IET Generation, Transmission & Distribution

Special Issue: New Trends in the Planning of Distribution Network with High Penetration of Renewables and Flexible Loads

GPU-based parallel real-time volt/var optimisation for distribution network considering distributed generators

ISSN 1751-8687
Received on 29th November 2017
Revised 17th March 2018
Accepted on 3rd May 2018
E-First on 7th June 2018
doi: 10.1049/iet-gtd.2017.1887
www.ietdl.org

Shengjun Huang[1,2] ✉, Venkata Dinavahi[1]
[1]Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta T6G 2V4, Canada
[2]College of Information System and Management, National University of Defense Technology, Changsha, Hunan 410073, People's Republic of China
✉ *E-mail: shengjun@ualberta.ca*

Abstract: Although the wide integration of advanced metering infrastructure on distribution network facilitates the application of volt/var optimisation (VVO) in real-time circumstance, the contradiction between heavy computation load and low solution efficiency is still a big challenge, thus the system scales investigated in the literature are limited. In this study, the full AC real-time VVO is formulated based on particle swarm optimisation (PSO) framework and direct approach (DA) power flow method, where all components, such as distributed generator and on-load tap changer transformer, are formulated and integrated into the iterative DA process. Since both PSO and DA are suitable for parallel implementation, the graphics processing unit (GPU) is introduced for acceleration in order to achieve the possibility for real-time application. All the solution process is executed by GPU with the well-established data structure and thread organisation pattern, resulting in high efficiency by guaranteeing coalesced access within each warp. Case studies are conducted on four systems with sizes ranging from 136-bus to 1760-bus. Solution accuracy and convergence property are validated by the popular open source package Matpower. Based on the results from solution efficiency comparison between CPU sequential, CPU parallel, and GPU parallel programs, the promise of the proposed parallel implementation scheme for practical application is established.

Nomenclature

[]	vector or matrix of a series of elements
ΔQ_i^c	reactive power in each switch step of the switched capacitor (SC) at node i
ΔV	voltage updating at nodes in each iteration
min, max	minimum and maximum values, respectively
a_{ij}^t	tap ratio of the on-load tap changer (OLTC) between nodes i and j
B_l	branch current of branch l
BCBV	branch-current to bus-voltage matrix
BIBC	bus-injection to branch-current matrix
DLF	distribution load flow matrix
f_i^g	power factor of the distributed generator (DG) installed at node i
I_i^k	equivalent current injection of bus i at iteration k
n_b	total number of buses
n_c	total number of circuits/branches
P_i^d	active power demand at node i
P_i^g	active power output of the DG installed at node i
P_loss	total active power losses
Q_i^c	reactive power injection of the SC at node i
Q_i^d	reactive power demand at node i
Q_i^g	reactive power output of the DG installed at node i
s_i^c	switch step of the SC at node i
S_{ij}^t	short circuit admittance of OLTC between nodes i and j
V_i^k	bus voltage of bus i at iteration k
Y_i	shunt admittance at node i
Y_{ij}	total lumped shunt admittance for medium-length line between nodes i and j
z_{ij}/z_l	series impedance of branch ij or line l
P_{ij}	real power flow of branch ij
Q_{ij}	reactive power flow of branch ij
S_{ij}^max	maximum complex power of branch ij

1 Introduction

Distribution network is the final stage in the power delivery to bridge individual consumers with the transmission system [1]. One of its major responsibilities is the voltage and reactive power (var) management, i.e. achieving high efficiency, reliability, and quality on the power supply. A lot of control devices are available to fulfil that goal, such as on-load tap changer (OLTC) transformer, voltage regulator, and switched capacitor (SC), and so on. In terms of how to determine/adjust the operating parameters of these facilities, the volt/var optimisation (VVO) was proposed [2–4]: *minimising system active power losses while satisfying equality constraints to node active and reactive power balances, as well as lower/upper bounds of node voltages*.

Similar to the unit commitment problem in the transmission network, VVO is usually performed on a Day-Ahead time span based on the forecast demand [2, 5], which is marked as DAVVO henceforth in this paper for brevity. The daily operational constraints, e.g. the times of switching operations in a single day should be restricted to a specified number for the purpose of diminishing degradation, are frequently considered in the DAVVO formulation. Although DAVVO has been thoroughly investigated [2–14], its capability to withstand fast variation is limited.

In the last decades, distributed generators (DGs), including fuel cell, photovoltaic cell, wind turbine, micro-turbine, and so on, have been widely integrated into distribution networks, which is of great significance for the reduction of transmission loss as well as carbon emission. On the other hand, the high penetration of DG also brings large fluctuations, resulting in the fact that the DAVVO solution is non-optimal or even infeasible. Therefore, real-time VVO (RTVVO) has been proposed [15–18] to address these issues.

The popularity of the utilisation of new technologies, such as advanced metering infrastructure (AMI), keeps increasing in the context of smart grid. Two-way communication can be provided by the AMI system to collect the information/data from smart meters and distribute the instruction/command to devices at the same time, i.e. RTVVO is technically possible from the perspective of communication [19]. Nevertheless, in terms of fast computation for real-time decision making, the technique is far from mature.

Due to different nature of control devices, VVO is conventionally formulated as a mixed-integer non-linear programming (MINLP) problem with complex numbers and constraints. Two types of methods have been widely utilised for the solution of such a non-deterministic polynomial hard (NP-hard) problem: (i) mathematical programming methods, including robust optimisation [4, 7], mixed-integer programming [2, 8], Benders decomposition [9], model predictive control [10], and so on; (ii) meta-heuristic methods, such as genetic algorithm (GA) [11, 12], particle swarm optimisation (PSO) [13, 14], simulated annealing [15], and so on. A common feature for many of the proposed mathematical programming methods is the relaxation of original variables and constraints [11]. For example, linear power flow formulation is utilised in [2], and integer variables are relaxed into continuous during computation and discretised subsequently in [6]. Although relaxation can facilitate computation and mathematical programming methods are deterministic, the accuracy is sacrificed. On the other hand, the meta-heuristic methods are capable to address various types of problems of their original form, such as non-linear, non-convex, mixed-integer, NP-hard, combinatorial, and so on. Therefore, the AC power flow (ACPF) calculation for distribution network can be fully integrated into the VVO framework. Nevertheless, they are usually computational intensive and the global optimality is not guaranteed since their searching process is probabilistic. Although it may trap in and terminate at a local optimal solution, the solution quality is still satisfactory due to its large global searching capability.

In order to alleviate the computational burden and accelerate the solution process of VVO, parallel computing technique has been introduced in [13]. The parallel realisation of PSO was fulfilled with OpenMP [20] on CPU, resulting speedups of 1.95×, 3.42×, and 3.72× with two, four, and eight threads, respectively. It is observable that the parallel efficiency faced with the bottleneck on the eight threads. In addition, the investigated system is only 14-bus, which is partially due to the contradiction between real-time requirement and solution efficiency. Actually, for other references related to VVO, whether the real time and ACPF are considered or not, the target system scales are also limited. Table 1 gives a brief summary of these references. It is noticeable that all the RTVVO literature enables full ACPF since the feasibility is of higher priority in real-time circumstance, where the in-time corrective operation is very difficult. On the other hand, without the introduction of high-performance computation platform and parallel computing technique, the system scale is restricted.

Since the capability for general-purpose scientific computing has been revealed by Nvidia in late 2006 with the release of compute unified device architecture (CUDA) [21], the graphics processing unit (GPU) has gained a lot of popularity on the power systems optimisation and simulation problems, such as transient stability simulation [22], electromagnetic transient simulation [23, 24], static security analysis [25], real-time optimal power flow [26], dynamic state estimation [27, 28], and so on; nevertheless, it has never been introduced for RTVVO to the best of our knowledge. Therefore, this paper intends to fill this gap. The main contributions of this work are as follows:

- Instead of utilising of DCPF or other types of relaxations, full ACPF calculation of distribution network has been integrated into RTVVO based on the DA [29] with the consideration of DG, OLTC, SC, and medium-length lines. Inspired by the well-known backward–forward sweeping method, the DA provides a very compact vectorised formulation with outstanding computational and convergence properties [30]. Nevertheless, during the application of DA on RTVVO in the previous work [15], the formulation details are not sufficient, e.g. transformers are simplified as branches.
- Based on full ACPF, the RTVVO solution framework has been developed, where PSO is utilised for evolution and convergence. In order to achieve the capability for real-time application, the GPU parallel computing technique has been introduced to accelerate the RTVVO solution process. A lot of details are revealed to facilitate the parallel implementation: (i) redesign the data structure of matrix [**DLF**] in DA; (ii) reorganise the thread organisation to translate the memory access pattern from scattered into coalesced in each step.
- Different with the previous work with limited system size, the proposed parallel implementation framework has been validated on a lot of distribution networks with the scale ranging from 136-bus to 1760-bus. In addition to the accuracy validation with open source package Matpower [31], the solution efficiency comparison is conducted between CPU sequential (both Matpower in Matlab and DA in C++), CPU parallel (with OpenMP), and GPU parallel (with CUDA) programs. As indicated by results, the proposal is promising for practical application.

The remaining of this paper is organised as follows. Section 2 is devoted to the problem formulation, including the iterative process of DA method, mathematical model of components, and the whole MINLP optimisation model of RTVVO. PSO solution framework for this problem is briefly introduced in Section 3. Implementation details related to parallel computing on GPU are revealed in Section 4, such as the design of data structure and organisation of threads. Section 5 validates the solution accuracy and efficiency of the proposed implementation scheme with case studies on four test systems. Finally, conclusion and future work are summarised in Section 6.

2 Problem formulation

In this work, three types of control devices (SC, OLTC, and DG) are considered for VVO with the objective of minimising the total

Table 1 Parts of references for VVO

References	Scales	RTVVO	Full ACPF	Parallel	GPU-enabled
[13], 2015	14-bus	×	√	√	×
[7], 2015	30-bus	×	×	×	×
[17], 2016	33-bus	√	√	×	×
[18], 2016	33-bus	√	√	×	×
[15], 2015	37-bus	√	√	×	×
[5], 2017	37-bus	×	√	×	×
[3], 2009	68-bus	×	√	×	×
[2], 2015	69-bus	×	×	×	×
[14], 2010	70-bus	×	√	×	×
[6], 2014	119-bus	×	×	×	×
[4], 2017	123-bus	×	×	×	×
[11], 2006	220-bus	×	√	×	×
proposed work	1760-bus	×	√	√	√

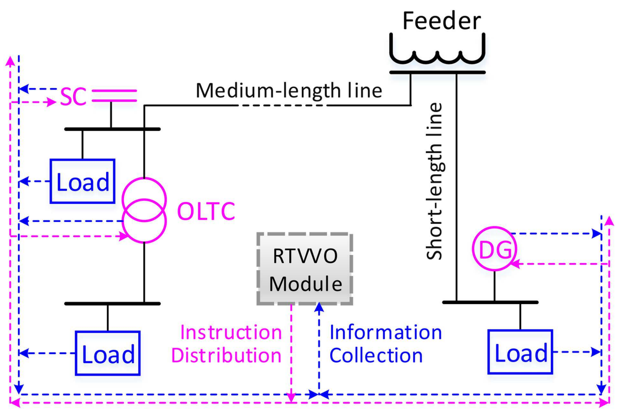


Fig. 1 *Schematic of RTVVO framework for the distribution network*

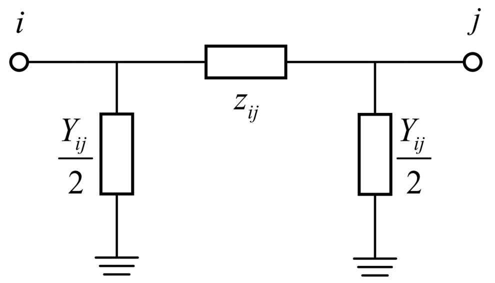


Fig. 2 *Pi-equivalent model of the medium-length line*

active power loss. Based on the integration of AMI, a general implementation scheme of RTVVO for distribution network is given in Fig. 1. At the beginning, information related to the current and historical status of SC, OLTC, DG, and load is sampled and collected through smart meters and AMI. Based on these data as well as the parameters of all components (including short/medium-length line and other devices), the RTVVO module performs the fast computing and outputs the optimal coordinated schedule for different types of equipment within a fixed time interval, e.g. 10 s. Finally, these instructions are distributed to the corresponding control devices via AMI. Except for the information collection and instruction distribution, the solution of MINLP RTVVO problem turns to be the most important task, which is addressed with RTVVO module in Fig. 1. Within RTVVO module, the ACPF calculation will be intensively involved, thus its solution efficiency is of great significance for the whole decision-making process. In this paper, the DA proposed in [29] is utilised for distribution network ACPF calculation. It should be noted that the DA is robust for various types of distribution networks, whatever it is single-phase balanced or three-phase unbalanced, radial or weekly meshed. In the following, the single-phase balanced radial distribution network is presented for illustration.

2.1 DA power flow method

Given a distribution network with n_b nodes (where node 1 is regarded as the reference bus), the equivalent current injection for node i at the kth iteration can be described as

$$I_i^k = \left(\frac{P_i^d + jQ_i^d}{V_i^k v}\right)^*, \quad i \in [2, n_b], \tag{1}$$

where $*$ is the conjugate operator. Accordingly, the branch current vector can be obtained

$$[\boldsymbol{B}_l]_{n_c \times 1} = [\mathbf{BIBC}]_{n_c \times (n_b - 1)}[\boldsymbol{I}]_{(n_b - 1) \times 1}. \tag{2}$$

Subsequently, the vector for voltage updating can be generated

$$[\Delta \boldsymbol{V}^k]_{(n_b - 1) \times 1} = [\mathbf{BCBV}]_{(n_b - 1) \times n_c}[\boldsymbol{B}_l]_{n_c \times 1}. \tag{3}$$

Finally, node voltage vector can be updated

$$[\boldsymbol{V}^{k+1}]_{(n_b - 1) \times 1} = [\boldsymbol{V}^0]_{(n_b - 1) \times 1} + [\Delta \boldsymbol{V}^k]_{(n_b - 1) \times 1}, \tag{4}$$

where $[\boldsymbol{V}^0]$ is a vector with all elements valued as the voltage of reference bus.

Given an initial flat voltage profile, (1)–(4) can be solved sequentially and iteratively until the system reaches a steady state, i.e. the node voltage difference between two successive iterations is less than the specified threshold.

Ultimately, the total active power loss can be obtained

$$P_{\text{loss}} = \sum_{l=1}^{n_c} z_l |B_l|^2. \tag{5}$$

2.2 Mathematical formulation of components

The above formulation is the basic version of DA, where the transformers and lines are formulated as simple series impedances. Although this is acceptable for short-length lines and untapped transformers, modifications are required to deal with other common devices shown in Fig. 1, such as DG, SC, medium-length line, and OLTC transformer.

2.2.1 Constant power factor model of DG: DG can be installed by the combination of different energy sources (fuel, wind, and solar) and conversion devices (induction generator, static power converter, and synchronous generator), resulting in various output characteristics [15]. In order to formulate them, three models are developed [32]: constant power factor model, constant voltage model, and variable reactive power model, of which the first one is utilised in this work due to its great popularity. It should be noted that the DA method is capable for all three DG models mentioned above.

With specified active power output and power factor, the reactive power output can be calculated [32]

$$Q_i^g = P_i^g \tan\left(\cos^{-1}(f_i^g)\right), \tag{6}$$

then the equivalent node current injection given in (1) should be updated as

$$I_i^k = \left(\frac{(P_i^d - P_i^g) + j(Q_i^d - Q_i^g)}{V_i^k}\right)^*, \quad i \in [2, n_b]. \tag{7}$$

2.2.2 Discrete steps of SC: Given an SC with step s_i^c, its reactive power injection can be given as

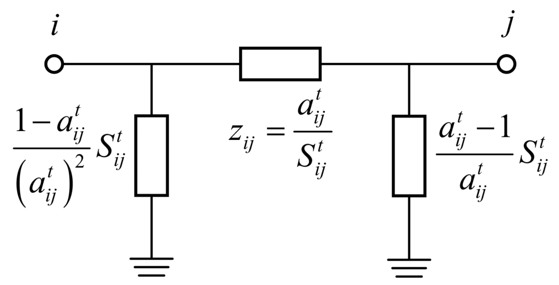
$$Q_i^c = s_i^c \Delta Q_i^c. \tag{8}$$

Accordingly, based on (7), the node current injection needs to be updated

$$I_i^k = \left(\frac{(P_i^d - P_i^g) + j(Q_i^d - Q_i^g - Q_i^c)}{V_i^k}\right)^*, \quad i \in [2, n_b]. \tag{9}$$

2.2.3 Pi-equivalent model of medium-length line: Given a medium-length line between nodes i and j with series impedance z_{ij} and total lumped shunt admittance Y_{ij}, a pi-equivalent model can be formulated as Fig. 2. Take the shunt admittance $Y_i = Y_i + 0.5Y_{ij}$ into consideration, the node current injection given in (7) should be updated as

$$I_i^k = \left(\frac{(P_i^d - P_i^g) + j(Q_i^d - Q_i^g - Q_i^c)}{V_i^k}\right)^* + Y_i V_i^k, \quad i \in [2, n_b]. \tag{10}$$



i
j
$\frac{1-a_{ij}^t}{(a_{ij}^t)^2}S_{ij}^t$
$z_{ij}=\frac{a_{ij}^t}{S_{ij}^t}$
$\frac{a_{ij}^t-1}{a_{ij}^t}S_{ij}^t$

Fig. 3 *Pi-equivalent model of the OLTC transformer*

2.2.4 Pi-equivalent model of OLTC transformer: Given an OLTC transformer between nodes i and j with short circuit admittance S_{ij}^t and regulation tap a_{ij}^t, a pi-equivalent model can be formulated as Fig. 3. Similar to Fig. 2 and (10), the node current injection needs to be updated with the following shunt admittances:

$$Y_i = Y_i + \frac{1-a_{ij}^t}{(a_{ij}^t)^2}S_{ij}^t, \quad Y_j = Y_j + \frac{a_{ij}^t - 1}{a_{ij}^t}S_{ij}^t. \tag{11}$$

In addition to the shunt admittances, the variation on the series impedance shown in (12) also needs to be integrated into the DA iterative solution process

$$z_{ij} = \frac{a_{ij}^t}{S_{ij}^t}. \tag{12}$$

Initially, z_{ij} is implicitly included in (3) according to the following relationship:

$$[\mathbf{BCBV}]_{(n_b-1)\times n_c} = [\mathbf{BIBC}]^{\mathrm{T}}_{n_c\times(n_b-1)}[z]_{(n_b-1)\times(n_b-1)}, \tag{13}$$

where $[z]$ is a diagonal matrix with all elements are corresponding z_{ij}. It should be noted that, the radial distribution network always has $n_c = n_b - 1$, thus the matrix multiplication in (13) is established; while for the meshed networks, as reported in [29, 30], similar processes are also possible. According to (2), (3), and (13), (3) can be rewritten as

$$[\Delta V^k] = [\mathbf{BIBC}]^{\mathrm{T}}[z][\mathbf{BIBC}][I] = [\mathbf{DLF}][I]. \tag{14}$$

It should be noted that [**BIBC**] is constant since it is determined by the network topology, whereas [**DLF**] may change as the variation of parameter a_{ij}^t.

2.3 Mathematical formulation of RTVVO

Based on the DA power flow method and component modelling, the mathematical formulation of RTVVO can be given as follows.

2.3.1 Objective function: The objective of RTVVO is to minimise the total active power loss

$$\min_{[P^g],[f^g],[s^c],[a^t]} \left\{ P_{\text{loss}} = \sum_{l=1}^{n_c} z_l |B_l|^2 \right\}. \tag{15}$$

It should be noted that the objective function (15) can be easily extended to more sophisticated ones, such as minimisation of switching operations of OLTC and SC, minimisation of voltage derivation, and minimisation of overall energy consumption, based on the combination of DA and PSO, which will be explained in the next section. For more details on the construction and calculation of objective function, please refer to [17].

2.3.2 Constraints: For this practical problem, the constraints include:

- Distribution network power flow equations

$$\text{steady state of } (1)-(14). \tag{16}$$

- Active power constraints of DG

$$P_i^{g,\min} \le P_i^g \le P_i^{g,\max}, \quad P_i^g \text{ is continuous}. \tag{17}$$

- Power factor constraints of DG

$$f_i^{g,\min} \le f_i^g \le f_i^{g,\max}, \quad f_i^g \text{ is continuous}. \tag{18}$$

- Switch step of SC

$$s_i^{c,\min} \le s_i^c \le s_i^{c,\max}, \quad s_i^c \text{ is discrete (integer)}. \tag{19}$$

- Tap of OLTC transformer

$$a_{ij}^{t,\min} \le a_{ij}^t \le a_{ij}^{t,\max}, \quad a_{ij}^t \text{ is discrete (integer)}. \tag{20}$$

- Bus voltage magnitude limits

$$V^{\min} \le |V_i| \le V^{\max}. \tag{21}$$

- Distribution line thermal limits

$$\sqrt{P_{ij}^2 + Q_{ij}^2} \le S_{ij}^{\max}. \tag{22}$$

- Reactive power overcompensation limits

$$\sum_{i=1}^{n_b} Q_i^c + \sum_{i=1}^{n_b} Q_i^g \le \sum_{i=1}^{n_b} Q_i^d. \tag{23}$$

It is noticeable that the practical operation limits for SC and OLTC are usually included in DAVVO, e.g. there are maximum allowable daily operating times for each device. Since the RTVVO is designed for a specified time point rather than the whole day, these constraints cannot be directly added. Instead, it can be replaced by another constraint in this work: a component is available for adjusting new commands only when it has been working at a fixed status for a specified number of time intervals. Accordingly, if OLTC mk is not available for adjusting at the current RTVVO, then the values of $a_{mk}^{t,\min}$ and $a_{mk}^{t,\max}$ in constraint (20) should be adjusted into the former status of a_{mk}^t, thus OLTC mk will still working at a constant status in the next time interval. That is the reason why historical operation data is required in RTVVO module. Similar operations can also be conducted on (19) for SC.

3 Solution framework

In this paper, the PSO framework is utilised for the solution of RTVVO, where DA is integrated for the fitness evaluation of each particle. A general flowchart is given in Fig. 4, where the data exchanging with Fig. 1 is highlighted with dashed lines. Since the PSO has been fully established in the literature, only a few basic details are introduced here:

- *Solution encoding:* The decision variables in RTVVO are P_i^g, f_i^g, s_i^c, and a_{ij}^t, where the former two are encoded as continuous numbers and the rest are regulated as integer. A single particle is the vector $\boldsymbol{x}_i = [P_i^g, f_i^g, s_i^c, a_{ij}^t]$ containing all the decision variables.
- *Solution initialisation:* All particles are uniformly sampled within the solution space shaped by constraints (17)–(20). Rounding process will be carried out for the integer decision variables s_i^c and a_{ij}^t.
- *Fitness evaluation:* Penalty factor will be added in objective function (15) to punish the violations of constraints (21)–(22). On the other hand, constraint (16) will be satisfied by DA power flow, and (17)–(20) will be addressed in solution updating process. To sum up, for a given solution, the fitness evaluation process starts with the distribution network power flow calculation with DA method, resulting in that the branch current and node voltage are available. Based on these data, the

objective value of power loss and other sophisticated goals can be generated. Finally, the constraints of (21)–(22) and others (might be included in the future) are validated to determine whether adding a penalty to the obtained objective value or not.

- *Velocity updating:* Mechanism reported in [33, 34] is utilised for the velocity updating. Equations (24) and (25) correspond to continuous and discrete decision variables, respectively

$$v_i^{k+1} = w_0 v_i^k + c_1 w_1 (p_i^k - x_i^k) + c_2 w_2 (g^k - x_i^k), \qquad (24)$$

$$v_i^{k+1} = \text{round}\big(w_0 W_0 v_i^k + c_1 W_1 (p_i^k - x_i^k) + c_2 W_2 (g^k - x_i^k)\big), (25)$$

where $w_0 = 1 - 0.6k/N$, $c_1 = 2$, $c_2 = 2$ are fixed parameters; k is the current iteration order; N is the maximum number of iterations; w_1 and w_2 are uniformly random numbers within $[0, 1]$; W_0 is random number taking discrete values of 0, -1, or 1; W_1 and W_2 are random discrete numbers of either 0 or 1. Compared with (24), (25) provides more possibilities by the introduction of W_0, W_1, and W_2, such as temporarily eliminating the influence of velocity, local best, and global best by setting W_0, W_1, and W_2 as 0, respectively. The search direction can even be turned totally inverse if $W_0 = -1$. These possibilities bring more diversity to the searching process, resulting in larger global searching capability.

- *Solution updating:* The solution updating is performed with (26). It can be seen from the initialisation process and (25) that the discrete decision variables are always kept in integer form. If the updated decision variable is out of the range regulated by (17)–(20), it will be forced to the nearest boundary

$$x_i^{k+1} = x_i^k + v_i^{k+1}. \qquad (26)$$

Algorithm 1: PSO framework for RTVVO

```
1: Prepare matrix [BIBC] based on graph searching.
2: Input state variables $P_i^d$, $Q_i^d$, $z_{ij}$, $Y_{ij}$, $\Delta Q_i^c$, and $S_{ij}^t$.
3: Initialise the population.
4: for each iteration do
5:     for each particle do
6:         Read decision variables $P_i^g$, $f_i^g$, $s_i^c$, and $a_{ij}^t$.
7:         Update network configurations $Q_i^g$, $Q_i^c$, $Y_i$, and $z_{ij}$ according to (6), (8), (11), and (12).
8:         Update matrix $[\mathbf{DLF}] = [\mathbf{BIBC}]^T[z][\mathbf{BIBC}]$.
9:         Initialise a flat node voltage vector $[V^0]$.
10:        while max $\{|V^k - V^{k-1}|\} > \in$ do
11:            Calculate $[I^k]$ according to (10).
12:            Compute $[\Delta V^k]$ based on (14).
13:            Update $[V^{k+1}]$ based on (4), and set $k = k + 1$.
14:        end while
15:        Calculate the active power loss based on (2) and (5).
16:        Check the constraints (21)–(22) and update the fitness value with penalty factor if it is required.
17:    end for
18:    Determine the local and global particles, update the velocity and position $[P_i^g, f_i^g, s_i^c, a_{ij}^t]$ based on (24)–(26).
19: end for
20: Output the global best particle.
```

- *Termination criteria:* In order to guarantee fair comparison in case studies, i.e. the computation load for different runs are the same and insensitive to random numbers, the PSO will terminate at a fixed number of iteration in this paper. It is also easy to be extended to other criteria, such as terminating if the global best has not been updated for a specified number of iterations. It should be noted that the global optimality of the final solution is not guaranteed whatever termination criteria is utilised since the searching process is probabilistic and the probability to sink in local optima always exist.

4 Parallel implementation

For the purpose of facilitating description, the solution process given in Section 3 is summarised as Algorithm 1 with the detailed data flow from DA and PSO. Since all particles are mutually independent, each particle at a specified iteration can be manipulated with one thread or block on GPU, therefore, the parallel implementation seems to be trivial. However, in order to gain superior performance, the design of data structure and thread organisation require further investigation.

Algorithm 2: DLF translating from $\{p, i, xp, xx\}$ to $\{p, i, x\}$

```
1: Initialise all $x(i)$ into 0 (suppose $x$ has $n_x$ elements).
2: for $i = 1 \cdots n_x$ do
3:     for $j = xp(i) \cdots (xp(i+1) - 1)$ do
4:         $x(i) = x(i) + z(xx(j))$.
5:     end for
6: end for
```

4.1 Data structure

As can be seen from Algorithm 1, each particle maintains a copy of all intermediate vectors and matrix [**DLF**], whereas the matrix [**BIBC**] is constant and the same for all particles. For different vector copies across particles, they are stored as a unified matrix with each row corresponds to one particle. An illustrative example for vector [$\boldsymbol{Y}$] is given in Fig. 5. In this paper, both [**DLF**] and [**BIBC**] are stored with the compressed sparse row (CSR) format [35] since it enables fast row access and matrix–vector multiplications. As the CSR structure $\{p, i, x\}$ can be decomposed into three arrays p, i, and x, the unified vector storage pattern shown in Fig. 5 is also valid for matrix storage. Since the [**DLF**] across different particles have the same pattern, vectors $\boldsymbol{p}$ and $\boldsymbol{i}$ are the same for each particle, thus only vector $\boldsymbol{x}$ is required in different particles for distinction, i.e. the storage space for $\boldsymbol{p}$ and $\boldsymbol{i}$ is saved.

Although the sparse technique has been utilised, the calculation of [**DLF**] by two times of matrix multiplication $[\mathbf{DLF}] = [\mathbf{BIBC}]^T[z][\mathbf{BIBC}]$ is still computationally intensive. In Fig. 6, at least 27 atom operations are involved. As the matrix

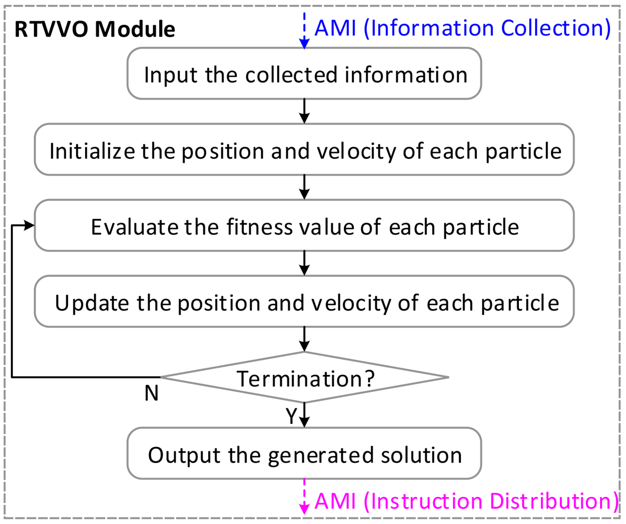

RTVVO Module
AMI (Information Collection)
Input the collected information
Initialize the position and velocity of each particle
Evaluate the fitness value of each particle
Update the position and velocity of each particle
Termination?
N
Y
Output the generated solution
AMI (Instruction Distribution)

Fig. 4 *General flowchart of the PSO utilised in the RTVVO module*

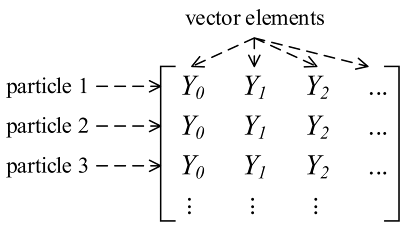


Fig. 5 *Unified vector storage structure across different particles*

[**BIBC**] is fixed and all its elements are either 0 or 1, we intend to improve the computation efficiency by translating the matrix multiplication into the point-wise substitution. Fig. 6 gives an illustrative example. First, we change the CSR format of [**DLF**] from $\{p, i, x\}$ into $\{p, i, xp, xx\}$. The array xx is derived from x by decomposing all atom operations into successive entries, i.e. expanding one element $z_1 + z_2$ into two elements 1 and 2. For each original element of x, its new order in xx is recorded as the array xp. Just like the array p, one more element is supplemented at the end. Given an array $[z]$, it is easy to obtain $\{p, i, x\}$ from $\{p, i, xp, xx\}$ by Algorithm 2 with only 16 atom operations. The performance difference between matrix multiplication and point-wise substitution is much larger for bigger systems, which will be demonstrated in case studies.

4.2 Thread organisation

CUDA provides a three-level parallelism based on thread hierarchy [21]: different streaming multiprocessor (SM) in GPU is possible for concurrent execution; each SM can launch multiple blocks at the same time; the threads within one block will be executed simultaneously. Different blocks are independent with each other, while threads within one block can cooperate via shared memory or barrier. The barrier synchronises all threads in one block by forcing them to wait at a specified point until everyone has reached. The shared memory is possessed by each block and is visible to all threads within that block, but the size is limited. Since the access latency of shared memory is low, it should be fully utilised when it is possible. In addition, each thread can access the much bigger global memory of GPU with a higher latency.

Although SM receives computation task in the unit of block, it creates, manages, schedules, and executes threads in groups of 32 and named warps [21]. A warp executes one common instruction at a time. If the target address for all 32 threads is successive, the access can be fulfilled with full efficiency (called coalesced access); otherwise, the divergence (scattered access) will occur and result into the low bandwidth. Since the efficiency of warps dominates the whole solution performance, how to guarantee successive data access is of high priority when organising threads [25].

In the following subsections, key steps of Algorithm 1 are tuned for coalesced access with thread organisation.

4.2.1 Parallel regular mapping: There are a lot of element-wise calculations and updating operations in Algorithm 1, such as lines 7, 9, 11, 13, and 16, which can be regarded as regular mapping. Take the line 11 as an example, the last term of (10) is $I_i = Y_i V_i$, Fig. 7 illustrates the thread organisation pattern, where $\otimes$ represents the element-wise mapping process. In alliance with Fig. 5, vectors $[\boldsymbol{Y}]$ and $[\boldsymbol{V}]$ across particles are stored in one matrix. It can be seen that there is no leap during the access, thus we naturally distribute each particle (one row) into one block. In Fig. 7, all particles are concurrently executed since multiple blocks can be launched at the same time. Within all blocks, coalesced access is fulfilled by each warp, e.g. the successive address from Y_0 to Y_{31} are accessed by threads 0 to 31.

4.2.2 Parallel reduction: Different with regular mapping process, where the length of the input vector is the same as the output vector, the reduction process takes a vector as input but output only one scale value. This type of operation is also involved in Algorithm 1, such as the maximising in line 10, the minimising in line 18, and the summation in line 15. It should be noted that the reduction operation in line 15 is due to (5), where the term $z_l |B_l|^2$ is calculated with parallel regular mapping and stored as an intermediate vector. Similar to Fig. 7, each particle (vector) is handled with one block. The thread organisation for parallel reduction within each block is illustrated in Fig. 8. The first step is copying the original vector $[\boldsymbol{a}]$ stored in global memory into $[\boldsymbol{b}]$ in shared memory. During this step, reduction process may be required, e.g. reducing a_0 and a_{1024} to b_0. There should be a barrier at the end of this step to guarantee that all shared memory has finished data updating, otherwise, dirty data will appear and result into a wrong output. The following steps are familiar with the first one, except that both input and output data are stored in the shared memory and the number of active thread is reduced into half of the former. It can be seen that the coalesced access is guaranteed at each step, i.e. there is no leap within each warp.

4.2.3 Parallel matrix–vector multiplication: Although the matrix–matrix multiplication in $[\mathbf{DLF}] = [\mathbf{BIBC}]^{\mathrm{T}}[z][\mathbf{BIBC}]$ has been eliminated by data structure design in the above, the matrix–vector multiplication is inevitable in line 12 of Algorithm 1. In alliance with the previous process, each block is assigned to one particle, i.e. one matrix–vector multiplication process will be fulfilled within one block. Fig. 9 demonstrates two types of thread organisation for parallel matrix–vector multiplication within each block. After the updating in line 8 of Algorithm 1, the matrix [**DLF**] is stored in CSR format, which is shown in the bottom of Fig. 9. The naive parallel implementation of matrix–vector multiplication is to assign each row into one thread as shown in Fig. 9*a*. However, the scattered access will appear when the CSR format is utilised for matrix storage. For example, thread 0 to 3 need to access the first non-zero number in the corresponding row at the same time, but these numbers are discontinuous in the vector $\boldsymbol{x}$. On the other hand, we assign each row into one warp as shown in Fig. 9*b*, where the coalesced access has been achieved with illustrative dashed arrows. It should be noted that the parallel reduction for summation in each row is required.

4.2.4 Parallel irregular mapping: Based on the above parallel processing, each step in Algorithm 1 can be efficiently executed except for line 8, where irregular mapping is confronted. Actually, line 8 of Algorithm 1 is realised with Algorithm 2. Conventionally, if each block is assigned to one particle, the scattered access will appear as shown in Fig. 10*a*. The reason lies in line 4 of Algorithm 2. During the process to access $z(xx(j))$, although j is in successive for each thread, the result of $xx(j)$ is discontinuous, resulting in irregular access of vector $[z]$. In this example, $x(0)$ and $x(1)$ correspond to $z(2)$ and $z(1)$, respectively. In order to achieve the coalesced access, matrices $[\boldsymbol{x}]$ and $[\boldsymbol{z}]$ shown in Fig. 10*a* are transposed, and we distribute each element to one block as shown in Fig. 10*b*. For the new scheme, threads 0 and 1 in block 0 are launched to update $x(0)$ for particle 1 and 2, respectively. Since the mapping relationship between $x(0)$ and $z(2)$ holds for all particles, these two threads should access $z(2)$ of particles 1 and 2, thus the successive addresses are accessed.

4.2.5 Parallel matrix transpose: In the process of parallel irregular mapping, matrix transpose is involved. In order to coordinate with other processes, parallel matrix transpose should be executed. Fig. 11 illustrates two kinds of implementation. As shown in Fig. 11*a*, the naive matrix transpose will result in scattered access. Nevertheless, the transposing process is divided into two steps in Fig. 11*b* with the introduction of shared memory. In step 1, a small piece of the original matrix is copied into shared memory with one warp for each row. Step 2 is copying that piece from shared memory to target address with one warp for each column of the original matrix (when transposed, it appears as one row in the new matrix). It can be seen that in each step, the coalesced access is obtained. The reason for transposing piece by piece is that the size of shared memory is limited.

5 Case studies

In this section, four distribution networks retrieved from [36] are employed for case studies. Based on the original network topology and component data, a lot of VVO devices are added. For simplicity, the parameter of each type of component added into different networks is identical. For each system, a maximum of 30% of the total active demand can be provided with DGs. Each DG has a capacity of 0.5 MW, with a power factor between −0.9 (lagging) and 0.9 (leading). A maximum of 10% of the total reactive demand can be supported by SC. The number of

$$[DLF]=[BIBC]^T[z][BIBC]$$

$$=\begin{bmatrix}1&&&&\\1&1&&&\\1&&1&&\\&&&1&\\&&&1&1\end{bmatrix}\begin{bmatrix}z_1&&&&\\&z_2&&&\\&&z_3&&\\&&&z_4&\\&&&&z_5\end{bmatrix}\begin{bmatrix}1&1&1&&\\&1&&&\\&&1&&\\&&&1&1\\&&&&1\end{bmatrix}=\begin{bmatrix}z_1&z_1&z_1&&\\z_1&z_1+z_2&z_1&&\\z_1&z_1&z_1+z_3&&\\&&&z_4&z_4\\&&&z_4&z_4+z_5\end{bmatrix}$$

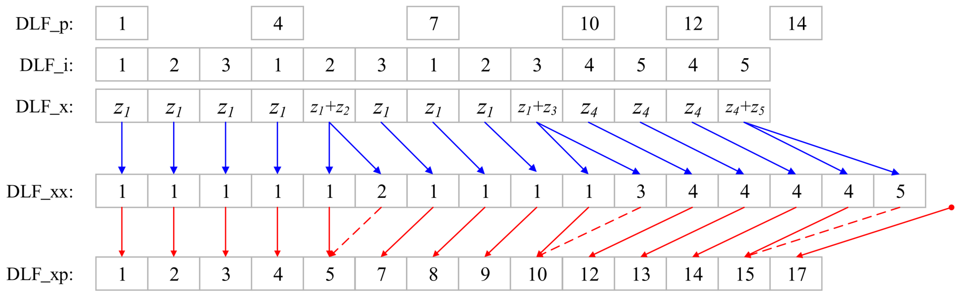


Fig. 6 *Sparse storage patterns of DLF matrix in CSR format with* $\{p, i, x\}$ *and* $\{p, i, xp, xx\}$

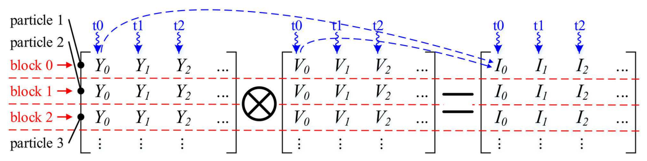


Fig. 7 *Thread organisation for parallel regular mapping*

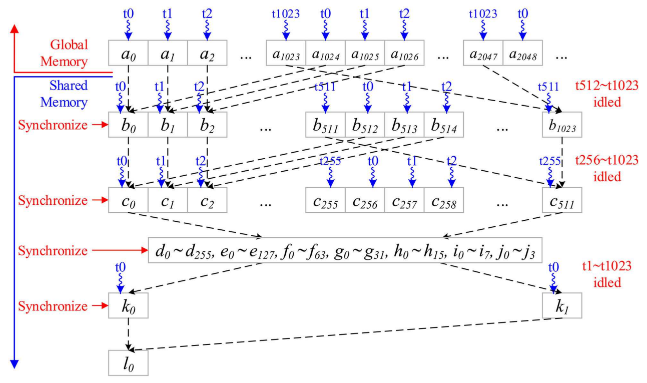


Fig. 8 *Thread organisation for parallel reduction within each block*

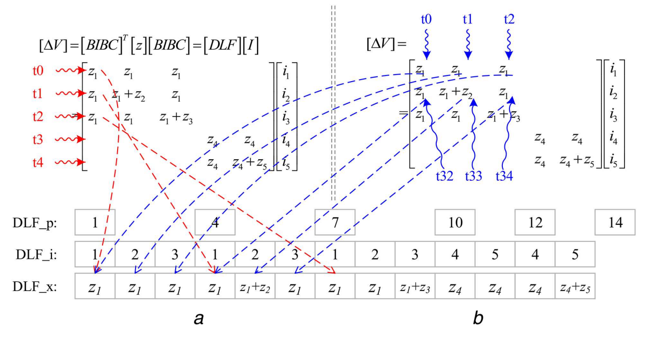


Fig. 9 *Thread organisation for parallel matrix–vector multiplication within each block*
(a) One thread per row (scattered access), ***(b)*** One warp per row (coalesced access)

switchable stages for each SC is 5, and the stage step is 0.04 MVar. The number of OLTC and medium-length line equal to 10 and 1% of the total number of branches, respectively. Each OLTC has a tap ratio from 0.9 to 1.1 and can be divided into 21 steps. Table 2 summarises the basic configuration of each system with the number of components. All the detailed data is randomly generated according to Table 2, such as where to install these components. Since different types of implementation utilise the same randomly generated input data, the comparison on solution accuracy and efficiency is reasonable. It should be noted that only RTVVO module (i.e. the decision-making process) shown in Fig. 1 is considered in case studies, whereas communication between RTVVO module and network components are out of the scope of this paper, which means the component data/status is directly generated and given as input for RTVVO module without the utilisation of AMI aggregator.

In order to validate the performance of the GPU-based parallel RTVVO, four types of implementation have been carried out for comparison:

- *CPU_M:* PSO framework given in Fig. 4, where the fitness evaluation of each particle is performed by Matpower with the built-in Newton–Raphson method;
- *CPU_S:* sequential version of Algorithm 1 in CPU with C++;
- *CPU_P:* parallel version of Algorithm 1 in CPU with OpenMP, where 12 threads are launched;
- *GPU_P:* parallel version of Algorithm 1 in GPU with CUDA.

For each implementation, the population size of PSO is set as 512, and the maximum number of iteration is 200. All tests are implemented on the same platform including: Intel Xeon E5-2620 CPU with 32 GB RAM, Nvidia GeForce GTX 1080 GPU, Matlab version 2017a, CUDA version 8.0, and Visual Studio 2015.

5.1 Solution validation

Although the main objective of this paper is achieving high solution efficiency from parallel processing with GPU, the accuracy and convergence property of RTVVO should be guaranteed. Without loss of generality, the 1760-bus system is employed for validation.

5.1.1 Accuracy: In this work, the distribution network power flow should be calculated for thousands of times, thus its accuracy must be validated. In Section 2, formulations of DG, OLTC, and SC are integrated into the solution framework of DA method. Therefore, open source package Matpower is introduced to validate the accuracy of DA. On the other hand, parallel implementation of DA on GPU may also introduce errors, thus the comparison between sequential and parallel is implemented. In this test, 20 PSO particles are randomly generated and solved with either DA or Matpower in all four implementation environments. The average difference on the obtained active power loss for different methods is reported in Table 3. It can be seen that the error between Matpower and DA is smaller than 10^{-6}, which means the accuracy of the developed DA is acceptable. On the other hand, the difference of DA running on different platforms is smaller than 10^{-8}, indicating that parallel implementation does not spoil the accuracy.

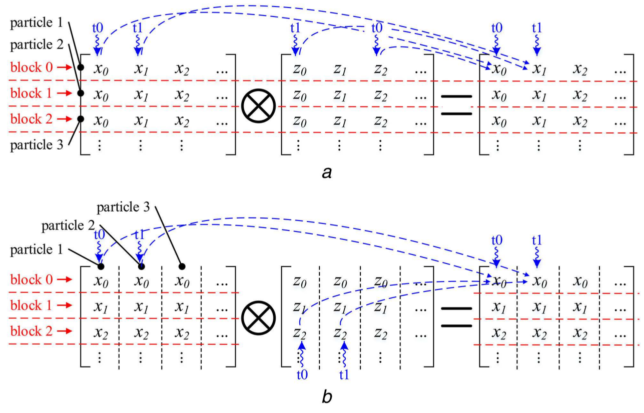


Fig. 10 *Thread organisation for parallel irregular mapping*
(a) One block per particle (scattered access), ***(b)*** One block per element (coalesced access)

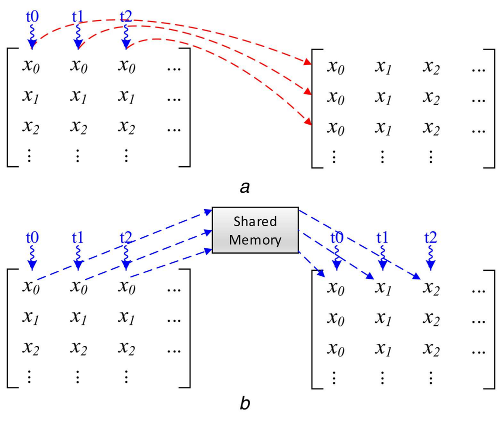


Fig. 11 *Thread organisation for parallel matrix transpose*
(a) Naive matrix transpose (scattered access), ***(b)*** Shared memory harnessed transpose (coalesced access)

5.1.2 Convergence property: According to the above accuracy analysis, it can be concluded that the difference in the power flow calculation is negligible. Thus, the convergence test is devoted to the validation of the PSO framework. Based on the same input data and PSO parameters, the convergence property of two types of implementation CPU_M and GPU_P corresponding to iteration is demonstrated in Fig. 12. Although there are fluctuations in each line, the convergence property is similar: (i) the objective value drops very fast in the first 40 iterations, indicating that the PSO is effective to find a high-quality solution for this problem; (ii) the improvement during the next 40 iteration is still observable, which is due to the disturbance introduced in (25); (iii) finally, the population turns to be stable in the remaining iterations, showing that the particles searching around the obtained global best optimal.

5.2 Solution efficiency

In this subsection, the improvement on the solution efficiency gained by two proposals given in Section 4 will be exemplified on four test systems.

5.2.1 Performance improvement gained by data structure design: As shown in Fig. 6, the number of atom operations required by Algorithm 2 is smaller than the matrix–matrix multiplication. Quantitative results on four large systems are given in Table 4. It can be seen that the data structure redesign is beneficial to reduce over 50% of the atom operations for all test systems although the sparsity of [**BIBC**] and [**DLF**] are different. In addition to the superiority of fewer atom operations, the number of index calculation when traversing CSR matrices is smaller for Algorithm 2. Therefore, Algorithm 2 is at least two times faster than the matrix–matrix multiplication.

Table 2 Number of components for different test systems

Cases	Branches	DG	SC	OLTC	Medium-length line
136-bus	135	12	4	14	1
415-bus	415	86	52	42	4
880-bus	873	74	37	87	9
1760-bus	1746	150	74	175	17

Table 3 Average active power loss error between different implementations for the 1760-bus system in 20 trials

Methods	CPU_M	CPU_S	CPU_P	GPU_P
CPU_M	0	4.91×10^{-7}	4.56×10^{-7}	9.16×10^{-7}
CPU_S	4.91×10^{-7}	0	6.21×10^{-9}	8.13×10^{-9}
CPU_P	4.56×10^{-7}	6.21×10^{-9}	0	2.79×10^{-9}
GPU_P	9.16×10^{-7}	8.13×10^{-9}	2.79×10^{-9}	0

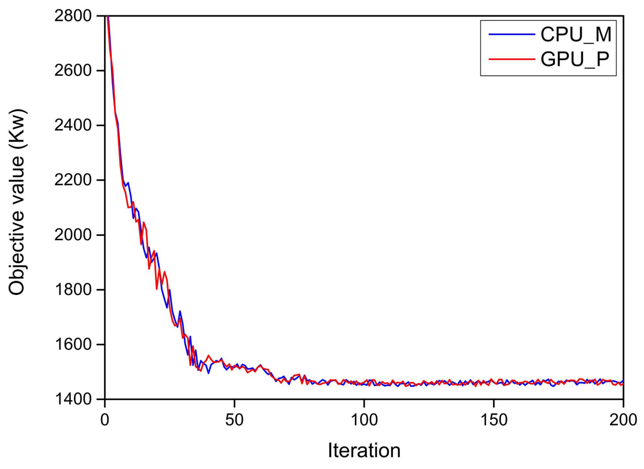


Fig. 12 *Convergence property of two types of implementations CPU_M and GPU_P*

5.2.2 Performance improvement gained by GPU parallel implementation: In order to validate the performance of various methods, all of them are executed for 20 times with the same input data. One of them has been demonstrated with Fig. 12. The average execution time for different methods is collected in Table 5. It can be seen that the CPU_M takes much longer time than the other methods for all test systems. The reason is two-fold: (i) the complexity of Newton–Raphson method is higher than DA; (ii) the execution efficiency of C++ code is higher than the Matlab code. Although the CPU_S is much faster than CPU_M, its application in real-time circumstance is still questionable since minutes of time is required. According to the data reported in Table 5, the capability of CPU_P and GPU_P for practical application is promising. A similar conclusion can also be drawn if the average ACPF execution time is considered.

Taking the execution time of CPU_M for different systems as the basis, the achieved speedup can be obtained as shown in Table 6. The superiority of DA over Matpower in this kind of problem is established. Consider CPU_S as the benchmark, the speedup is reported in Table 7. It can be seen that the parallel efficiency of OpenMP and CUDA are satisfactory with ranges from 10.20 to 11.40 and from 19.97 to 62.28, respectively. In addition, the speedup goes higher as the system scale increases.

5.3 Discussion

One of the main contributions of this paper is the parallel calculation of large numbers of distribution network power flows with DA method. Its performance within PSO framework has been illustrated in the above tests. In this section, we intend to validate that the proposal is also beneficial for other types of population-based meta-heuristic algorithm frameworks, such as GA and ant colony optimisation (ACO). Within various framework, although

Table 4 Number of atom operations for updating of [DLF] with two different methods

Cases	Dimensions	[BIBC]		[DLF]		Atom operations		Improvement
		Non-zeros	Sparsity	Non-zeros	Sparsity	M–M multiplication	Algorithm 2	
136-bus	135	976	5.36%	2,465	13.53%	26,504	12,746	51.91%
415-bus	415	1,950	1.13%	3,585	2.08%	28,010	13,030	53.48%
880-bus	873	23,147	3.04%	125,285	16.44%	4,220,153	2,098,503	50.27%
1760-bus	1746	46,294	1.52%	250,570	8.22%	8,440,306	4,197,006	50.27%

Table 5 Execution time of the RTVVO for various implementation schemes

Cases	Total RTVVO execution time, s				Average execution time for single ACPF, ms			
	CPU_M	CPU_S	CPU_P	GPU_P	CPU_M	CPU_S	CPU_P	GPU_P
136-bus	574.46	47.34	4.64	2.37	5.61	0.46	0.045	0.023
415-bus	1295.36	96.36	8.95	3.06	12.65	0.94	0.087	0.030
880-bus	2744.32	184.88	16.77	3.80	26.80	1.81	0.164	0.037
1760-bus	5149.69	336.92	29.55	5.41	50.29	3.29	0.289	0.053

Table 6 Achieved speedup over CPU_M for various methods

Cases	CPU_M	CPU_S	CPU_P	GPU_P
136-bus	1.00	12.13	123.81	242.39
415-bus	1.00	13.44	144.73	423.32
880-bus	1.00	14.84	163.64	722.19
1760-bus	1.00	15.28	174.27	951.88

Table 7 Achieved speedup over CPU_S for various methods

	CPU_M	CPU_S	CPU_P	GPU_P
136-bus	0.082	1.00	10.20	19.97
415-bus	0.074	1.00	10.77	31.49
880-bus	0.067	1.00	11.02	48.65
1760-bus	0.065	1.00	11.40	62.28

the evolutionary strategy and process are different, the fitness evaluation step are the same. Actually, it is the essential component of these three algorithms and dominates the efficiency of each algorithm, therefore, lines 6–16 in Algorithm 1 are integrated into GA and ACO frameworks as an independent module. Different from the above tests, the termination criteria is changed, i.e. the solution process will be terminated if the global best value has not been updated for 30 iterations.

Fig. 13 illustrates the main results. It can be seen from Fig. 13*a* that the PSO gains smaller power loss than GA and ACO, showing that the velocity updating strategy 25 is beneficial to increase the PSO global searching capability. The objective function values of GA and ACO are similar, which means there are several local optimal solutions, and both of them cannot jump out to get a lower power loss without specific improvements. Fig. 13*b* reports the number of iterations utilised before termination. PSO utilises the largest numbers of iterations due to the continuous updating of objective function values, whereas GA and ACO are terminated due to the lack of diversity and searching capability. Fig. 13*c* shows similar trends with Fig. 13*b*, which means the execution time is proportional to the number of iterations, i.e. lines 6–16 in Algorithm 1 gains the same efficiency in different frameworks. Therefore, it can be concluded that the proposal is suitable for various types of population-based meta-heuristic algorithms. It should be noted that, although PSO consumed the longest time, it gained the best objective value. Given that the GPU_P has gained satisfactory performance in the above sections, the optimality is of higher priority in this section since all three algorithms equipped with the same kernel and running on the same platform. Therefore, to sum up, the GPU_P gains a better trade-off between efficiency and optimality than GA and ACO.

6 Conclusion

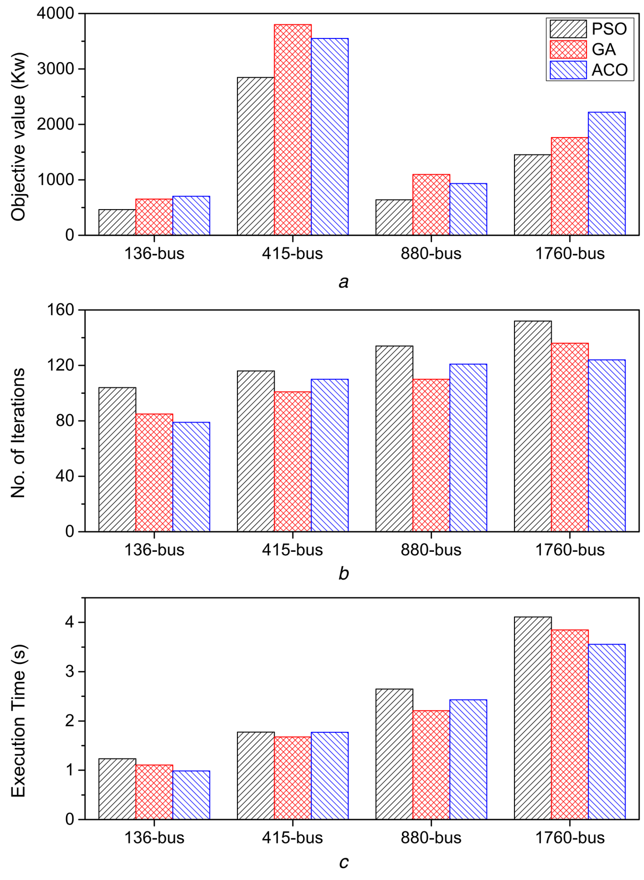


Fig. 13 *Performance comparison between PSO, GA, and ACO*

The popularity of RTVVO is increasing with the widely utilisation of AMI; nevertheless, the computation efficiency is still not sufficient to meet with the practical requirements, e.g. making the decision within 10 s. This paper intends to enhance the performance via parallel processing with GPU. PSO is employed as the solution framework for RTVVO, where the full ACPF is tackled with the DA method. In the iterative process of DA, the detailed mathematical models for DG and other VVO control devices are integrated. Although PSO and DA are suitable for parallel processing, the best performance is far away to be reached by the naive implementation due to the random access of addresses, therefore designs on data structure and thread organisation are proposed. After tuning, all threads in one warp are assigned to access the successive address, i.e. coalesced access is achieved. In the case study, four systems with the size ranging from 136-bus to 1760-bus are introduced. The results indicate that the accuracy and convergence property is satisfactory, and the parallel efficiency is suitable for practical application.

Future work will consider more practical concerns, such as the device action time, communication bandwidth, AMI aggregation process, and so on. In addition, various types of distribution networks will be investigated, such as weakly meshed and three-phase unbalanced systems. For the latter system, more complicated components are required to be formulated, such as the phase shifting transformer. On the other hand, the hot start strategy is also possible to accelerate the solution process since the environment variation between two successive decision points is minor.

7 Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). Shengjun Huang was sponsored by the China Scholarship Council (CSC) under grant no. 201403170337.

8 References

[1] Zhu, J.: '*Optimization of power system operation*' (John Wiley & Sons, New Jersey, 2015, 2nd edn.)

[2] Ahmadi, H., Marti, J. R., Dommel, H. W.: 'A framework for volt-VAR optimization in distribution systems', *IEEE Trans. Smart Grid*, 2015, **6**, (3), pp. 1473–1483

[3] Saric, A. T., Stankovic, A. M.: 'A robust algorithm for volt/Var control'. Proc. Power Syst. Conf. Expo., Seattle, WA, USA, Mar. 2009, pp. 1–8

[4] Zheng, W., Wu, W., Zhang, B.*, et al.*: 'Robust reactive power optimisation and voltage control method for active distribution networks via dual time-scale coordination', *IET Gener. Transm. Distrib.*, 2017, **11**, (6), pp. 1461–1471

[5] Sayadi, F., Esmaeili, S., Keynia, F.: 'Two-layer volt/var/total harmonic distortion control in distribution network based on PVs output and load forecast errors', *IET Gener. Transm. Distrib.*, 2017, **11**, (8), pp. 2130–2137

[6] Mohapatra, A., Bijwe, P. R., Panigrahi, B. K.: 'An efficient hybrid approach for volt/Var control in distribution systems', *IEEE Trans. Power Deliv.*, 2014, **29**, (4), pp. 1780–1788

[7] Rahimi, S., Zhu, K., Massucco, S.*, et al.*: 'Stochastic volt-Var optimization function for planning of MV distribution networks'. Proc. IEEE Power Energy Soc. Gen. Meeting, Denver, CO, USA, Jul. 2015, pp. 1–5

[8] Borghetti, A.: 'Using mixed integer programming for the volt/var optimization in distribution feeders', *Electr. Power Syst. Res.*, 2013, **98**, pp. 39–50

[9] Fang, X., Li, F., Wei, Y.*, et al.*: 'Reactive power planning under high penetration of wind energy using benders decomposition', *IET Gener. Transm. Distrib.*, 2015, **9**, (14), pp. 1835–1844

[10] Wang, Z., Wang, J., Chen, B.*, et al.*: 'MPC-based voltage/var optimization for distribution circuits with distributed generators and exponential load models', *IEEE Trans. Smart Grid*, 2014, **5**, (5), pp. 2412–2420

[11] Malachi, Y., Singer, S.: 'A genetic algorithm for the corrective control of voltage and reactive power', *IEEE Trans. Power Syst.*, 2006, **21**, (1), pp. 295–300

[12] Ulinuha, A., Masoum, M., Islam, S.: 'Hybrid genetic-fuzzy algorithm for volt/var/total harmonic distortion control of distribution systems with high penetration of non-linear loads', *IET Gener. Transm. Distrib.*, 2011, **5**, (4), pp. 425–439

[13] Fukuyama, Y.: 'Parallel particle swarm optimization for reactive power and voltage control verifying dependability'. Proc. IEEE Congr. Evol. Comput., Sendai, Japan, May 2015, pp. 304–310

[14] Niknam, T., Firouzi, B. B., Ostadi, A.: 'A new fuzzy adaptive particle swarm optimization for daily volt/Var control in distribution networks considering distributed generators', *Appl. Energy*, 2010, **87**, (6), pp. 1919–1928

[15] Chaudhary, D., Sun, W., Zhou, Q.*, et al.*: 'Chance-constrained real-time volt/var optimization using simulated annealing'. Proc. IEEE Power Energy Soc. Gen. Meeting, Denver, CO, USA, Jul. 2015, pp. 1–5

[16] Zakariazadeh, A., Modaghegh, H., Jadid, S.: 'Real time volt/Var control using advance metering infrastructure system in FAHAM project'. Proc. Int. Conf. and Exhibition on Electricity and Distribution, Stockholm, Sweden, Jun. 2013, pp. 1–4

[17] Manbachi, M., Sadu, A., Farhangi, H.*, et al.*: 'Real-time co-simulation platform for smart grid volt-var optimization using IEC 61850', *IEEE Trans. Ind. Inf.*, 2016, **12**, (4), pp. 1392–1402

[18] Manbachi, M., Sadu, A., Farhangi, H.*, et al.*: 'Real-time co-simulated platform for novel volt-VAR optimization of smart distribution network using AMI data'. Proc. IEEE Int. Conf. Smart Energy Grid Eng., Oshawa, Canada, Aug. 2015, pp. 1–7

[19] Feng, X., Peterson, W., Yang, F.*, et al.*: 'Smarter grids are more efficient', *ABB Rev.*, 2009, **3**, pp. 33–37

[20] Chandra, R., Dagum, L., Kohr, D.*, et al.*: '*Parallel programming in OpenMP*' (Morgan Kaufmann, San Francisco, 2001)

[21] NVIDIA: '*CUDA c programming guide 8.0*' (NVIDIA Corporation, Santa Clara, CA, USA, 2017)

[22] Jalili-Marandi, V., Zhou, Z., Dinavahi, V.: 'Large-scale transient stability simulation of electrical power systems on parallel GPUs', *IEEE Trans. Parallel Distrib. Syst.*, 2012, **23**, (7), pp. 1255–1266

[23] Zhou, Z., Dinavahi, V.: 'Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large power systems', *IEEE Power Energy Tech. Syst. J.*, 2017, **4**, (3), pp. 51–64

[24] Yan, S., Zhou, Z., Dinavahi, V.: 'Large-scale nonlinear device-level power electronic circuit simulation on massively parallel graphics processing architectures', *IEEE Trans. Power Electron.*, 2017, **PP**, (99), pp. 1–19

[25] Zhou, G., Feng, Y., Bo, R.*, et al.*: 'GPU-accelerated batch-ACPF solution for N-1 static security analysis', *IEEE Trans. Smart Grid*, 2017, **8**, (3), pp. 1406–1416

[26] Huang, S., Dinavahi, V.: 'Fast batched solution for real-time optimal power flow with penetration of renewable energy', *IEEE Access*, 2018, **PP**, (99), pp. 1–13

[27] Karimipour, H., Dinavahi, V.: 'Parallel relaxation-based joint dynamic state estimation of large-scale power systems', *IET Gener. Transm. Distrib.*, 2016, **10**, (2), pp. 452–459

[28] Karimipour, H., Dinavahi, V.: 'Extended Kalman filter-based parallel dynamic state estimation', *IEEE Trans. Smart Grid*, 2015, **6**, (3), pp. 1539–1549

[29] Teng, J.: 'A direct approach for distribution system load flow solutions', *IEEE Trans. Power Deliv.*, 2003, **18**, (3), pp. 882–887

[30] Cano, J., Mojumdar, M. R., Norniella, J. G.*, et al.*: 'Phase shifting transformer model for direct approach power flow studies', *Int. J. Electr. Power Energy Syst.*, 2017, **91**, pp. 71–79

[31] Zimmerman, R., Murillo-Sanchez, C., Thomas, R.: 'MATPOWER: steady-state operations, planning, and analysis tools for power systems research and education', *IEEE Trans. Power Syst.*., 2011, **26**, (1), pp. 12–19

[32] Teng, J.: 'Modeling distributed generations in three-phase distribution load flow', *IET Gener. Transm. Distrib.*, 2008, **2**, (3), pp. 330–340

[33] Murugan, P: 'Modified particle swarm optimisation with a novel initialisation for finding optimal solution to the transmission expansion planning problem', *IET Gener. Transm. Distrib.*, 2012, **6**, (11), pp. 1132–1142

[34] Huang, S., Dinavahi, V.: 'Multi-group particle swarm optimization for transmission expansion planning solution based on LU decomposition', *IET Gener. Transm. Distrib.*, 2017, **11**, (6), pp. 1434–1442

[35] Davis, T.A.: '*Direct methods for sparse linear systems*' (Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2006)

[36] Roberge, V.: 'Distribution feeder reconfiguration (DFR) test cases', http://roberge.segfaults.net/joomla/index.php/dfr, (accessed July 2017)