in PSO to find optimum values of forgetting factors.

$$\lambda_0^i = \lambda_{min} + Rand(\lambda_{max} - \lambda_{min}) \tag{14}$$

$$v_0^i = \frac{\lambda_0^i}{t_s} \tag{15}$$

$$v_{k+1}^i = w v_k^i + c_1 Rand\left(\frac{p^i - \lambda_k^i}{t_s}\right) + c_2 Rand\left(\frac{p_k^g - \lambda_k^i}{t_s}\right) \tag{16}$$

$$\lambda_{k+1}^i = \lambda_k^i + v_{k+1}^i t_s \tag{17}$$

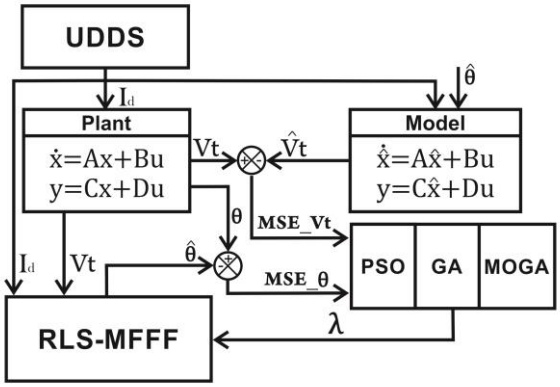


Figure 2. The optimization method of forgetting factors values

λ_k^i and v_k^i represent the i^{th} particle at time k of the positions and velocities, respectively. The upper and lower bounds on the positions are denoted by λ_{max} and λ_{min}. *Rand* is a uniformly distributed random variable whose value is between 0 and 1. t_s denotes a positive scalar. The initial positions λ_0^i and initial velocities v_0^i are randomly generated by Equation (14) and (15). For the next iteration, velocities of each particle is given by Equation (16). p^i is the best positions of each particle over time in current and all previous moves. p_k^g is the best global positions of a certain particle in the current swarm with respect to a predefined fitness function. The new search direction incorporates three pieces of information which have each own weight factor. The first part is current motion which is multiplied by its inertia factor w. The second part is particle memory influence which is multiplied by its cognitive factor c_1, and the third part is swarmed influence which is multiplied by its social factor c_2. Position update of each particle is given by Equation (17).

In order to minimize mean square error values of open circuit voltage and internal resistance, the following fitness function is used.

$$\mathrm{F_t} = \alpha \mathrm{F_1} + (1-\alpha)\mathrm{F_2} \tag{18}$$

where

$$\mathrm{F_1} = \frac{1}{\mathrm{N_s}} \sum_{\mathrm{k=1}}^{\mathrm{N_s}} \left(1 - \frac{\mathrm{OCV(k)}}{\mathrm{OCV^*(k)}}\right)^2 \tag{19}$$

$$\mathrm{F_2} = \frac{1}{\mathrm{N_s}} \sum_{\mathrm{k=1}}^{\mathrm{N_s}} \left(1 - \frac{\mathrm{R_0(k)}}{\mathrm{R_0^*(k)}}\right)^2 \tag{20}$$

$$0 < \alpha < 1 \tag{21}$$

By normalizing performance indexes in Equation (11) and (12), their corresponding dimensionless fitness functions are obtained in Equation (19) and (20). The total fitness function in Equation (18) is a sum of the weighted normalized fitness functions. Values of the weight α are listed in Table 1.

Genetic Algorithm (GA) is an evolutionary algorithm which imitates evolution of living creature. Many variants of GAs exists depending on evaluation method of new chromosomes, a calculation method using serial or parallel processors, combination with some local optimization algorithms (hill climbing, etc), and other factors [23].

A computer code script (m file in Matlab®) has been built to realize a GA according to the following procedure: First, define parameter values including number of initial population/chromosomes N_{ip}, number of genes in a chromosome is 4, boundary value of each gene ($0 < \lambda_i < 1$), and number of bits for each genotype to construct phenotype N_b.Second, define probability rate values including selection probability rate P_s, crossover probability rate P_c, and mutation probability rate P_m. Each probability rate is divided into three sets which are generated randomly, namely small (random value from 0.1 to 0.3), medium (random value from 0.4 to 0.6), and large (random value from 0.7 to 0.9). Thus, there exist 27 sets of probability rate values which yield 27 best chromosomes from 27 different evolutions. Third, create initial random chromosomes. Fourth, evaluate fitness of each chromosome using fitness function in Equation (18), and select best individuals using ranking method. Fifth, create mating pool and generate offsprings by applying a single point crossover. Sixth, reproduce and ignore few chromosomes. Seventh, performs mutation by bit flipping operation randomly according to the mutation probability rate. Elitism principle is used to control mutation. Finally, back to step 4, until termination criterion is achieved.

Method 1 and method 2 above are used to solve the single objective function in Equation (18). In order to solve the original multiple objectives optimization problem described in the problem formulation at the previous section, multiple objectives GA (MOGA) is also implemented. A fast elitist multiobjective GA known as nondominated sorting genetic algorithm II (NSGAII) is used to solve this problem since this algorithm has three advantages, i.e. a fast non-dominated sorting procedure, a fast crowded distance estimation process, and a simple crowded comparison operator. The main loop of the NSGA II procedure is described below [24]. First, combine parent and offspring population and saved as R_t. Second, execute the fast non-dominated sorting procedure against R_t, and save the result of all non-dominated fronts of R_t into $F = (F_1, F_2, \cdots)$. Third, set initial values of parent population $P_{t+1} = 0$, and generation counter $i = 1$. Fourth, run iteration of generation until the parent population is filled and $|P_{t+1}| + |F_i| \leq N$. Execute the crowded distance estimator in F_i, include i-th non-dominated front in the parent population, then check the next front for inclusion $i = i + 1$. Fifth, sort F_i in

Table 1.
Weight of finess function

No	1	2	3	4	5	6	7	8	9
α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

descending order using the crowded comparison operator. Sixth, choose the first $(N - |P_{t+1}|)$ elements of F_i and include them into the parent population. Seventh, use selection, crossover, and mutation to create offspring Q_{t+1}. Finally, increment the generation counter $t = t + 1$. More details about the algorithm can be seen in [24].

IV. Results and discussion

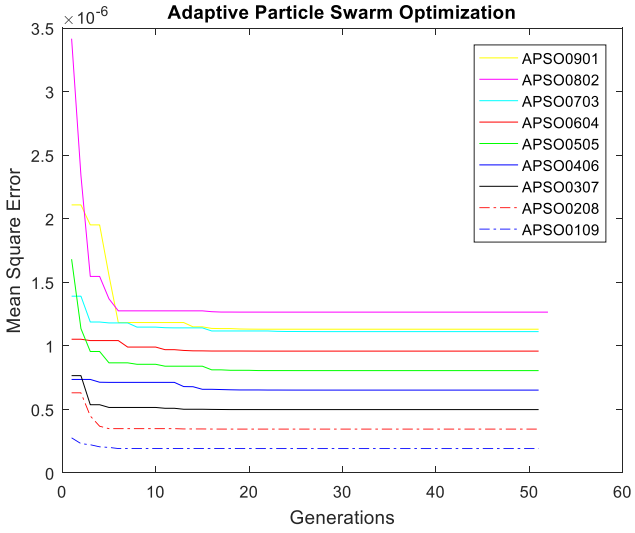
In order to validate the proposed method, computer simulation has been conducted. The swarm size in PSO and initial population in GA is set to 64. The population size is chosen based on the crossover operation in GA, it is easier to choose a 2^n number. Larger n needs more calculation time each iteration but yields smaller number of generation. Based on this consideration we choose n=6. For the sake of equality and comparability, the swarm size in PSO is chosen the same number.

The optimization is executed iteratively until a termination criterion is achieved. Fitness function tolerance is set to $10e^{-6}$ while stall iteration is set to 50. For method 1, the cognitive factor and social factor are set $c_1 = 1.49$ and $c_2 = 1.49$. In order to maintain the speed of convergence while avoiding local optima, the inertia factor is changed linearly with iteration counter k as follows.

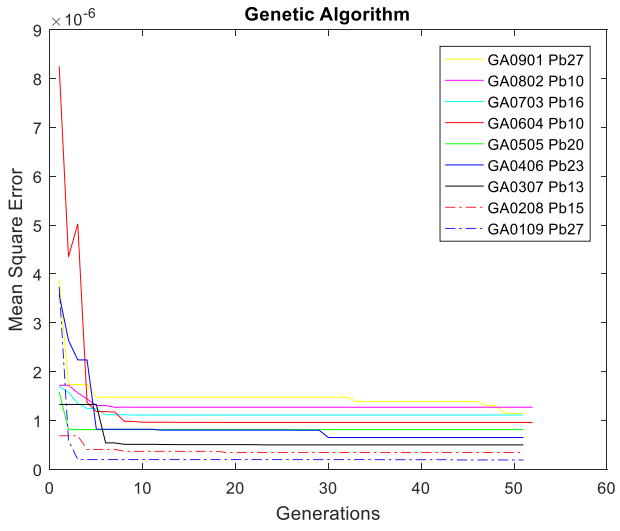
$$w = w_i - \frac{(w_i - w_f)}{N} k \tag{22}$$

In this simulation, parameter values related to inertia factors are set as follows: $w_i = 1.1$, $w_f = 0.1$, and $N = 50$.

Figure 3 shows trajectories of fitness function F_t as a function of generation for 9 different weight values in Table 1. Figure 3(a) plots the results of method 1 while Figure 3(b) those of method 2. In method 2, every single weight produces 27 sets of solutions according to the values of selection, crossover, and



(a)



(b)

Figure 3. Trajectories of fitness function F_t; (a) PSO; (b) SOGA

mutation probability rates. The best solution is selected among 27 choices. Therefore, in Figure 3(b) we have 9 curves of the best-selected solutions. It is obvious that the value of weight affects the fitness function value significantly. The best result of method 1 and method 2 in Figure 3 are plotted together in Figure 4.

From Figure 4, some important results can be summarized as follows: First, the SOGA and PSO provide similar performance index values at the end of generation (after 52 iterations). Second, at the 3rd and 4th generation, SOGA provides better performance than PSO. Third, the 5th generation, SOGA and PSO provide similar performance.

Fourth, at the 6th generation, PSO gives better performance than SOGA, and this condition remains until the 43rd generation. During this condition, the performance difference is around 10^{-8} this implies that PSO provides better performance than SOGA in terms of less generation number.

Depending on the engineering problem solved, a performance difference of 10^{-8} may be considered as substantially small, so that one may argue that SOGA and PSO have the same capability for solving optimization problem such as this paper. However, in this paper, the cognitive and social factor values of PSO are fixed. Investigation of the impact of different cognitive and social factor on the performance is left for further study.

Figure 5 shows the Pareto front obtained by NSGA II. From this result, it can be seen that NSGA II provides several optimal solutions of the original multi-objectives optimization problem stated in Equation (13). In other words, this implies that NSGA II leaves the final decision to us to select a solution. In this paper, a solution is selected which gives the similar performance of fitness functions F_1 and F_2 from PSO and SOGA above. Thus, $F_1 = 1.5733e-6$ and $F_2 = 1.3829e-6$.

In respect to the time consumed or a number of generation during iteration, the following results are obtained: First, PSO requires a smaller number of generation to yields better MSE performance than SOGA. Second, MOGA requires much longer time than PSO and SOGA because it computes Pareto front containing several numbers of optimum solutions.

Table 2.
Forgetting factors obtained through optimization

Method	λ_1	λ_2	λ_3	λ_4
PSO	0.9298	0.0101	0.7171	0.2316
SOGA	0.9395	0.0508	0.7489	0.2692
NSGA II	0.9365	0.9185	0.8148	0.3062

Table 3.
Performance index value

No	Performance Index	Values		
		PSO	SOGA	NSGAII
1	J_0	2.0574e-08	2.1339e-08	2.2961e-08
2	J_1	2.4773e-05	2.4912e-05	2.4339e-05
3	J_2	1.1559e-11	1.1559e-11	4.1533e-10

Table 2 lists up the forgetting factors obtained by PSO, SOGA, and NSGA II. These forgetting factors are used together with MFFF-RLS to estimate battery terminal voltage, OCV, SOC, and internal resistance R_0.

Figure 6 shows battery terminal voltage and its estimation error during the UDDS testing using the forgetting factors in Table 2. Red line is the results of PSO, the blue line is the results of SOGA, and the green line is the results of NSGA II. Figure 7 shows the corresponding OCV while Figure 8 shows the corresponding SOC and its estimation error. Figure 9 shows time history of internal resistance estimate $\hat{R}_0(k)$ and its error $e\hat{R}_0(k) = R_0(k) - \hat{R}_0(k)$.

Table 3 lists performance index values obtained from these results. As expected PSO, SOGA and NSGA II give similar performances in terms of mean square error. However, PSO and MOGA provide a little better performance than SOGA in terms of OCV MSE value.

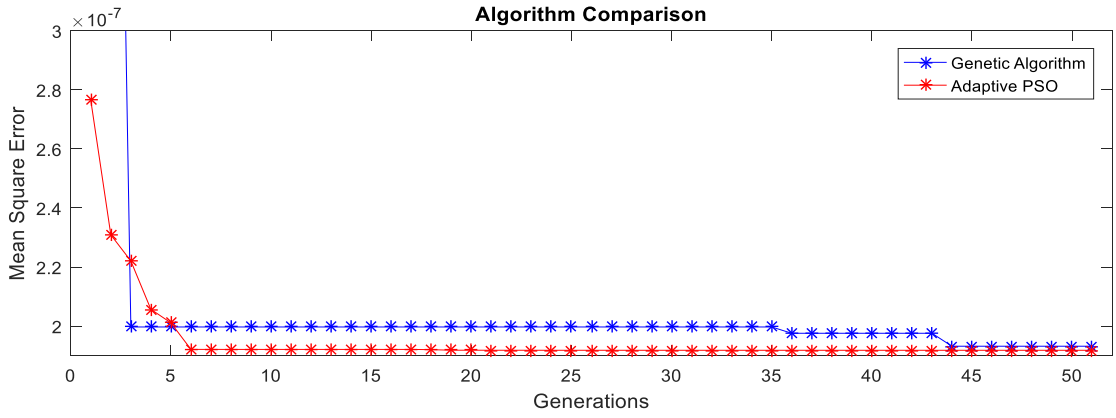


Figure 4. The best performance index F_t of PSO and SOGA

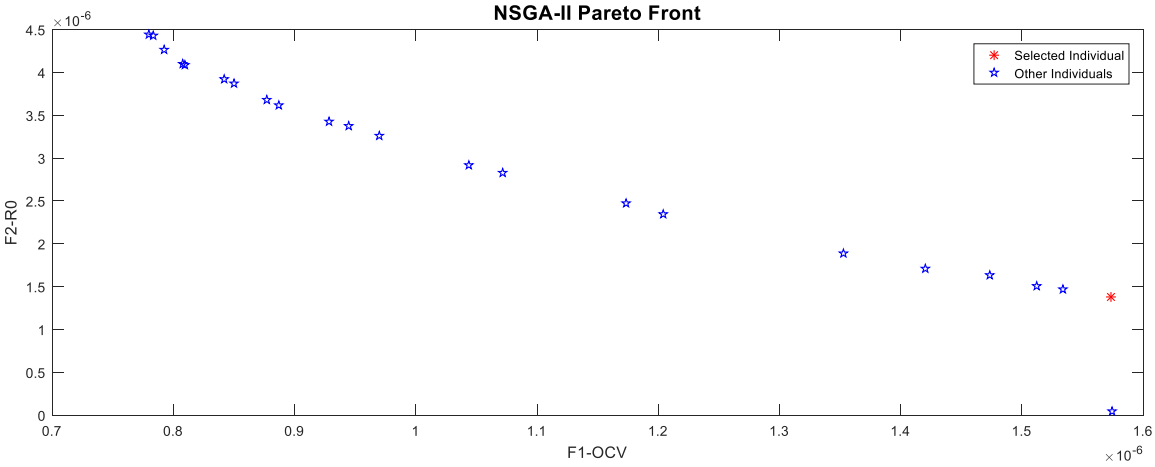
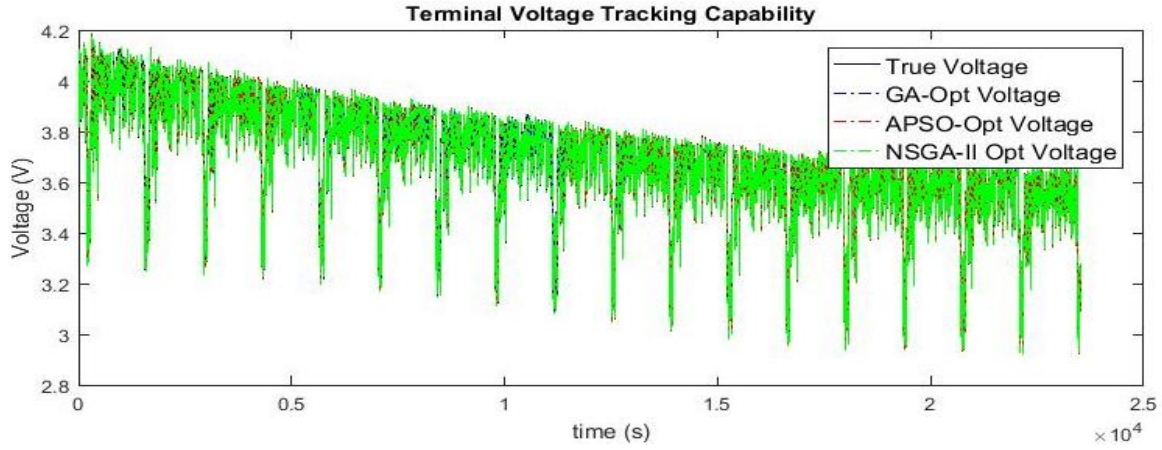
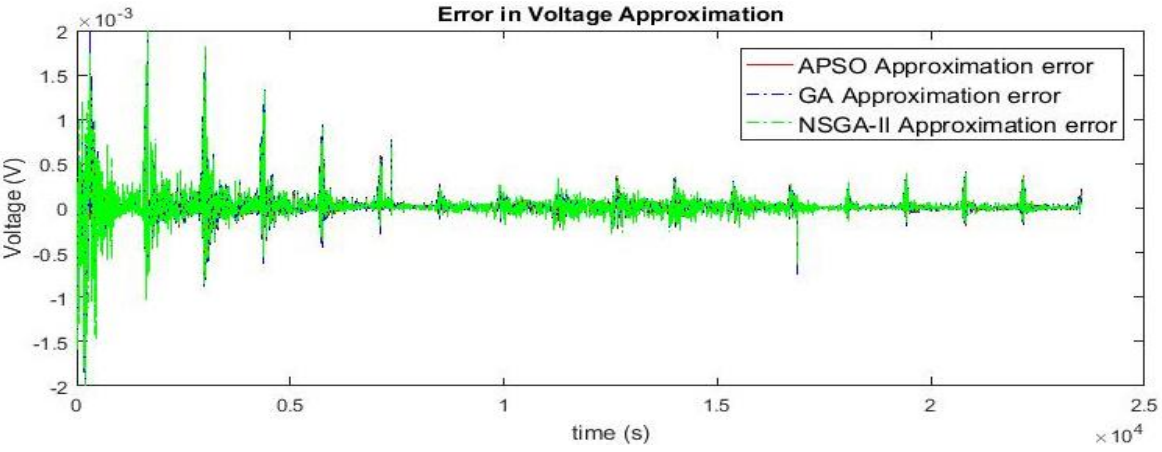


Figure 5. Pareto front of NSGA II



(a)



(b)

Figure 6. Tracking performance of various methods; (a) Terminal voltage; (b) estimation error

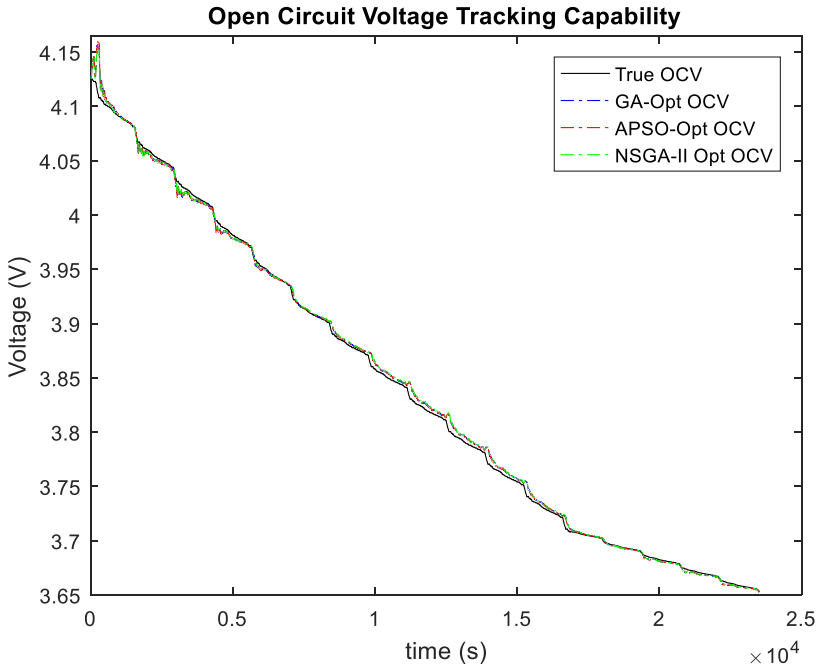
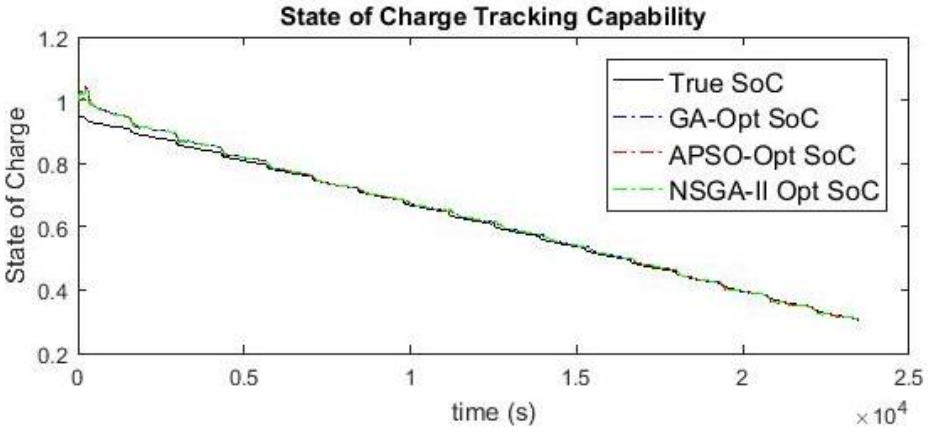
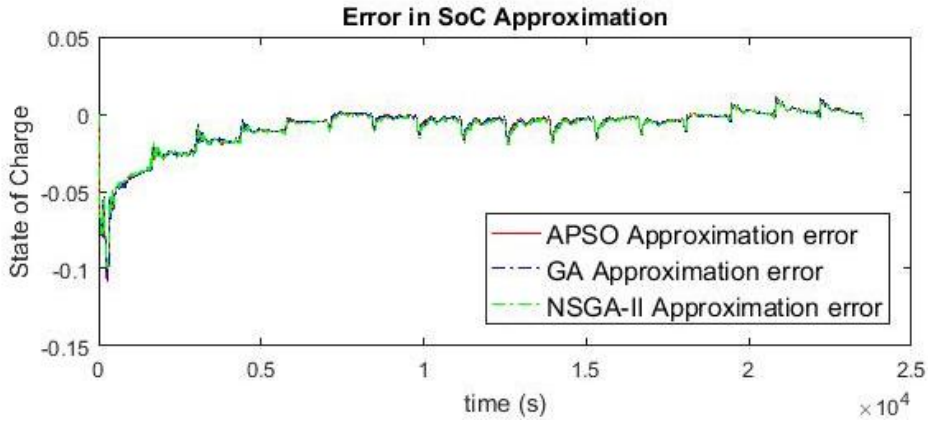


Figure 7. Open circuit voltage

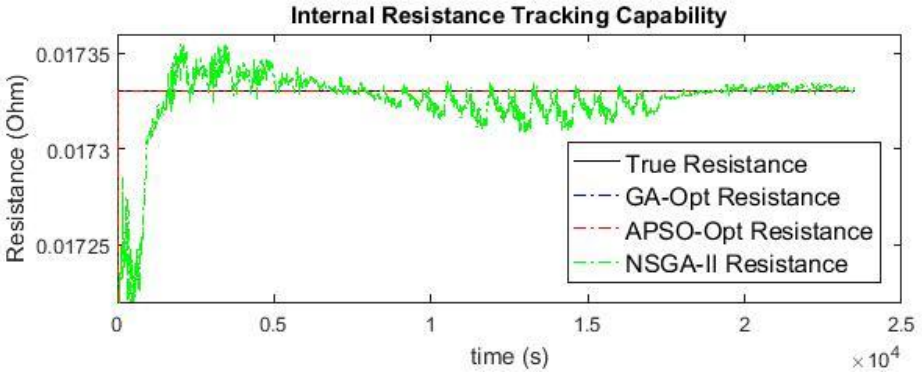


(a)

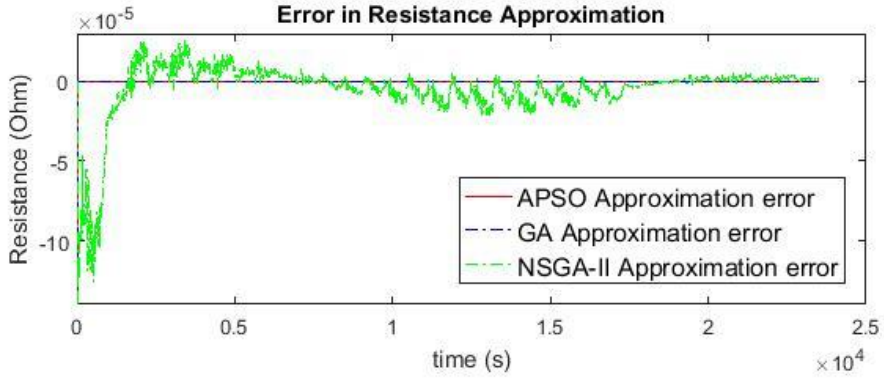


(b)

Figure 8. Tracking performance of various methods; (a) Time history of state of charge; (b) SoC error



(a)



(b)

Figure 9. Tracking performance of various methods (a) Internal resistance; (b) Error in internal resistance

V. Conclusions

From the computer simulation results, the following conclusion can be drawn. By selecting proper probability rates of selection, crossover, and mutation, SOGA was able to produce almost similar performance with PSO in terms of MSE. Considering the number of generation, PSO provides better performance than SOGA in terms of less generation number. MOGA provides Pareto fronts containing optimum solutions where a specific solution can be selected to have MSE performance as good as PSO. However, the MOGA requires much longer time than PSO and SOGA because it computes Pareto fronts containing several numbers of optimum solutions.

Acknowledgement

The authors thank to the Indonesian Institute of Sciences (LIPI) for providing financial support in the scheme of excellent research programme with the contract number 1975.3/D3/PG/2016 of the financial year of 2016. They also deliver gratitude to the Ministry of Science, Technology, and Higher Education of the Republic of Indonesia in providing financial support for conducting individualized immersion programme at Centre for Mobility & Transport, Coventry University, United Kingdom in 2016.

References

[1] F. Codeca *et al.*, "On battery state of charge estimation: A new mixed algorithm," in *Proc. IEEE Int. Conf. Control Appl.*, pp.102-107, 2008.

[2] F. Codeca *et al.*, "The mix estimation algorithm for battery State-of-Charge estimator – Analysis of the sensitivity to measurement errors," pp.8083-8088, 2009.

[3] A. Nugroho *et al.*, "Battery state of charge estimation by using a combination of Coulomb Counting and dynamic model with adjusted gain," in *International Conference on Sustainable Energy Engineering and Application (ICSEEA)*, pp.54-58, 2015.

[4] J. Xu *et al.*, "The state of charge estimation of lithium-ion batteries based on a proportional-integral observer," *IEEE Trans. Veh. Technol.,* vol. 63, no. 4, pp. 1614-1621, 2014.

[5] R. Xiong *et al.*, "A robust state-of-charge estimator for multiple types of lithium-ion batteries using adaptive extended Kalman filter," *J. Power Sources,* vol. 243, pp. 805-816, 2013.

[6] D. Li *et al.*, "State of charge estimation for LiMn2O4 power battery based on strong tracking sigma point Kalman filter," *J. Power Sources,* vol. 279, pp. 439-449, 2015.

[7] X. Chen *et al.*, "A novel approach for state of charge estimation based on adaptive switching gain sliding mode observer in electric vehicles," *J. Power Sources,* vol. 246, p. 667–678, 2014.

[8] X. Hu *et al.*, "Online Estimation of an Electric Vehicle Lithium-Ion Battery Using Recursive Least Squares with Forgetting," 2011.

[9] H. He *et al.*, "Online model-based estimation of state-of-charge and open-circuit voltage of lithium-ion batteries in electric vehicles," *Energy,* vol. 39, no. 1, p. 310–318, 2012.

[10] H. R. Eichi and M. Chow, "Adaptive parameter identification and State-of-Charge estimation of lithium-ion batteries,"

IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 4012-4017, 2012.

[11] L. Rozaqie and E. Rijanto, "SOC Estimation for Li-Ion Battery using Optimum RLS Method Based on Genetic Algorithm, "*Proc. International Conference on Information Technology, Electrical Engineering*, Date of Conference Oct, 2016. Added to IEEE Xplore 28 February 2017.

[12] M. N. Ramadan *et al*., "Comparative Study Between Internal Ohmic Resistance and Capacity for Battery State of Health Estimation," *Journal of Mechatronics, Electrical Power, and Vehicular Technology,* vol. 6, no. 2, pp. 113-122, 2015.

[13] V. Surendar *et al*., "Estimation of State of Charge of a Lead Acid Battery Using Support Vector Regression," *Procedia Technology 21,* pp. 264-270, 2015.

[14] J. C. Anton *et al*., "A new predictive model for the state of charge of a high power lithium-ion cell based on a PSO optimized multivariate adaptive regression splines approach," *IEEE Transactions on Vehicular Technology ,* 2015.

[15] B. Wahono *et al*., "Prediction Model of Battery State of Charge and Control Parameter Optimization for Electric Vehicle," *Journal of Mechatronics, Electrical Power, and Vehicular Technology,* vol. 6, no. 1, pp. 31-38, 2015.

[16] G. Walder *et al*., "Adaptive State and Parameter Estimation of Lithium-Ion Batteries Based on a Dual Linear Kalman Filter," *Proceeding of Second International Conference on The Society of Diital Information and Wireless Communications (SDIWC),* , 2014.

[17] T. Kim *et al*., "Online State of Charge and Electrical Impidance Estimation for Multicell Lithium-ion Batteries," *IEEE Transportation Electrification Conference and Expo (ITEC),* pp. 1-6, 2013.

[18] A. Vahidi *et al*., "Simultaneous mass and time-varying grade estimation for heavy-duty vehicles," *Proc. 2003 American Control Conference,* vol. 6, p. 4951–4956, 2003.

[19] A. Vahidi, *et al*., "Recursive least squares with forgetting for online estimation of vehicle mass and road grade: theory and experiments," *Veh. Syst. Dyn.,* vol. 43, no. 1, pp. 31-55, 2005.

[20] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Ntworks,* pp. 1942-1945, 1995.

[21] J. Yisu *et al*., "The landscape adaptive particle swarm optimizer," *Applied Soft Computing,* vol. 8, pp. 295-304, 2008.

[22] A. Sahu *et al*., "Fast Convergence Particle Swarm Optimization for Functions Optimization," *Procedia Technology,* vol. 4, pp. 319-324, 2012.

[23] A. Munawar *et al*., "A survey: Genetic algorithms and the fast evolving world of parallel computing," in *Proc. - 10th IEEE Int. Conf. High Perform. Comput. Commun. HPCC 2008, pp. 897–902*, 2008.

[24] K. Deb *et al*., "A fast and elitist multiobjective genetic algorithm: NSGA II," *IEEE Transactions on Evolutionay Computation,* vol. 6, no. 2, pp. 182-197, 2002.