Parameter estimation of photovoltaic model via parallel particle swarm optimization algorithm

Jieming Ma1,2,*†, Ka Lok Man2, Sheng-Uei Guan2, T. O. Ting3 and Prudence W. H. Wong4

1School of Electronic and Information Engineering, Suzhou University of Science and Technology, No. 1 Ke Rui Road, Suzhou High-Tech Zone, Suzhou, Jiangsu Province, 215009, China
2Department of Computer Science and Software Engineering (CSSE), Xi’an Jiaotong-Liverpool University, Science Building, No. 111 Ren’ai Road, Suzhou Industrial Park, Suzhou, Jiangsu Province, 215123, China
3Department of Electrical and Electronic Engineering (EEE), Xi’an Jiaotong-Liverpool University, Science Building, No. 111 Ren’ai Road, Suzhou Industrial Park, Suzhou, Jiangsu Province, 215123, China
4Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool, L69 3BX, UK

SUMMARY

Recently, bio-inspired metaheuristic algorithms have been widely used as powerful optimization tools to estimate crucial parameters of photovoltaic (PV) models. However, the computational cost involved in terms of the time increases as data size or the complexity of the applied PV electrical model increases. Hence, to overcome these limitations, this paper presents the parallel particle swarm optimization (PPSO) algorithm implemented in Open Computing Language (OpenCL) to solve the parameter estimation problem for a wide range of PV models. Experimental and simulation results demonstrate that the PPSO algorithm not only has the capability of obtaining all the parameters with extremely high accuracy but also dramatically improves the computational speed. This is possible and is shown in this work via the inherent capabilities of the parallel processing framework. Copyright © 2015 John Wiley & Sons, Ltd.

KEY WORDS

photovoltaic cells; modeling; parameter estimation; parallel algorithms; solar energy

Correspondence

*Jieming Ma, School of Electronic and Information Engineering, Suzhou University of Science and Technology, No. 1 Ke Rui Road, Suzhou High-Tech Zone, Suzhou, Jiangsu Province, 215009, China.
†E-mail: jieming84@gmail.com

Received 15 November 2014; Revised 21 March 2015; Accepted 6 May 2015

1. INTRODUCTION

The progress in photovoltaic (PV) research and development has been improving rapidly over the past years. The prominent features of solar energy – clean, abundant, and renewable – make the PV generation popular and promising in various industrial applications [1].

Since the initial silicon PV cell was developed by using the single crystal, varieties of silicon materials have been used to develop PV cells. For example, polycrystalline and amorphous silicon cells were designed to be less energy intensive. Thin silicon cells make a compromise between crystalline and amorphous cells and were reported to achieve better efficiency and stability [2]. With numerous PV cells made of various semiconductor materials using different manufacturing processes, a general performance estimation tool, known as PV electrical model, is crucial to predict the electrical characteristics of these cells before installation. Unfortunately, PV electrical model cannot be directly used because of the lack of proper model parameters characterizing PV cells. The term parameter estimation refers to the process of using sample data to estimate parameters of the selected PV electrical model [3]. With the parameters obtained in such a way, the differences between simulated and experimental data can be minimized considerably.

In the literature [4], conventional parameter estimation methods are classified into two categories:

i. analytical technique [5–8] represents model parameters mathematically by a series of equations;
ii. numerical technique [9–11] extracts parameters utilizing numerical methods to minimize the error of the applied model.

Feasible as they are, both of them have inevitable defects. The former method addresses the parameter estimation problem by analytical expressions in terms of the
key points on the PV current–voltage (I–V) curve (e.g., the maximum power point, short-circuit current \( I_{sc} \), and open-circuit voltage \( V_{oc} \), etc.). Its errors can be significant and cannot be further improved if these fundamental elements are incorrectly specified. Numerical parameter extraction is normally considered as an accurate approach in parameter estimation as all the measured data are used in the calculation. It is axiomatic that its performance depends on the type of fitting algorithm, the cost function, and the initial values of parameters to be extracted [9]. Moreover, many numerical algorithms can be computationally expensive as the size of the required data is relatively large.

More recently bio-inspired metaheuristic algorithms, such as genetic algorithm [12], particle swarm optimization [13,14], bacterial foraging algorithm [15], pattern search (PS) [16], simulated annealing (SA) [17], differential evolution [18,19], and cuckoo search [20] have been proposed to determine the values of PV-model parameters. Albeit accurate, most of these methods apply multiple agents or particles in random search and do not facilitate a meaningful improvement in computational efficiency. Today’s programming environments, such as Open Computing Language (OpenCL), are more multifaceted, and they enable an algorithm to be executed in a wide range of central processing units (CPUs), digital signal processors, field programmable gate arrays, and graphic processing units (GPUs) [21]. These programming environments or application programming interfaces exploit the capabilities of computing devices using the languages that only require the highest-level descriptions of parallel process management [22].

With the aim of distributing the workload of a parameter estimation algorithm appropriately to computing devices in parallel mode, this paper presents a form of computation in which the PSO-based parameter estimation algorithm is carried out simultaneously. It is desirable that the parallel particle swarm optimization (PPSO) outperforms the sequential particle swarm optimization (SPSO) in two aspects:

i. the computational speed tends to be faster than the SPSO with the same amount of work load;
ii. more computational units can be utilized in optimization, and thus, it is scalable.

The accuracy and computational efficiency of the proposed method are evaluated by identifying the parameters of the two most widely applicable PV electrical models. The remainder of this paper is organized as follows. The next section briefly illustrates PV electrical models. This is followed by the problem formulation in Section 3. Section 4 elaborates on the SPSO, followed by the implementation of the proposed parallel method, the PPSO. Extensive simulation is run on CPUs and GPUs, and the obtained results are discussed in Section 5. Finally, Section 6 derives conclusions with some proposed insights for future work.

2. MATHEMATICAL MODELING OF PHOTOVOLTAIC DEVICES

2.1. Single-diode model

As briefly discussed in the introduction, PV system designers are usually interested in modeling PV devices as mapping of the electrical characteristics is of significance to the understanding, optimization, and development of PV power harvesting systems.

The elementary PV device is a PV cell, which is basically a semiconductor diode that generates a reverse current when its \( p–n \) junction is exposed to light. This reverse current is termed as photocurrent \( I_{ph} \). In darkness, the PV cell behaves like a diode, and thus, its dark \( I–V \) characteristics are usually mathematically expressed by Shockley diode equation [23]:

\[
I_D = I_{ph} \left( e^{\frac{V}{V_t}} - 1 \right),
\]

where \( V_D \) represents the electrical potential difference between the two ends of the diode, \( I_{ph} \) denotes the reverse saturation current, and \( A_I \) is the diode ideality factor. \( V_t \) is known as thermal voltage, and its value can be estimated as a function of temperature \( T \), namely, \( V_t = kT/q \), where \( k \) and \( q \) represent the Boltzmann constant \((1.380650 \times 10^{-23} \text{ J/K})\) and the electron charge \((1.602176 \times 10^{-19} \text{ C})\), respectively. Assume that the superposition principle holds, the full \( I–V \) characteristic is simply the sum of the dark and illuminated \( I–V \) characteristics:

\[
I = I_{ph} - I_{ph} \left( e^{\frac{V}{V_t}} - 1 \right).
\]

In the literature [24] [25], 2 is also the mathematical expression of an ideal PV model, in which the \( I_{ph} \) is modeled as a current source. As reported by the authors in [23] and in [26], the output current \( I \) is dependent on the resistances of \( p \) and \( n \) bodies, the contact resistance of the \( n \) layer with the top metal grid, the resistance of the grid, and the contact resistances of the metal base with the \( p \) semiconductor layer, as well as the leakage current of the \( p–n \) junction. These losses are roughly represented by series resistance \( R_s \) and shunt resistance \( R_p \) in the single-diode model, whose circuitry diagram is shown in Figure 1(a). The corresponding equivalent circuit equation is expressed in 3 [27]:

\[
I = I_{ph} - I_{ph} \left( e^{\frac{V+IR_p}{V_t}} - 1 \right) - \frac{V + IR_p}{R_p}.
\]

2.2. Double-diode model

The dark characteristics of PV cells have been intensively studied by many researchers. A simple approach of improving the single-diode model is to model the junction...
recombination, which can be achieved by adding a second diode in parallel with the first one [28]. Figure 1(b) shows the circuitry diagram of the double-diode model, and its mathematical model equation is as follows:

\[
I = I_{ph} - I_{o1} \left( e^{\frac{V + IR_s}{N_s V_t}} - 1 \right) - I_{o2} \left( e^{\frac{V + IR_s}{N_s V_t}} - 1 \right) - \frac{V + IR_s}{R_p},
\]

(4)

where \( I_{o1} \) and \( I_{o2} \) are the reverse saturation currents of the first and second diode, respectively. Similarly, the second diode’s ideality constant is denoted by \( A_2 \).

### 2.3. Photovoltaic module model

In a large PV generation system, PV modules are used as basic components rather than PV cells because the output power of a PV cell is limited at high voltage levels. Researchers develop the PV module model so as to predict the \( I-V \) characteristics before modeling the whole system.

Because the PV module is a packaged, connected assembly of \( N_s \) PV cells, its output voltage and resistance are scaled in accordance with the following rules:

\[
\begin{align*}
V' &= N_s V, \\
R_s' &= N_s R_s, \\
R_p' &= N_s R_p,
\end{align*}
\]

(5)

where \( V' \) and \( R' \) represent the voltage and series resistance of the PV module, respectively.

For the convenience of description, the models discussed in this paper are abbreviated as follows: (i) SDC, single-diode cell; (ii) DDC, double-diode cell; (iii) SDM, single-diode model; and (iv) DDM, double-diode model.

### 3. PROBLEM FORMULATION

Based on an optimization algorithm, the parameter estimation method minimizes the differences between calculated and measured data by adjusting PV parameters [20]. After importing several parameters, the parameter estimation algorithm starts evaluating possible solutions by using the objective function with the measured \( I-V \) data. In general, the objective function is formulated by the root mean square (RMS) error \( f_{rms} \) which serves to aggregate absolute differences into a single measure of predictive power. If the number of experimental data is denoted by \( N \), the RMS error can be mathematically described by the following equation:

\[
f_{rms} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left( f_D(V, I, X) \right)^2},
\]

(6)

where \( V \) and \( I \) denote the measured voltage and current, respectively, \( f_D(V, I, X) \) is the objective function for the \( d^{th} \) data, and \( X \) is a vector representing the model parameters. Take the SDC for an example, \( f_D(V, I, X) \) is a homogeneous form of 3, namely,

\[
f_D(V, I, X) = I_{ph} - I_{o1} \left( e^{\frac{V + IR_s}{N_s V_t}} - 1 \right) - \frac{V + IR_s}{R_p} - I,
\]

(7)

where \( X \) is a vector involving the model parameters \( I_{ph,1}, I_{o1,1}, A_1, R_s, \) and \( R_p \).

### 4. PARAMETER ESTIMATION ALGORITHM

#### 4.1. Sequential particle swarm optimization

By mimicking the swarm behavior of fishes and birds, Kennedy and Eberhart [29] developed a nature-inspired metaheuristic algorithm in 1995. This derivative-free method is particularly suited for continuous variable problems and has been successfully applied to many engineering optimization problems. In [30], Kennedy et al. implemented the algorithm in a procedural C-program. We call it SPSO in this paper.

The basic idea behind the SPSO is to search a space by adjusting the trajectories of particles, which represent possible solutions of the objective function. The pseudocode depicting the SPSO is shown in Algorithm 1. Assume that the swarm size is \( P \) and the problem dimension is \( D \). The \( i^{th} \) (\( i = 1, 2, \ldots, P \)) and \( j^{th} \) (\( j = 1, 2, \ldots, D \)) dimension is denoted by \( x_{i,j} \). Similarly, the \( i^{th} \) velocity in \( j^{th} \) dimension is \( v_{i,j} \).

The SPSO firstly initializes the algorithm parameters (e.g., inertia weight, learning parameters, etc.) as well as the velocity and position of each particle. In an iteration \( t (t = 1, 2, \ldots, t_{max}) \), the fitness of particles are evaluated individually by its objective function. When...
a particle \( i \) arrives a location that is better than any positions it arrived, it records the new position as local best position \( p_{besti} \). In a swarm of particles, there are \( P \) local best positions. Among them, the one with the best solution is termed as global best position \( g_{best} \) in the literature. Kennedy and Eberhart proposed that the movements of particles are mainly attracted toward the \( p_{besti} \) and \( g_{best} \), and the new position of a particle in iteration \( t+1 \) can be mathematically expressed in the following manner:

\[
x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1},
\]

where \( v_{ij}^{t+1} \) is the velocity, expressed as

\[
v_{ij}^{t+1} = w v_{ij}^t + a \epsilon_1 \left( x_{ij}^t - g_{best} \right) + \beta \epsilon_2 \left( x_{ij}^t - p_{besti} \right).
\]

In 9, the notations \( a \) and \( \beta \) are the learning parameters. Typically, \( a \approx \beta \approx 2 \). The two random vectors \( \epsilon_1 \) and \( \epsilon_2 \) are in the range between 0 and 1. The inertia weight \( w \) is used to balance global and local search abilities. It can be taken either as a constant from 0.5 to 0.9 for simplicity or a linear function in terms of iteration \( t \). In this paper, the value of \( w \) is defined as

\[
w^t = w_{max} - \left( w_{max} - w_{min} \right) \frac{t}{t_{max}},
\]

where \( w_{max} \) and \( w_{min} \) represent the maximum and minimum of the \( w \), respectively.

4.2. Implementation of parallel particle swarm optimization

The workload behaviors can be generally classified into two types: data intensive and control intensive. In fact, there is no best architecture that runs optimally on all types of workloads. According to [21], control-intensive applications tend to run faster on super-scalar CPUs, where significant computing efforts have been devoted to branch prediction mechanisms, while data-intensive applications tend to run fast on vector architectures, where the same operation is applied to multiple data items concurrently.

The structure of SPSO has a mix of the workload characteristics. Consider the fitness evaluation function. In procedural C-program, the RMS errors are computed particle by particle in a for loop. In order to parallelize this function, we choose to generate a separate execution instance to perform fitness evaluation for each particle. Figure 2 depicts the concurrent process. With the measured I-V data, the RMS errors can be calculated concurrently in a kernel, which actually is a piece of code executing tasks on a multi-core processor. The fitness evaluation process for a particle is independent of any other particle, and thereby possesses significant data level parallelism. On the other hand, the function updating the swarm’s velocities and
positions, especially the process checking whether the values exceed the predefined bounds, can be assigned to the category of control-intensive applications because it involves explicit flow-control constructs such as if-then-else. From these considerations, it is desirable that a programming framework with the capability of execution across a wide range of device types so that the workload can be executed most efficiently on a specific style of hardware architecture. The OpenCL, managed by the nonprofit technology consortium Khronos Group, is such a heterogeneous programming framework that supports a wide range of levels in parallelism and efficiently maps to a variety of computing devices [21]. A host and a device-side language are both defined in the OpenCL. The former offers a management layer that supports efficient plumbing of complicated concurrent programs, while the latter maps the heavy work load into a wide range of memory systems.

In our implementation, the main program was written in OpenCL code. Application programming interfaces were used to configure a context which allows commands and data passing to the device. Figure 3 represents the whole algorithmic flow of the proposed PPSO. After initialization, velocities and positions of particles are transferred from the host to the device. In the OpenCL kernel function, we choose to decompose fitness evaluations to perform the evaluations concurrently on a multi-processor device. Global synchronization or barrier function is applied to ensure that all of the fitness evaluations are completed before they are transferred back to the host. The local best and global best positions are aided to determine the new velocities and positions of particles. The algorithm will then return to the parallel process by evaluations through the objective function until the stopping criterion is satisfied.

Figure 3. Flow chart of the parallel particle swarm optimization algorithm (a) the main program (b) parallel evaluations of root mean square (RMS) errors.
5. RESULTS AND DISCUSSIONS

The proposed PPSO was implemented in OpenCL, and simulations were performed under Microsoft’s Windows 7 64-bit operating system. Its algorithm parameters were set as the learning factors $c_1 = c_2 = 2$, the maximum inertia factor $w_{\text{max}} = 0.9$, and the minimum inertia factor $w_{\text{min}} = 0.4$. With the aim of conducting a comprehensive evaluation, both single-diode model and double-diode model were applied in parameter estimation. The experimental $I-V$ data of a 57 mm diameter commercial silicon PV cell (R.T.C. France) and a PV module (Photowatt-PWP 201) comprising 36 polycrystalline silicon PV cells are considered as test examples in this paper. Their values were obtained under the controlled conditions from an automated measuring system with a CBM8096 microcomputer as demonstrated in [10]. It is assumed that all the silicon cells in a PV module are identical and work under the same temperature (R.T.C. France PV cell at 33°C and Photowatt-PWP 201 PV module at 45°C).

Extensive simulation results and statistical analysis are presented in the subsequent sections. Section 5.1 studies the parameter estimation capability by evaluating the evolution performance and distribution of fitness values for the proposed PPSO method. Besides RMS error, the mean absolute error $\bar{e}$ is used to evaluate how close the simulated current values $I$ are to the measured data $\hat{I}$. It is mathematically expressed as

$$\bar{e} = \frac{1}{N} \sum_{d=1}^{N} |I_d - \hat{I}_d|.$$  \hspace{1cm} (11)

In Section 5.2, we demonstrate how the PPSO method outperforms its sequential version in terms of computational speed. Speedup is used to qualify the ratio of sequential execution time to parallel execution time:

$$S = \frac{T_s}{T_p}.$$ \hspace{1cm} (12)

where $T_s$ is the execution time of sequential algorithm on the host processor and $T_p$ is the execution time of parallel algorithm on multi-core devices.

5.1. Parameter estimation capability

Table I shows the estimated parameters for different PV electrical models obtained from the best of 30 runs of the proposed PPSO method, in which the swarm size and maximum iteration number are set to 2048 and 80,000, respectively.

In order to make a comprehensive comparison, the parameters estimated by the other methods, such as least square optimization [10], pattern search (PS) [16], and simulated annealing (SA) algorithms [17], are also listed.

<table>
<thead>
<tr>
<th>PV model</th>
<th>Method</th>
<th>$I_{\text{ph}}$ (A)</th>
<th>$R_s$ (Ω)</th>
<th>$R_p$ (Ω)</th>
<th>$I_{\text{sc}}$ (μA)</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$\bar{e}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDC</td>
<td>PPSO</td>
<td>0.7608</td>
<td>0.0364</td>
<td>53.7185</td>
<td>0.3230</td>
<td>1.4812</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>LSO [10]</td>
<td>0.7608</td>
<td>0.0364</td>
<td>53.7634</td>
<td>0.3223</td>
<td>1.4837</td>
<td>—</td>
<td>6.8384 E-4</td>
</tr>
<tr>
<td></td>
<td>PS [16]</td>
<td>0.7617</td>
<td>0.0313</td>
<td>64.1026</td>
<td>0.9980</td>
<td>1.6000</td>
<td>—</td>
<td>1.4129 E-2</td>
</tr>
<tr>
<td></td>
<td>SA [17]</td>
<td>0.7620</td>
<td>0.0345</td>
<td>43.1034</td>
<td>0.3130</td>
<td>1.5172</td>
<td>—</td>
<td>2.1536 E-3</td>
</tr>
<tr>
<td>SDM</td>
<td>PPSO</td>
<td>1.0305</td>
<td>1.2013</td>
<td>981.9823</td>
<td>3.4823</td>
<td>1.3512</td>
<td>—</td>
<td>1.4277 E-3</td>
</tr>
<tr>
<td>DDC</td>
<td>PPSO</td>
<td>0.7608</td>
<td>0.0370</td>
<td>56.5710</td>
<td>0.3230</td>
<td>1.4317</td>
<td>1.1793</td>
<td>6.6415 E-4</td>
</tr>
<tr>
<td></td>
<td>SA [17]</td>
<td>0.7623</td>
<td>0.0345</td>
<td>43.1034</td>
<td>0.3230</td>
<td>1.5172</td>
<td>1.1793</td>
<td>1.3792 E-3</td>
</tr>
<tr>
<td>DDM</td>
<td>PPSO</td>
<td>1.0305</td>
<td>1.2013</td>
<td>981.9845</td>
<td>3.4823</td>
<td>1.3512</td>
<td>0.0001</td>
<td>2.0800 E-3</td>
</tr>
</tbody>
</table>

PV, photovoltaic; SDC, single-diode cell; SDM, single-diode module; DDC, double-diode cell; DDM, double-diode module; PPSO, parallel particle swarm optimization; LSO, least square optimization; PS, pattern search; SA, simulated annealing.

![Figure 4](https://example.com/figure4.png)

**Figure 4.** Convergence characteristics of parallel particle swarm optimization with relevant photovoltaic models (a) single-diode cell, (b) double-diode cell, (c) single-diode module, and (d) double-diode module.
in Table I. Among these test results, the $\bar{e}$ obtained by the PPSO with DDC achieves the lowest value, recording 6.6415E-4, which is 51.85% lower than the SA with the same PV electrical model, and is approximately 3% lower than the PPSO with the SDC. It is observed that $\bar{e}$ can be reduced if we apply the DDC instead of SDC. However, the complicated model DDM does not always give accurate simulation results in parameter estimation. In the simulation results for the PV module, the accuracy cannot be improved by using the DDM.

Figure 4 shows the qualitative representation of the average evolution performance of the PPSO method for different electrical PV models. The fitness value, namely, the RMS error, is averaged over 30 runs of the applied parameter estimation methods. In general, the average fitness of PPSO drops dramatically in the convergence traces, especially before the first 2000 iterations. The average fitness of the SDM reaches the lowest value after 10,000 iterations by using the PPSO with 2048 particles as seen in the plots. Whichever model we use, the algorithm with a larger swarm size tends to be faster in terms of convergence speed.

Based on the previously discussed analysis, the PPSO shows its consistent performance of extracting the parameters from the experimental data with a high accuracy. Figure 5 further demonstrates the distribution of the fitness values obtained from the PPSO method after 20,000 generations. The swarm size is respectively set at 64, 256, 512, 1024, and 2048. It is observed that the medium values tend
to decrease with the increasing of the particle number, which agrees well with the simulation results in Figure 4. The trend of the decrease implies that the PPSO with a large swarm size has a higher possibility of achieving good fitness value without changing the iteration number. In this sense, the PPSO can improve the accuracy in a unit time on a specified device.

From another perspective, the PPSO executes particle evolution processes concurrently with the applied computing device, and in such a way, the efficiency of parameter evaluation can be improved. The speedup, as well as the parallel efficiency of the proposed PPSO method, will be discussed in the subsequent subsection.

5.2. Speedup and parallel efficiency

In the implementation of PPSO, we follow a hybrid approach whereby the fitness is evaluated in the kernel and the other processes (e.g., position updating and velocity updating) are performed in the host device. The results averaged over 30 trials of the PPSO with 20,000 iterations. To show how much the parallel processing speeds up the fitness evaluation function, the execution time on the host and the device are denoted by bars with light and dark colors separately. A comparison of the total execution time and fitness values, both measured in the proposed PPSO-based parameter estimation and its sequential counterpart for PV electrical models, is made in Figure 6. In Figure 6(a)–(d), we observe that the execution time on the fitness evaluation makes up much larger percentage than that on the other functions in sequential processing. Except for the fitness evaluation, the codes of the PPSO and SPSO are exactly the same, and therefore, their execution time on the host is similar. As seen in Figure 6(e)–(h), the computation time takes in fitness valuation function can be significantly reduced by the PPSO. In addition, the total execution time for the DDC is longer than the one for SDC. This happens because the computational complexity of the double-diode model is higher than that of the single-diode model.

To further evaluate the parallel performance of the proposed PPSO algorithm, We evaluate the speedup of PPSO on a number of multi-core computing devices, which includes Intel Core i3-3220 CPU (2 cores, 2 threads, 3.3 GHz), Intel Core i5-3470 CPU (4 cores, 4 threads, 3.2 GHz), Intel Core i7-4770K CPU (4 cores, 8 threads, 3.5 GHz), NVIDIA GeForce GTX 760 GPU (1152 Compute Unified Device Architecture (CUDA) cores, 980 MHz), NVIDIA GeForce GT 620 GPU (96 CUDA cores, 700 MHz), and AMD Radeon R9 200 GPU (2048 stream processors, 1150 MHz).

Table II lists the average speedup of the PPSO with different swarm size on these devices. From the simulation results, some conclusions can be drawn. In most tests, the speedup of the PPSO is above 1. In other words, the execution time of the PPSO is normally shorter than that of the SPSO. Moreover, the parallel program with larger swarm

<table>
<thead>
<tr>
<th>Model</th>
<th>Host Device</th>
<th>Swarm size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>SDC</td>
<td>i3-3220 i3-3220</td>
<td>1.51</td>
</tr>
<tr>
<td></td>
<td>i5-3470 i5-3470</td>
<td>2.07</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k i7-4770 k</td>
<td>2.13</td>
</tr>
<tr>
<td>DDC</td>
<td>i3-3220 i3-3220</td>
<td>1.46</td>
</tr>
<tr>
<td></td>
<td>i5-3470 i5-3470</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k i7-4770 k</td>
<td>2.25</td>
</tr>
<tr>
<td>DDC</td>
<td>i7-4770 k GT620</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k GTX760</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k R9 200</td>
<td>1.40</td>
</tr>
<tr>
<td>SDM</td>
<td>i3-3220 i3-3220</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>i5-3470 i5-3470</td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k i7-4770 k</td>
<td>1.16</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k GT620</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k GTX760</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k R9 200</td>
<td>1.37</td>
</tr>
<tr>
<td>SDM</td>
<td>i3-3220 i3-3220</td>
<td>1.73</td>
</tr>
<tr>
<td></td>
<td>i5-3470 i5-3470</td>
<td>2.31</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k i7-4770 k</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k GT620</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k GTX760</td>
<td>0.46</td>
</tr>
<tr>
<td></td>
<td>i7-4770 k R9 200</td>
<td>1.35</td>
</tr>
</tbody>
</table>

PPSO, parallel particle swarm optimization; SDC, single-diode cell; DDC, double-diode cell; SDM, single-diode module; DDM, double-diode module.
size tends to perform at a faster speed. The exception made by the PPSO with a swarm size of 64 particles. Its speed is even lower than the corresponding sequential version on GT 620 and GTX 760, which implies the overheads on data communication and kernel scheduling are more significant on the two GPUs. With more applied particles, the speedup appears to be a larger ratio. This is because the speedup on the applied multi-core devices over the host processor is large enough to compensate for the initial data transfer cost. From Figure 6(a) and (e), we can conclude that the PPSO can achieve better fitness values if taking the same amount of execution time as the SPSO. Similar trends are observable in the speedup for the DDC, SDM, and DDM. Among these tests, the parallel program with Intel i7-4770 k and AMD R9 200 series exhibits the minimum execution time, recording an average speed up ratio from 3.4426 to 4.6062 for a swarm size set of 2048 particles.

6. CONCLUSION

In this work, a parallel computing paradigm has been shown to speed up the parameter estimation process for various PV models. The proposed PPSO implemented in OpenCL can be executed in a wide range of multi-core computing devices. Fitness evaluations were performed concurrently on multi-processor devices, and the simulation results showed that the PPSO with 2048 particles is capable of accelerating the computational speed by at least 64% on the relevant computing devices. The PPSO records low calculation errors and shows improvement in terms of computational speed. Hence, it is evident that the PPSO possesses exceptional capability in the parameter estimation. Because branching is difficult for all the computing devices, especially GPUs, the process of updating positions and velocities of particles in the PPSO has not been parallelized. This shall be investigated in our future work. Also, we intend to explore more aggressive multi-core computers which are equipped with more powerful GPU capabilities.

REFERENCES


