IET Generation, Transmission & Distribution

Research Article

IET Journals
The Institution of Engineering and Technology

Fast distribution network reconfiguration with graph theory

ISSN 1751-8687
Received on 5th February 2018
Revised 22nd March 2018
Accepted on 1st April 2018
E-First on 22nd May 2018
doi: 10.1049/iet-gtd.2018.0228
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: Owing to mixed-integer and non-linear properties, the distribution network reconfiguration (DNRC) problem has been widely addressed with meta-heuristic algorithms. To accelerate the solution process, two essential components of meta-heuristic algorithms are investigated in this study: solution representation and fitness evaluation. Instead of the popular binary and integer numbers, decimal encoding is employed. Decoding is based on the proposed probability-based loop destruction strategy. The fitness evaluation is based on the power flow calculation of radial network. Different from backward/forward sweep method, the advantageous direct solution technique is utilised, where the matrix generation process has been accelerated. Both improvements are based on the graph theory and fully explained with illustrative examples. Case studies are implemented on five benchmark systems. The superiority of the proposed methods over their advanced counterparts has been established with intensive comparisons. Finally, these methods are integrated into a standard particle swarm optimisation framework for the solution of DNRC. Results indicate that the proposals significantly improve the solution efficiency without the loss of quality.

Nomenclature

BCBV	branch-current to bus-voltage
BFS	backward/forward sweep
BIBC	bus-injection to branch-current
BRD	branch-based reachability detection
DA	direct approach
DN	distribution network
DNPF	distribution network power flow
DNRC	distribution network reconfiguration
FD	fast decoupled
GA	genetic algorithm
GS	gauss-Seidel
MINLP	mixed-integer non-linear programming
MRD	matrix-based reachability detection
MST	minimum spanning tree
NR	Newton-Raphson
PLD	probability-based loop destruction
PSO	particle swarm optimisation
RTS	radial tree structure
MILP	mixed-integer linear programming

1 Introduction

As the most intensive part of power systems, the distribution network (DN) is directly connected to large numbers of consumers, where unexpected failures and load fluctuations are inevitable. To deal with these uncertainties, flexibility is indispensable. Generally, the DN is built as interconnected with switches, while radial tree structure (RTS) is maintained in operation [1]. The radial tree is obtained from the opening of switches, and the transformation of one tree into another is possible with the adjusting of on/off status of switches; therefore, the flexibility is achieved. Finding an optimal RTS from the DN with specified objective while satisfying an operating constraint is classified as the DN reconfiguration (DNRC) problem. Owing to low-voltage levels, the DN produces a large number of power losses, thus minimising system power loss comprises the major objective of DNRC in this work. Other goals considered in the literature include high reliability [2] and smooth voltage profile [3].

In addition to the global optimality of the final solution, execution time is of great significance for DNRC. For any decision-making problem, the obtained solution is optimal only when the input status has not changed; otherwise, it might be suboptimal or invalid. Since the customer action keeps changing, the solution time of DNRC should be minimised to preserve the optimality. Nevertheless, the solution process of DNRC is not trivial. Actually, it is a mixed-integer non-linear programming (MINLP) problem with combinatorial property [4]. Linearisation techniques [5–8] are widely utilised to formulate the MINLP into an MINLP problem, where deterministic methods [9–11] and heuristic strategies [12–17] are available. However, the accuracy might be sacrificed due to the utilisation of simplification and relaxation. On the other hand, meta-heuristic algorithms [1, 4, 18, 19] are capable to accurately solve the MINLP problem. To achieve satisfactory optimality and efficiency from the solution process of DNRC with the computationally intensive meta-heuristic algorithm, the following two major concerns should be fully addressed:

- *RTS:* Generally, the DN is described as a graph $G = \{V(G), E(G)\}$, where $V(G)$ and $E(G)$ are finite sets containing vertices and edges. Thus, the DNRC problem is to find out an optimal tree $T = \{V(T), E(T)\}$ with respect to specific goals, where $V(T) = V(G)$ and $E(T) \subseteq E(G)$. In the meta-heuristic algorithm, T is described as the individual and evolutionary operation is conducted on it to achieve better individuals. However, the newly obtained T might be infeasible since all elements in $V(T)$ should be connected and no cycles are permitted in T. Generally, graph theory [20] should be followed for the generation of T.
- *DN power flow (DNPF):* The values of resistance R and reactance X in the DN are of a great range, which may result in a big ratio of R/X. In this case, the efficient fast decoupled (FD) PF algorithm may fail to converge since the major assumption $R \ll X$ is violated [1]. On the other hand, it is complicated and time-consuming to introduce the Gauss–Seidel or Newton–Raphson (NR) methods for the solution of DNPF due to the RTS of DN [4].

The key point to address RTS concern is guaranteeing all the generated individuals are feasible. The infeasible individual will destroy the evolutionary process since the fitness value cannot be identified, i.e. the evolutionary direction cannot be evaluated. In the literature, two types of heuristics are widely investigated to obtain feasible individuals:

- *Modification:* This kind of method starts from an RTS tree T. To obtain a new T, only two steps are required. First, select one edge e_i in $E(G)\setminus E(T)$ and add it into $E(T)$. Owing to RTS feature and graph theory, the resulted T does not hold RTS since a cycle is formulated by e_i. Second, in order to obtain RTS again, select one edge e_j belonging to the cycle and add it into $E(G)$. This method is named *branch exchange* in the literature, i.e. exchanging $e_j \in E(T)$ with $e_i \in E(G)\setminus E(T)$. Within this framework, different strategies are proposed [12–14] to determine e_i and e_j.
- *Generation:* This type of method starts from the full graph G. A successive process should be implemented to generate a candidate RTS. At each step, identify one cycle in G, and then select one edge e_i in that cycle to delete. Each step means breaking a circle/loop. This process continues until there is no cycle in G. On the basis of how to select e_i in each loop, different methods are developed [15–17, 21].

Although these heuristics are straightforward and easy to implement, they are greedy and the final solution depends on the initial configuration [1], thus it is a local optimum rather than the global best. Therefore, they are commonly integrated into a meta-heuristic framework to achieve the global optima. Nevertheless, the encoding and decoding strategies are usually complicated or inefficient due to the identification of different circles. For example, the integer coded genetic algorithm is utilised in [18], where crossover and mutation operations are based on the matroid theory, which is highly dependent on the circles. Instead of dealing with loops, a novel decimal encoding technique is proposed in [4], where the RTS is naturally guaranteed for each candidate based on the minimum spanning tree (MST) calculation. Given an undirected graph G whose branches are weighted with decimal numbers, different trees can be generated by advanced MST algorithms. This method is direct, but the solution process is time-consuming because the MST searching is computationally intensive and should be executed thousands of times. In this paper, a new encoding and decoding strategy [(probability-based loop destruction (PLD)] is proposed based on decimal numbers. Instead of the heavy MST computation, only simple operations are involved in the PLD method.

To address the DNPF concern shown above, techniques widely utilised for transmission network have been modified and applied to accommodate DN such as the FD [22–24] and NR [25–27] algorithms. One limitation of this kind of method is that the successive admittance matrices must be updated in each iteration due to the topology variation. The construction and factorisation of these matrices bring heavy computation burden; thus, the solution efficiency is limited. On the other hand, the backward/forward sweep (BFS) method [28] and its variants [19, 29, 30] have gained great popularity for the DNPF solution due to their good convergence and easy applicability. One major drawback of the BFS as summarised in the literature [31] is the requirement on the numbering schemes for the system buses and/or branches, which could reduce the flexibility and affect its ability to accommodate changes in topology. In addition, the solution time of DNPF with BFS is mainly determined by the number of system buses, thus the capability for large-scale systems is restrained.

Instead of the time-consuming LU decomposition and backward/forward substitution, a distinctive direct approach (DA) was proposed in [32], where only the matrix multiplications were involved. The essential processes related to two newly defined matrices, i.e. the bus-injection to branch-current (BIBC) matrix and the BC to bus-voltage (BCBV) matrix. The advantages of DA has been revealed in [32, 33]. Although the formulation algorithm [branch-based reachability detection (BRD)] of BIBC and BCBV reported in [32] is intuitive, the efficiency is limited since only one branch is considered at each step. In this paper, a novel algorithm [matrix-based RD (MRD)] for the construction of BIBC and BCBV is developed based on the graph theory, where all switches are considered concurrently by adjacency matrix and path matrix.

In summary, two improvement proposals are developed in this paper to address two major concerns of a meta-heuristic algorithm for the solution of DNRC, i.e. PLD and MRD for RTS and DNPF, respectively. To validate the accuracy and efficiency, both PLD and MRD, as well as their advanced counterparts MST and BRD, are coded and integrated into a standard particle swarm optimisation (PSO) framework provided by the MATLAB global optimisation toolbox [34]. Five benchmark systems are introduced for implementation and comparison. The PLD is $27.02\times$ faster than MST method for the decoding of 10,000 individuals. Compared to BRD, MRD can reduce the execution time by 62.93%. In addition to the efficiency, PLD is also beneficial on the convergence property. The results indicate that the superiority of the proposed strategies is significant within the evolutionary computing framework. It is believable that their application on other algorithm frameworks such as deterministic and heuristic are also promising.

The rest of this paper is organised as follows. Section 2 is devoted to the concerns of RTS, where the detailed steps of PLD are described and explained. To achieve high efficiency on the solution of DNPF, the MRD is proposed and embedded within the DA framework in Section 3. Case studies and discussion are provided in Section 4. Finally, Section 5 concludes this paper.

2 Solution encoding and decoding strategies

The DNRC solution consists of a series of open branches, which can be intuitively represented with binary and integer numbers. Although both of them are straightforward and easy to implement, the radial topology of the network cannot be maintained efficiently, which results in a large number of infeasible solutions during the evolutionary process; therefore, the convergence property is limited and the capability for the large-scale system solution is weak. To alleviate these concerns, a novel encoding technique was developed in [4], where each branch corresponds to one element in the solution vector and assigned with a real number. When decoding, the real number corresponding to each branch is regarded as the weight, thus a weighted undirected graph was established. By doing an MST computation, the radial topology can be uniquely determined. This method guarantees all solutions are feasible, i.e. the RTS is always preserved.

Although there are several advanced algorithms for MST computing such as Prim's, Kruskal's, and Boruvka's algorithms, the computational complexity is still high. In addition, the MST calculation should be performed for thousands of times, corresponding to the generation of each candidate. To alleviate the computational burden and improve the solution efficiency, a two-stage probability-based encoding and decoding processes are developed in this paper.

2.1 Stage 1: network analysis

As a preliminary process, this stage is independent of decoding process and will be executed for only once. The main objective is to find out the shortest cycle for each tie switch and organise them in a specified order. On the basis of this order, Stage 2 is designed to break these cycles and generate the RTS. For simplicity, a small-scale distribution system is introduced for illustration, which is shown in Fig. 1. The target network consists of $n_d = 14$ nodes, $n_b = 13$ branches, and $n_s = 3$ tie switches (indicated by dashed lines). The following steps are responsible for network analysis, where step-by-step temporary results corresponding to Fig. 1 are also revealed:

- *Step 1:* Open all the switches to generate a radial tree. The result is indicated by Fig. 2*a*.
- *Step 2:* Close each tie switch to find a corresponding cycle. This step can be fulfilled by calculating the shortest path between two

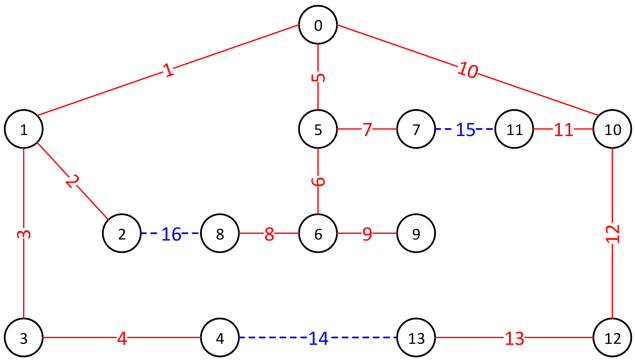
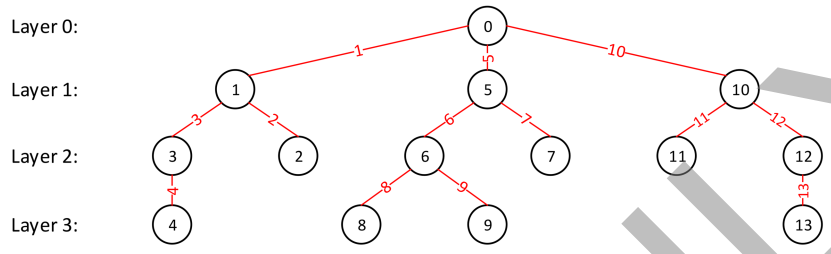


Fig. 1 *Configuration of the target DN*

(a)



(b)

Switch ID	Circle (with branch ID)	Circle (with node ID)
14	4 – 3 – 1 – 10 – 12 – 13 – **14**	4 – 3 – 1 – 0 – 10 – 12 – 13 – 4
15	7 – 5 – 10 – 11 – **15**	7 – 5 – 0 – 10 – 11 – 7
16	2 – 1 – 5 – 6 – 8 – **16**	2 – 1 – 0 – 5 – 6 – 8 – 2

(c)

C: 4 – 3 – 1 – 10 – 12 – 13 – **14** – 7 – 5 – 10 – 11 – **15** – 2 – 1 – 5 – 6 – 8 – **16**

D: **0** **7** **12** **18**

Fig. 2 *Intermediate results of the network analysis*

(a) generated radial tree structure by the opening of tie switches, ***(b)*** identified cycles corresponding to each tie switch with shortest path calculation, ***(c)*** organized vectors for cycles and indices

nodes of each tie switch in the radial tree. Fig. 2*b* illustrates these cycles with responding to both branch and node ID.

- *Step 3:* Join the cycles with branch ID into one whole vector $\boldsymbol{C}$, and determine the index of each switch in $\boldsymbol{C}$ to generate another vector $\boldsymbol{D}$. Finally, insert a 0 at the beginning of $\boldsymbol{D}$. Fig. 2*c* demonstrates the ultimate results of Stage 1.

Obviously, the number of cycles is n_s; therefore, $\boldsymbol{D}$ is of length $n_s + 1s$. On the other hand, the length of $\boldsymbol{C}$ is problem dependent due to branch reputation at different cycles. To sum up, the input of network analysis is the initial configuration, and the outputs are vectors $\boldsymbol{C}$ and $\boldsymbol{D}$.

2.2 Stage 2: solution representation

On the basis of vectors $\boldsymbol{C}$ and $\boldsymbol{D}$, this stage illustrates how to encode and decode decimal solution vector $\boldsymbol{R}$.

2.2.1 Encoding: To determine whether a branch should be open or close, a probability value $r_i \in [0, 1]$ is granted for each branch in this paper; therefore, the $1 \times (n_b + n_s)$ decimal solution vector $\boldsymbol{R}$ can be encoded intuitively as

$$\boldsymbol{R} = \{r_1, r_2, \ldots, r_{n_b}, \ldots, r_{n_b+n_s}\}.$$

2.2.2 Decoding: Given a real number encoded solution $\boldsymbol{R}$ as shown in Fig. 3*a*, the PLD decoding processes are developed as follows:

- *Step 1:* Generate a $1 \times n_s$ temporary vector $\boldsymbol{T}$, whose elements are corresponding to the probability of each tie switch in $\boldsymbol{R}$.
- *Step 2:* Sort $\boldsymbol{T}$ in ascending order and store the $1 \times n_s$ indexes vector as $\boldsymbol{P}$. Set $k = 0$.
- *Step 3:* Let $k = k + 1$, find out the $\boldsymbol{P}_k$th cycle based on $\boldsymbol{C}$ and $\boldsymbol{D}$.
- *Step 4:* Look up the probability value in $\boldsymbol{R}$ for each branch of the $\boldsymbol{P}_k$th cycle.
- *Step 5:* Select the branch with the largest probability value as the one for breaking, which is marked as $\boldsymbol{S}_k$.
- *Step 6:* Update the probability value of the $\boldsymbol{P}_k$th cycle in $\boldsymbol{R}$ as −1.00.
- *Step 7:* If $k < n_s$, go back to *Step 3*; otherwise, output the decoded solution $\boldsymbol{S}$.

Temporary results from *Steps 1 and 2* are illustrated in Fig. 3*b*, while other results are demonstrated in Fig. 3*c*.

2.3 Supplementary explanation

The basic idea of PLD is *branch exchange*: Stage 1 is utilised to find the cycle formulated by closing one switch, and that loop is destroyed in Stage 2 by opening one branch based on the probability. All cycles generated from Stage 1 are stored in two vectors $\boldsymbol{C}$ and $\boldsymbol{D}$ with respect to the specified order. Intuitively, these circles can be broken with the same order for all decoding process. Nevertheless, this will destroy the randomness and restrict the solution space. For example, if the destruction of the cycle $\{4-3-1-10-12-13-14\}$ is always earlier than a cycle $\{7-5-10-11-15\}$, then the branch 10 in the second cycle will never be broken since its probability is updated into −1.00 after the destruction of the first cycle. To address this concern, the breaking of different cycles should be conducted in a random order, that is, the reason to introduce the vector $\boldsymbol{P}$ in *Steps 3–6*. In terms of how

(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\boldsymbol{R}$:	0.44	0.41	0.25	0.71	0.78	0.24	0.81	0.94	0.89	0.07	0.99	0.12	0.74	0.63	0.12	0.73

(b)

Swithch ID:	14	15	16										Sorted $\boldsymbol{T}$:	0.12	0.63	0.73
$\boldsymbol{T}$:	0.63	0.12	0.73										$\boldsymbol{P}$:	2	1	3

(c)

$k=1$ $\boldsymbol{P}_k=2$ **Circle:** 7 – 5 – 10 – 11 – 15 **Prob:** 0.81, 0.78, 0.07, 0.99, 0.12 **Max:** 0.99 $\boldsymbol{S}_k=11$																
$\boldsymbol{R}$:	0.44	0.41	0.25	0.71	**-1.00**	0.24	**-1.00**	0.94	0.89	**-1.00**	**-1.00**	0.12	0.74	0.63	**-1.00**	0.73
$k=2$ $\boldsymbol{P}_k=1$ **Circle:** 4 – 3 – 1 – 10 – 12 – 13 – 14 **Prob:** 0.71, 0.25, 0.44, -1.00, 0.12, 0.74, 0.63 **Max:** 0.74 $\boldsymbol{S}_k=13$																
$\boldsymbol{R}$:	**-1.00**	0.41	**-1.00**	**-1.00**	**-1.00**	0.24	**-1.00**	0.94	0.89	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	0.73
$k=3$ $\boldsymbol{P}_k=3$ **Circle:** 2 – 1 – 5 – 6 – 8 – 16 **Prob:** 0.41, -1.00, -1.00, 0.24, 0.94, 0.73 **Max:** 0.94 $\boldsymbol{S}_k=8$																
$\boldsymbol{R}$:	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	0.89	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**	**-1.00**

Fig. 3 *Encoded real number solution vector and its decoding*
(a) real number encoded solution, ***(b)*** decoding preparation, ***(c)*** decoding process

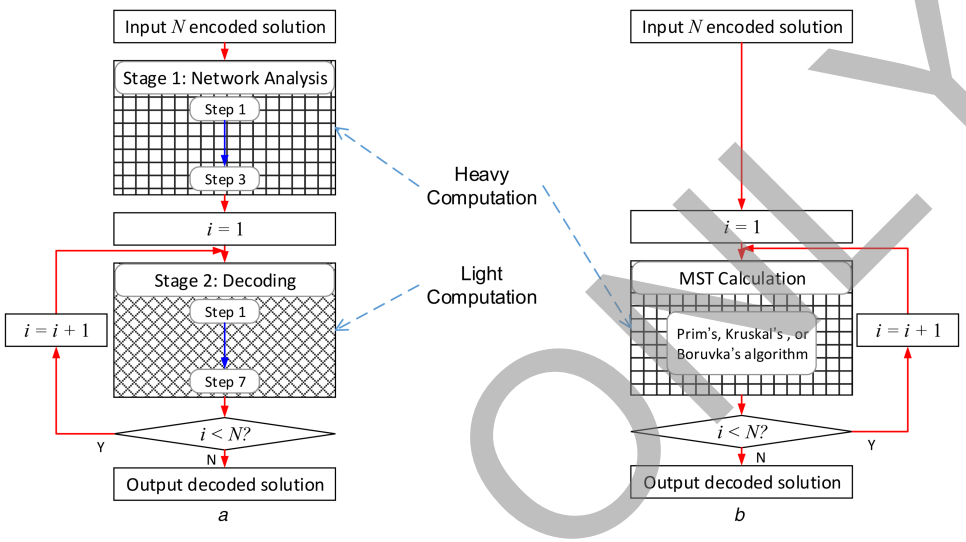


Fig. 4 *Implementation frameworks of different decoding techniques*
(a) PLD method, ***(b)*** MST method

to determine different $\boldsymbol{P}$ for the various decoding process, the easiest method is a random generation. However, this will result in diversity during the convergence process since one $\boldsymbol{R}$ may be decoded into different $\boldsymbol{S}$ if different $\boldsymbol{P}$ is utilised. To guarantee that one $\boldsymbol{R}$ can only be decoded into a unique $\boldsymbol{S}$, the $\boldsymbol{P}$ should be the same for each decoding. One possible strategy is directly deducing $\boldsymbol{P}$ from $\boldsymbol{R}$, which is fulfilled by *Steps 1 and 2*.

Although Stage 1 is straightforward, it is more computationally intensive than Stage 2 due to the shortest path searching. The complexity of Dijkstra algorithm with Fibonacci heap for the shortest path calculation is $O(n_b + n_s + n_d \log n_d)$ [35]. Similarly, the MST calculation also comes with heavy computation. The complexity of Kruskal's algorithm for MST is $O((n_b + n_s)\log n_d)$ [36]. Fig. 4 demonstrates the implementation framework of PLD and MST methods for encoding and decoding. It can be seen that the heavy computation workload is involved for all N times of decoding in the MST method, while only one network analysis is required for the PLD method; therefore, the computational complexity of PLD is lighter. Solid quantitative validation will be given in the case studies.

3 DNPF analysis

As indicated in Section 1, the DA proposed by Teng [32] is advantageous; therefore, its framework is determined by the solution of DNPF. The DA solution process is dominated by BIBC and BCBV matrices, which are generated by BRD in [32]. After a brief introduction of the solution process of DA, this section intends to propose a novel and efficient MRD for the substitution of BRD when formulating BIBC and BCBV.

3.1 Solution process of the DA

Given a DN with n_d nodes, the equivalent current injection for the node i at the kth iteration is given as

$$I_i^k = \left(\frac{P_i + \mathrm{j}Q_i}{V_i^k}\right)^* \quad i \in [1, n_d], \tag{1}$$

where V_i^k is the voltage of bus i at the kth iteration; P_i and Q_i are real and reactive power injections on the node i, respectively; $*$ is the conjugate operator. Consider $[\boldsymbol{V}^k]$ and $[\boldsymbol{I}^k]$ are vectors of V_i^k and I_i^k without reference node. Then the voltage update steps at the kth iteration are given as

$$[\Delta V^k] = [\mathbf{BCBV}][\mathbf{BIBC}][I^k] = [\mathrm{DLF}][I^k]. \tag{2}$$

Therefore, the voltage vector can be updated as

$$[V^{k+1}] = [\boldsymbol{V}^0] + [\Delta V^k], \tag{3}$$

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

On the basis of the above preliminary description and definition, the iterative solution process is summarised as Algorithm 1.

Algorithm 1: Iterative solution process of the DA method

1: Initialise $[\boldsymbol{V}^0]$, set iteration $k = 1$ and $[V^1] = [\boldsymbol{V}^0] + 2\epsilon$.
2: Generate matrices [**BIBC**], [**BCBV**], and [DLF].
3: **while** max $\left\{|V^k - V^{k-1}|\right\} > \epsilon$ **do**

From	To	Z
0	1	0.075 + *j* 0.10
4	3	0.04 + *j* 0.04
0	5	0.11 + *j* 0.11
8	6	0.11 + *j* 0.11
12	10	0.08 + *j* 0.11
10	0	0.11 + *j* 0.11
3	1	0.09 + *j* 0.18
7	5	0.11 + *j* 0.11
13	12	0.04 + *j* 0.04
6	9	0.08 + *j* 0.11
1	2	0.08 + *j* 0.11
6	5	0.08 + *j* 0.11
10	11	0.09 + *j* 0.12

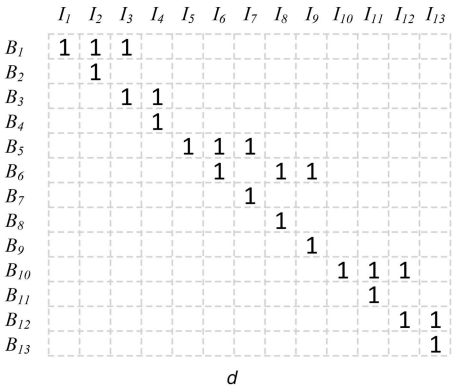
a

From	To
0	1
3	4
0	5
6	8
10	12
0	10
1	3
5	7
12	13
6	9
1	2
5	6
10	11

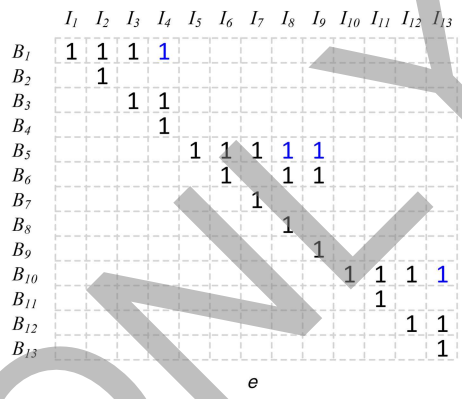
b

ID	From	To	Z
1	0	1	0.075 + *j* 0.10
2	1	2	0.08 + *j* 0.11
3	1	3	0.09 + *j* 0.18
4	3	4	0.04 + *j* 0.04
5	0	5	0.11 + *j* 0.11
6	5	6	0.08 + *j* 0.11
7	5	7	0.11 + *j* 0.11
8	6	8	0.11 + *j* 0.11
9	6	9	0.08 + *j* 0.11
10	0	10	0.11 + *j* 0.11
11	10	11	0.09 + *j* 0.12
12	10	12	0.08 + *j* 0.11
13	12	13	0.04 + *j* 0.04

c



d



e

Fig. 5 *Intermediate results of matrix BIBC generation*
(a) original branch data set, ***(b)*** reorganised branch data set, ***(c)*** ranked branch data set, ***(d)*** generated adjacency matrix, ***(e)*** calculated path matrix

4: Calculate $[I^k]$ according to (1).
5: Compute $[\Delta V^k]$ based on (2).
6: Update $[V^{k+1}]$ just as (3), and set $k = k + 1$.
7: **end while**
8: Calculate the power losses based on $[V^k]$ and $[I^{k-1}]$.

3.2 Matrix generation

It can be seen from Algorithm 1 and (1)–(3) that matrices [**BIBC**] and [**BCBV**] are essential for the iterative procedure. To illustrate how to generate [**BIBC**], the distribution system shown in Fig. 1 is utilised as an example, where the tie switches 4–13, 7–11, and 2–8 are open to formulate a radial tree. Consider B_i as the current for the branch i. The objective is finding matrices [**BIBC**] and [**BCBV**] such that

$$[B] = [\mathbf{BIBC}][I], \tag{4}$$

$$[\Delta V] = [\mathbf{BCBV}][B]. \tag{5}$$

At first, the formulation of [**BIBC**] is illustrated as follows:

- *Step 1: Reorganise the branch data:* Suppose the input data is the one shown in Fig. 5*a*, then a radial tree can be generated as in Fig. 2*a*. This step is exchanging the 'from' and 'to' ends of each branch such that the 'from' has a smaller layer number than the 'to'. The intermediate result is shown in Fig. 5*b*.
- *Step 2: Rank the branch data:* For any DN with n_d nodes, there are $n_d - 1$ branches. After the reorganisation, the 'to' nodes of branches are different from each other, which corresponds to $n_d - 1$ non-reference buses. This step ranks these branches in an ascending order of their 'to' nodes, and then assigns an ID to them. Fig. 5*c* illustrates the temporary result.
- *Step 3: Construct the adjacency matrix:* To describe the direct relationship between [***B***] and [***I***], an adjacency matrix [***A***] is defined. Its size is $(n_d - 1) \times (n_d - 1)$, containing all branches and non-reference buses. Each element is filled with a binary number, where $A_{ij} = 1$ means that I_j can be directly accessed by B_i, and vice versa. The construction process is as follows: (i) since B_i is identical with I_i, the diagonal of [***A***] are all ones; (ii) if there is a branch from non-reference node i to j, set $A_{ij} = 1$. Fig. 5*d* demonstrates [***A***], where ten off-diagonal elements are corresponding to ten branches without reference node.
- *Step 4: Calculate the path matrix:* According to Teng [32], [**BIBC**] is a binary matrix, and $\mathbf{BIBC}_{ij} = 1$ represents that I_j can be accessed by B_i either directly or indirectly. According to the graph theory, this definition is similar to the path matrix, thus the [**BIBC**] is generated as

$$[\mathbf{BIBC}] = f\left([\boldsymbol{A}]^{n_d - 1}\right), \tag{6}$$

where $f(\,)$ is a function that converts any non-zero elements into 1 and keeps zeros constant. The final obtained result [**BIBC**] is shown in Fig. 5*e*.

On the basis of (4), [**BIBC**] in Fig. 5*e* can be interpreted as

$$\begin{cases} B_1 = I_1 + I_2 + I_3 + I_4, \\ B_5 = I_5 + I_6 + I_7 + I_8 + I_9, \\ \quad \vdots \end{cases} \tag{7}$$

which is identical with Fig. 1. It was stated in [32] that [**BIBC**] is an upper triangular matrix. We intend to claim that this is not always true though Fig. 5*e* shows an upper triangular pattern. In Fig. 5*c*, if the 'from' is larger than 'to', matrices [***A***] and [**BIBC**] are both not upper triangular. For example, replace the branch 4–3 with 4–13, the resulting [**BIBC**] is not triangular.

As indicated in [32] that the construction processes of [**BIBC**] and [**BCBV**] are similar, thus the above process can be reused with minor revision. Actually, these two matrices were built in the same subroutine in [32] to save computation resources and time. In this paper, the [**BCBV**] is generated by the following simple equation:

Table 1 Scales of the benchmark systems

Systems	Buses	Feeders	Branches	Tie switches
14-bus	13	3	16	3
33-bus	32	1	37	5
70-bus	68	2	79	11
83-bus	83	11	96	13
136-bus	135	1	156	21

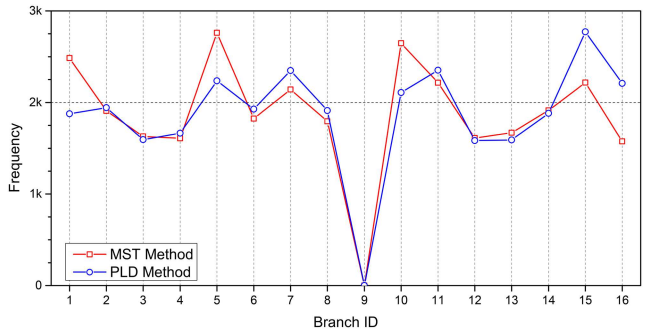


Fig. 6 *Frequency of branches chosen for breaking in the 14-bus system*

$$[\mathbf{BCBV}] = [\mathbf{BIBC}]^{\mathrm{T}}[Z], \tag{8}$$

where T is the transpose operator; $[\boldsymbol{Z}]$ is the matrix whose diagonal is the line impedance shown in Fig. 5*c*. Combining (5) and (8), we get

$$\begin{cases} V_0 - V_4 = Z_1B_1 + Z_3B_3 + Z_4B_4, \\ V_0 - V_9 = Z_5B_5 + Z_6B_6 + Z_9B_9, \\ \vdots \end{cases} \tag{9}$$

which is in accordance with Fig. 1.

3.3 Supplementary explanation

It should be noted that the $[\boldsymbol{A}]$ generated in *Step 3* is not the adjacency matrix according to the definition of graph theory [20] due to the non-zero diagonal. Regarding the tree as a directed graph (each branch is directed from the higher layer to the lower layer) and let $[\boldsymbol{Y}] = [\boldsymbol{A}] - [\boldsymbol{I}]$, then $[\boldsymbol{Y}]$ is the adjacency matrix coordinated with the definition. According to Harris *et al.* [20], the path matrix can be deduced by

$$[\tilde{\boldsymbol{Y}}] = [\boldsymbol{I}] + [\boldsymbol{Y}] + [\boldsymbol{Y}]^2 + \cdots + [\boldsymbol{Y}]^{n-1} + [\boldsymbol{Y}]^n, \tag{10}$$

where n is the size of $[\boldsymbol{Y}]$, i.e. $n = n_d - 1$ in this paper. The (i, j) entry of $[\boldsymbol{Y}]^k$ is equal to the number of walks from node i to j that use exactly k edges.

On the other hand, according to the binomial expansion theorem

$$[\boldsymbol{A}]^n = ([\boldsymbol{I}] + [\boldsymbol{Y}])^n = [\boldsymbol{I}] + C_n^1[\boldsymbol{Y}] + C_n^2[\boldsymbol{Y}]^2 + \cdots + [\boldsymbol{Y}]^n, \tag{11}$$

where C_n^r are constant numbers valued as $\{(n!)/[r!(n-r)!]\}$.

Since there are no cycles in the tree, the maximum number of paths between any two nodes is one; thus, the elements of $[\boldsymbol{Y}]^k$ as well as $[\tilde{\boldsymbol{Y}}]$ are zeros and ones. Therefore, (6) can be rewritten as

$$\begin{aligned} [\mathbf{BIBC}] &= f([\boldsymbol{A}]^n) = f(([\boldsymbol{I}] + [\boldsymbol{Y}])^n) \\ &= f([\boldsymbol{I}] + C_n^1[\boldsymbol{Y}] + C_n^2[\boldsymbol{Y}]^2 + \cdots + [\boldsymbol{Y}]^n) \\ &= f([\boldsymbol{I}]) + f(C_n^1[\boldsymbol{Y}]) + f(C_n^2[\boldsymbol{Y}]^2) + \cdots + f([\boldsymbol{Y}]^n) \\ &= f([\boldsymbol{I}]) + f([\boldsymbol{Y}]) + f([\boldsymbol{Y}]^2) + \cdots + f([\boldsymbol{Y}]^n) \\ &= [\boldsymbol{I}] + [\boldsymbol{Y}] + [\boldsymbol{Y}]^2 + \cdots + [\boldsymbol{Y}]^n = [\tilde{\boldsymbol{Y}}]. \end{aligned} \tag{12}$$

Although (6) is identical with (10), the difference on the computational burden is large. In (10), a lot of matrix power should be calculated such as $[\boldsymbol{Y}]^n$ and $[\boldsymbol{Y}]^{n-1}$, while there is only one matrix power $[\boldsymbol{A}]^n$ executed in (6). The complexity of $f(\,)$ is equivalent with the matrix addition. Thus (6) is much more efficient than (10).

Furthermore, based on the above analysis and (11), we obtain

$$f([\boldsymbol{A}]^{n+k}) = f([\boldsymbol{A}]^n) + [\boldsymbol{Y}]^{n+1} + \cdots + [\boldsymbol{Y}]^{n+k}, \quad k \geq 0. \tag{13}$$

Since the longest walk in the tree with n nodes is $n - 1$

$$[\boldsymbol{Y}]^n = [\boldsymbol{Y}]^{n+1} = \cdots = [\boldsymbol{Y}]^{n+k} = \mathbf{0}, \quad k \geq 0. \tag{14}$$

Thus

$$f([\boldsymbol{A}]^{n+k}) = f([\boldsymbol{A}]^n), \quad k \geq 0. \tag{15}$$

This property can be utilised to further improve the efficiency of (6), i.e. to reduce the number of matrix multiplications from $n - 1$ to $\lceil \log_2 n \rceil$. In this example, $n = n_d - 1 = 13$; therefore, $f([\boldsymbol{A}]^{13}) = f([\boldsymbol{A}]^{16})$, and $[\boldsymbol{A}]^{16}$ can be generated by $\lceil \log_2 n \rceil = 4$ times of matrix multiplications as follows:

$$[\boldsymbol{A}]^2 = [\boldsymbol{A}] \times [\boldsymbol{A}], \quad [\boldsymbol{A}]^4 = [\boldsymbol{A}]^2 \times [\boldsymbol{A}]^2,$$

$$[\boldsymbol{A}]^8 = [\boldsymbol{A}]^4 \times [\boldsymbol{A}]^4, \quad [\boldsymbol{A}]^{16} = [\boldsymbol{A}]^8 \times [\boldsymbol{A}]^8.$$

In [32], the **BIBC** is formulated by $n_d - 1$ steps. Since each step is dependent on the former step, the constitution process must be executed in sequence. On the other hand, there are only $\lceil \log_2(n_d - 1) \rceil$ sparse matrix multiplications in MRD method, and each multiplication can be accelerated with parallel processing. Comparison of the efficiency will be discussed with numerical experiments.

4 Numerical experiments

As demonstrated above, two methods PLD and MRD are proposed in this paper to accelerate the solution process of DNRC. To evaluate their performance, three types of numerical experiments are implemented in this section. In the beginning, the comparison between PLD and its advanced counterpart MST method is carried out for the solution decoding. Then, the MRD method is compared with the BRD method on the DNPF solution. Finally, these methods are integrated into a standard PSO framework for the solution of DNRC. The former two tests are the partial validation of the performances of PLD and MRD, while the last one is a complete evaluation of their potential for the DNRC solution.

Five benchmark systems generated from [37] are introduced as the testbed. Table 1 summarises the scales of these systems. All tests are implemented on a personal computer equipped with Intel Xeon E5-2620 central processing unit and Windows 8.1 operating system. MATLAB 2017a is employed for programming and execution.

4.1 PLD method versus MST method for solution decoding

Within the meta-heuristic algorithm framework, the solution decoding process will be executed a lot of times in each iteration; therefore, its efficiency is of great significance for the whole execution. On the other hand, the representation methodology should not contain any bias, i.e. the randomness is valid. This section is devoted to the performance evaluation of the PLD method in terms of randomness and efficiency, where the MST method reported in [4] is introduced for comparison.

4.1.1 Randomness: Without supplementary information, the global optimal may lie anywhere in the solution space. Therefore, the meta-heuristic algorithm always demands an evenly initialised population to cover the solution space as large as possible. Both the

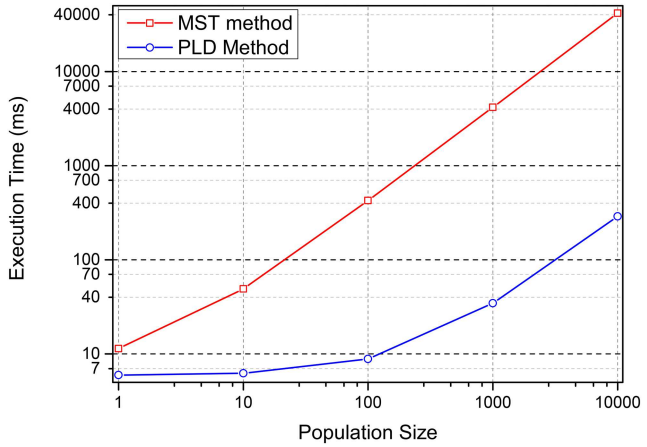


Fig. 7 *Execution time of the PLD and MST methods in double logarithmic coordinate system*

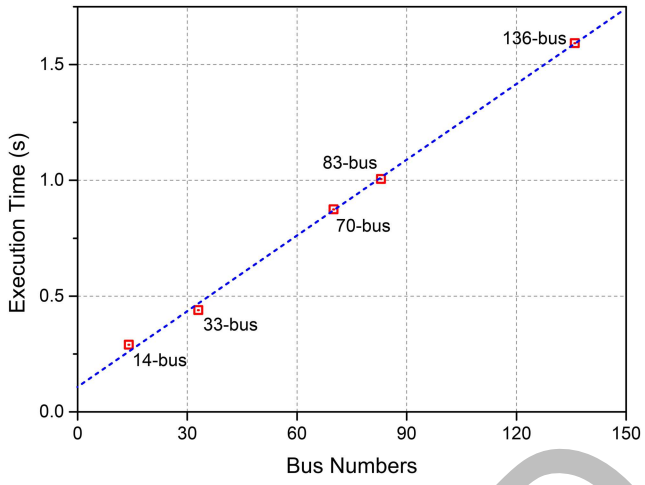


Fig. 8 *Linear relationship between the PLD execution time and bus numbers*

PLD and MST methods are designed for decimal encoded solutions, whose randomness is guaranteed by a lot of software such as MATLAB. Therefore, the randomness of the decoded integer solution is dominated by the transformation methodology.

For the performance validation and comparison, 10,000 real number encoded solutions are randomly generated for the 14-bus system. After decoding, 10,000 integer solutions with sizes of $1 \times n_s$ are obtained, which means there will be 30,000 branches to be chosen for breaking. Since the number of candidate branches for breaking is only 15 (branches 6–9 cannot open as it is not included in any cycles), the frequency of each branch should be 2000. Fig. 6 illustrates the results of both methods, it can be seen that the randomness of the PLD and the MST methods are similar and satisfactory. For the MST method, branches close to the root node have a high probability to be chosen for breaking such as branches 1, 5, and 10. While for the PLD method, branches contained by the shortest cycle are likely to be determined for loop destruction, e.g. branches 7, 11, and 15.

4.1.2 Efficiency: To compare the execution efficiency, both methods are implemented to decode $N = \{1, 10, 100, 1000, 10{,}000\}$ solutions for the 14-bus system. Table 2 summarises the comparative results. Fig. 7 demonstrates the relationship between the execution time and the population size in the double logarithmic coordinate system. For the MST method, decoding each solution involves one whole MST calculation, thus its execution has a linear relationship with N. On the other hand, the PLD method invokes two stages for solution decoding, where Stages 1 and 2 are executed for 1 and N times. In this example, the execution times for a single run of Stages 1 and 2 are 5.95 ms and 25.72 µs, respectively. Owing to the light computational burden of

Table 2 Execution time of the PLD and MST methods to decode N solutions for the 14-bus system

N	Execution time, ms		Speedup
	PLD method	MST method [4]	
1	5.981	11.424	1.91 ×
10	6.244	49.031	7.85 ×
100	8.840	426.557	48.25 ×
1,000	34.549	4176.972	120.90 ×
10,000	289.553	41,735.488	144.14 ×

Table 3 Execution time of the PLD and MST methods to decode $N = 10{,}000$ solutions for different systems

Systems	Execution time, s		Speedup
	PLD method	MST method [4]	
14-bus	0.290	41.735	144.14 ×
33-bus	0.439	41.905	95.46 ×
70-bus	0.874	42.344	48.45 ×
83-bus	1.006	42.704	42.45 ×
136-bus	1.592	43.012	27.02 ×

Stage 2, the total execution time of the PLD method experiences a slow increase. On the basis of different properties of execution time, the speedup keeps increasing as N increases, but the increase rate is reduced as the maximum speedup is approached.

Comparisons between the PLD and MST methods on other systems were also implemented. Table 3 demonstrates the results, where N is fixed as 10,000. The advantage of the PLD over MST method is established by the gained speedup. It should be noted that the execution time of the PLD method goes longer as the system size increases. Fig. 8 indicates that there is a linear relationship between the PLD execution time and bus numbers. While the running time of MST method for different systems seems to be constant. The reason is that the utilised method for MST calculation is the MATLAB built-in Kruskal's algorithm, which is highly optimised such that the execution time is insensitive to system scales when the sizes are moderate.

4.2 MRD method versus BRD method for DNPF solution

In this paper, the DA framework given in Algorithm 1 is utilised for DNPF solution, where [**BIBC**], [**BCBV**], and [**DLF**] in lines 1–2 can be generated by either MRD or BRD method. To compare the efficiency with more details, the solution process of Algorithm 1 is divided into three parts:

- *Part I:* Data preparation and matrix generation of [**BIBC**] including line 1 and part of line 2.
- *Part II:* Matrix generation of [**BCBV**] and [**DLF**], which is described in line 2.
- *Part III:* Iterative DNPF solution, consisting of lines 3–8.

Comparison is implemented to validate the accuracy and efficiency.

4.2.1 Accuracy: In this paper, the BRD method reported in [32] is utilised for accuracy validation of the MRD method. Since both methods are integrated into *Part I* of the DA framework, the final difference on the active power losses is fully dependent on the intermediate results provided by the BRD and MRD methods, i.e. the [**BIBC**] matrix. Since [**BIBC**] consists of zeros and ones, the difference is easy to identify. On the basis of the test results for various systems, the [**BIBC**] generated by the MRD method is exactly the same as the one formulated by the BRD method. Accordingly, the final power losses are the same. Therefore, the accuracy of the MRD method is preserved to be the same with the BRD method.

Table 4 Execution times of DNPF solution based on the MRD and BRD methods

Systems	Execution time, ms								Speedup				Execution time reduction, %
	MRD method				BRD method [32]								
	Part I	Part II	Part III	Sum	Part I	Part II	Part III	Sum	Part I	Part II	Part III	Sum	
14-bus	0.226	0.017	0.051	0.294	0.395	0.017	0.050	0.461	1.75	0.98	0.98	1.57	36.31
33-bus	0.292	0.043	0.079	0.414	0.764	0.042	0.077	0.883	2.61	0.98	0.98	2.13	53.15
70-bus	0.416	0.148	0.127	0.692	1.530	0.150	0.128	1.808	3.68	1.01	1.01	2.61	61.74
83-bus	0.441	0.181	0.160	0.782	1.814	0.179	0.161	2.154	4.12	0.99	1.00	2.76	63.71
136-bus	0.716	0.579	0.326	1.620	3.471	0.574	0.325	4.371	4.85	0.99	1.00	2.70	62.93

Table 5 Configuration of different algorithms

Algorithms	Configurations				
	PSO	PLD	MST	MRD	BRD
Alg1	•	—	•	—	•
Alg2	•	—	•	•	—
Alg3	•	•	—	—	•
Alg4	•	•	—	•	—

Table 6 Average active power losses by 20 trials (kW)

Systems	Alg1	Alg2	Alg3	Alg4
14-bus	84.880	84.880	84.880	84.880
33-bus	139.551	139.551	139.551	139.551
70-bus	203.384	202.929	201.412	201.412
83-bus	469.923	470.182	469.878	469.878
136-bus	285.395	286.412	280.954	280.877

4.2.2 Efficiency: In this section, full DA method is implemented to calculate the DNPF for various systems. Owing to alternative methods utilised in *Part I*, the partial as well as total execution times are different, which are all summarised in Table 4. As shown in the above, *Parts II* and *III* are the same for these two implementations, thus the execution time is similar and the speedup is close to 1.00. For *Part I*, the MRD method gains a speedup from 1.75 to 4.85 for different systems, which means that the method is advantageous. Since the improvement is only valid for *Part I*, the value of speedup on the total execution time is reduced to some extent. Taken the BRD method as a benchmark, the last column of Table 4 shows the execution time reduction gained by the MRD method. It is observable that the amount of reduction is larger than 50% for the majority of systems, and it goes larger as the system scale increases.

4.3 Performance evaluation with full DNRC solution

On the basis of the above results, the superiority of the proposed PLD and MRD methods over their counterparts is established. This section intends to evaluate their performance on the solution of full DNRC problems. For consistency, different methods are integrated into the same PSO framework, which is provided by the MATLAB optimisation toolbox [34]. Therefore, four algorithms are separated, whose configurations are illustrated in Table 5. **Alg1** is the benchmark implementation without any proposals utilised; MRD and PLD are employed in **Alg2** and **Alg3** to replace BRD and MST, respectively; finally, both acceleration strategies are integrated in **Alg4**. Each system is tested by all four algorithms for 20 trials. It should be noted that all the settings of PSO are kept as default from MATLAB except for the population size, which is valued as 256 and 512 for the 14-bus and all the other systems.

4.3.1 Quality: Average active power losses for different systems with various algorithms are summarised in Table 6. The PSO is a non-deterministic algorithm and the global optimality of the obtained solution is not guaranteed. For smaller systems with moderate complexity, the probability to reach the global optimal solution is larger, thus all methods achieve the same result for 14- and 33-bus systems. Owing to larger system size and more decision variables, the searching process is much easier to be trapped in a local optimal solution of the non-convex solution space. Therefore, different methods result in various results for 70-, 83-, and 136-bus systems. As discussed in the above section, the MRD shares the same accuracy with the BRD, thus **Alg1** and **Alg2** (**Alg3** and **Alg4**) should terminate with the same quality of results. This prediction has been validated by the last two columns of Table 6. On the other hand, **Alg3** (**Alg4**) presents less power loss than **Alg1** (**Alg2**) for all tests, indicating that the PLD method outperforms the MST method in the convergent property.

4.3.2 Efficiency: Table 7 summarises the average execution time of different algorithms for various systems, which is also illustrated in Fig. 9. It is obvious that **Alg4** performs better than all the other three algorithms. The difference between **Alg1** (**Alg3**) and **Alg2** (**Alg4**) is marked as DIF_1, which is due to the alternative DNPF solution methods, i.e. MRD and BRD. On the other hand, the difference between **Alg1** (**Alg2**) and **Alg3** (**Alg4**) is marked as DIF_2, which is due to the alternative solution decoding methods, i.e. PLD and MST. It can be seen from Fig. 9 that DIF_2 is larger than DIF_1, which means the improvement on the solution decoding is more significant on the DNRC solution. It is also identical with the results reported in Tables 3 and 4 that the speedup gained by PLD is larger than the MRD. However, DIF_1 and DIF_2 are not as large as the difference demonstrated in Tables 3 and 4, the reason is that the PLD and MRD are just parts of the DNRC solution process.

Fig. 10 depicts the speedup gained by **Alg4** over the other algorithms. Compared to **Alg1**, both MST and BRD are replaced in **Alg4**; thus, the speedup is the largest. According to Table 3, the speedup obtained by PLD is decreasing as the system scale increases. While the circumstance is reverse in Table 4 for MRD. In addition, the speedup value and decreasing rate in Table 3 are larger. Thus, the total speedup of **Alg4** *versus* **Alg1** is decreasing. Finally, it will end up with a compromise between the increasing and decreasing tendencies. Since the MST is updated by PLD, the speedup of **Alg4** *versus* **Alg2** is also significant. However, without the updating for the DNPF solution, the speedup decrease rate is sharper than that of **Alg4** *versus* **Alg1**. Both **Alg3** and **Alg4** utilise PLD for solution decoding, the only difference is BRD versus MRD; thus, the obtained speedup of **Alg4** *versus* **Alg3** in Fig. 10 is

Table 7 Average execution time by 20 trials (s)

Systems	Alg1	Alg2	Alg3	Alg4
14-bus	17.722	16.912	2.230	1.436
33-bus	20.122	18.021	4.955	2.730
70-bus	99.887	82.303	33.627	16.009
83-bus	110.360	89.776	42.422	17.830
136-bus	262.537	190.515	122.709	46.509

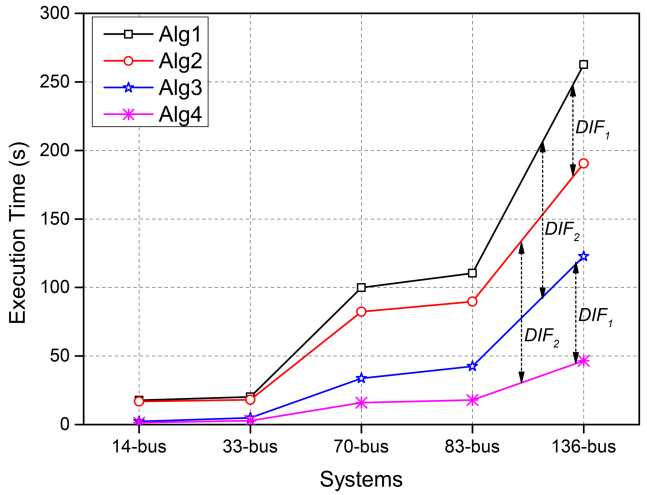


Fig. 9 *Execution time of different algorithms for the DNRC solution*

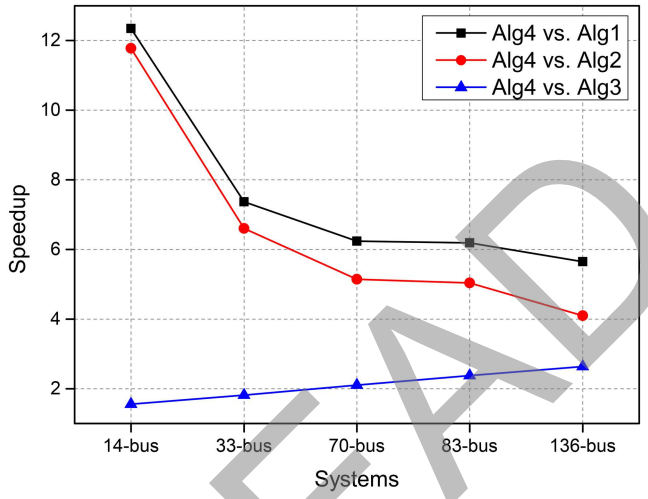


Fig. 10 *Speedup gained by **Alg4** for the DNRC solution*

similar with Table 4. The DNPF is much more time-consuming than the solution decoding in each iteration. Taking the 136-bus system as an example, they are 1.620 and 0.159 ms, respectively. Thus, the total speedups demonstrated in Fig. 10 are approaching the data reported in Table 4.

4.4 Discussion

The efficiency and effectiveness of two proposals PLD and MRD have been validated in the above small- and medium-scale systems, in this section, their potential for large-scale DN will be explored. The largest system with 4400 buses given in [4] is introduced for implementation. Compared to the power loss of 1905.3 kW reported in [4], 7.26% larger power losses (2043.7 kW) are generated for the final DNRC solution obtained with **Alg4**, indicating that the global optimal convergence capability of **Alg4** is not as well as the one given in [4]. The reason is that the PSO utilised in this paper is a MATLAB built-in function, whose parameters are tuned for general applications rather than this specified problem. To gain better performance for large-scale DNRC problems, a lot of preliminary experiments should be implemented for small- and medium-scale cases. It should be noted that the convergence property is determined by the PSO parameters rather than the two proposals; therefore, the potentials of PLD and MRD for practical application should be evaluated from other aspects.

In addition to the DN (single-phase balanced with RTS) investigated in this paper, two main features are commonly appeared in practical DNRC problems, resulting in the following concerns:

- *Three-phase unbalanced DN*: Since the RTS is required to be maintained, the generation of a feasible solution is not trivial. Therefore, the proposed PLD method is still effective and beneficial. Before discussing the robustness of MRD, the capability of DA for three-phase unbalanced DN should be validated. Fortunately, it has been done in [32] and the conclusion is positive. On the basis of this truth, we believe that MRD method is also beneficial for the generation of BIBC for three-phase unbalanced DN since the topology is same and the inner mechanism is similar.
- *Weakly meshed DN*: It should be noted that PLD is proposed to guarantee the feasibility of randomly generated or evolved solutions for DN with RTS topology. If there is no requirement on RTS, i.e. the DN is weakly meshed, the solution feasibility is much easier to be guaranteed. We intend to claim that the proposed PLD is still beneficial for this type of DN with minor revision, i.e. reducing the number of n_s in the decoding process given in Section 2.2.2. The DA method is also capable of weakly meshed DN [32], whose BIBC matrix generation process can be accelerated with MRD approach as well since the theories of adjacency and path matrices are still applicable.

5 Conclusion

Two main concerns are intensively involved in the solution of DNRC: preserving the RTS and calculating the DNPF. In this paper, an effective decimal encoding and decoding techniques are proposed, which guarantees the RTS with the least effort when compared with other state-of-the-art methods. In addition, the solution process of DNPF is enhanced by the introduction of adjacency and path matrices from graph theory. Case studies are conducted on five benchmark systems. In the beginning, solely comparison between the proposal and its counterpart is carried out. Results indicate that the developed methods outperform their advanced counterparts on the solution efficiency without accuracy deterioration. Then, these two improvement methods are integrated into a standard PSO framework to solve DNRC. It is observed that the proposed methods benefit from both efficiency and convergence properties. To fulfil the potential of the proposals for practical application, highly optimised meta-heuristic framework, three-phase unbalanced, and weakly meshed properties, and parallel processing are devoted to future research.

6 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.

7 References

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

[2] Lopez, J.C., Lavorato, M., Franco, J.F., *et al.*: 'Robust optimisation applied to the reconfiguration of distribution systems with reliability constraints', *IET Gener. Transm. Distrib.*, 2016, **10**, (4), pp. 917–927

[3] Arun, M., Aravindhababu, P.: 'A new reconfiguration scheme for voltage stability enhancement of radial distribution systems', *Energy Convers. Manage.*, 2009, **50**, (9), pp. 2148–2151

[4] Roberge, V., Tarbouchi, M., Okou, F.A.: 'Distribution system optimisation on graphics processing unit', *IEEE Trans. Smart Grid*, 2017, **8**, (4), pp. 1689–1699

[5] Lee, C., Liu, C., Mehrotra, S., *et al.*: 'Robust distribution network reconfiguration', *IEEE Trans. Smart Grid*, 2015, **6**, (2), pp. 836–842

[6] Ahmadi, H., Marti, J.R.: 'Linear current flow equations with application to distribution systems reconfiguration,' *IEEE Trans. Power Syst.*, 2015, **30**, (4), pp. 2073–2080

[7] Borghetti, A.: 'A mixed-integer linear programming approach for the computation of the minimum-losses radial configuration of electrical distribution networks,' *IEEE Trans. Power Syst.*, 2012, **27**, (3), pp. 1264–1273

[8] Franco, J.F., Rider, M.J., Lavorato, M., *et al.*: 'A mixed-integer LP model for the reconfiguration of radial electric distribution systems considering distributed generation', *Electr. Power Syst. Res.*, 2013, **97**, pp. 51–60

[9] Ahmadi, H., Marti, J.R.: 'Distribution system optimisation based on a linear power-flow formulation', *IEEE Trans. Power Deliv.*, 2015, **30**, (1), pp. 25–33

[10] Jabr, R.A., Singh, R., Pal, B.C.: 'Minimum loss network reconfiguration using mixed-integer convex programming', *IEEE Trans. Power Syst.*, 2012, **27**, (2), pp. 1106–1115

[11] Taylor, J.A., Hover, F.S.: 'Convex models of distribution system reconfiguration', *IEEE Trans. Power Syst.*, 2012, **27**, (3), pp. 1407–1413

[12] Gupta, N., Swarnkar, A., Niazi, K.R.: 'A modified branch-exchange heuristic algorithm for large-scale distribution networks reconfiguration'. Proc. IEEE Power Energy Society General Meeting, San Diego, CA, USA, July 2012, pp. 1–7

[13] Peng, Q., Low, S.H.: 'Optimal branch exchange for feeder reconfiguration in distribution networks'. Proc. 52nd IEEE Conf. Decision Control, Florence, Italy, December 2013, pp. 2960–2965

[14] Miguez, E., Cidras, J., Diaz-Dorado, E., *et al.*: 'An improved branch-exchange algorithm for large-scale distribution network planning', *IEEE Trans. Power Syst.*, 2002, **17**, (4), pp. 931–936

[15] Shirmohammadi, D., Hong, H.W.: 'Reconfiguration of electric distribution networks for resistive line losses reduction', *IEEE Trans. Power Deliv.*, 1989, **4**, (2), pp. 1492–1498

[16] Fan, J.-Y., Zhang, L., McDonald, J.D.: 'Distribution network reconfiguration: single loop optimisation', *IEEE Trans. Power Syst.*, 1996, **11**, (3), pp. 1643–1647

[17] Ding, F., Loparo, K.A.: 'A simple heuristic method for smart distribution system reconfiguration'. Proc. IEEE Energytech, Cleveland, OH, USA, May 2012, pp. 1–6

[18] Enacheanu, B., Raison, B., Caire, R., *et al.*: 'Radial network reconfiguration using genetic algorithm based on the matroid theory', *IEEE Trans. Power Syst.*, 2008, **23**, (1), pp. 186–195

[19] Chang, G.W., Chu, S.Y., Wang, H.L.: 'An improved backward/forward sweep load flow algorithm for radial distribution systems', *IEEE Trans. Power Syst.*, 2007, **22**, (2), pp. 882–884

[20] Harris, J.M., Hirst, J.L., Mossinghoff, M.J.: '*Combinatorics and graph theory*', (Springer, New York, 2008)

[21] Ju, Y., Wu, W., Zhang, B., *et al.*: 'Loop-analysis-based continuation power flow algorithm for distribution networks', *IET Gener. Transm. Distrib.*, 2014, **8**, (7), pp. 1284–1292

[22] Zimmerman, R.D., Chiang, H.-D.: 'Fast decoupled power flow for unbalanced radial distribution systems', *IEEE Trans. Power Syst.*, 1995, **10**, (4), pp. 2045–2052

[23] Lin, W.-M., Teng, J.-H.: 'Three-phase distribution network fast-decoupled power flow solutions', *Int. J. Electr. Power Energy Syst.*, 2000, **22**, (5), pp. 375–380

[24] Aravindhababu, P., Ashok Kumar, R.: 'A fast decoupled power flow for distribution systems', *Electr. Power Compon. Syst.*, 2008, **36**, (9), pp. 932–940

[25] Irving, M.R., Sterling, M.J.H.: 'Efficient Newton–Raphson algorithm for load-flow calculation in transmission and distribution networks', *IEE Proc., – Gener. Transm. Distrib.*, 1987, **134**, (5), pp. 325–330

[26] Garcia, P.A.N., Pereira, J.L.R., Carneiro, S., *et al.*: 'Three-phase power flow calculations using the current injection method', *IEEE Trans. Power Syst.*, 2000, **15**, (2), pp. 508–514

[27] Yang, H., Wen, F., Wang, L.: 'Newton–Raphson on power flow algorithm and Broyden method in the distribution system'. Proc. IEEE Second Int. Power Energy Conf., Johor Bahru, Malaysia, December 2008, pp. 1613–1618

[28] Shirmohammadi, D., Hong, H.W., Semlyen, A., *et al.*: 'A compensation-based power flow method for weakly meshed distribution and transmission networks', *IEEE Trans. Power Syst.*, 1988, **3**, (2), pp. 753–762

[29] Alinjak, T., Pavic, I., Stojkov, M.: 'Improvement of backward/forward sweep power flow method by using modified breadth-first search strategy', *IET Gener. Transm. Distrib.*, 2017, **11**, (1), pp. 102–109

[30] Saxena, K., Abhyankar, A.R.: 'Agent-based decentralised load flow computation for smart management of distribution system', *IET Gener. Transm. Distrib.*, 2017, **11**, (3), pp. 605–614

[31] Eltantawy, A.B., Salama, M.M.A.: 'A novel zooming algorithm for distribution load flow analysis for smart grid', *IEEE Trans. Smart Grid*, 2014, **5**, (4), pp. 1704–1711

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

[33] Alsaadi, A., Gholami, B.: 'An effective approach for distribution system power flow solution', *Int. J. Electr. Comput. Energetic Electron. Commun. Eng.*, 2009, **3**, (1), pp. 1–5

[34] Mathworks: 'Global optimisation toolbox'. Available at https://www.mathworks.com/help/gads/index.html, accessed 22 July 2017

[35] Fredman, M.L., Tarjan, R.E.: 'Fibonacci heaps and their uses in improved network optimization algorithms'. Proc. 25th Annual Symp. Foundations Computer Science, Singer Island, FL, USA, October 1984, pp. 338–346

[36] Kruskal, J.B.: 'On the shortest spanning subtree of a graph and the traveling salesman problem', *Proc. Am. Math. Soc.*, 1956, **7**, pp. 48–50

[37] Roberge, V.: 'Distribution feeder reconfiguration test cases'. Available at http://roberge.segfaults.net/joomla/index.php/dfr, accessed 13 August 2017