IET Generation, Transmission & Distribution

Research Article

IET Journals
The Institution of Engineering and Technology

Impact of grid partitioning algorithms on combined distributed AC optimal power flow and parallel dynamic power grid simulation

ISSN 1751-8687
Received on 4th August 2020
Revised 15th September 2020
Accepted on 2nd November 2020
E-First on 9th December 2020
doi: 10.1049/iet-gtd.2020.1393
www.ietdl.org

Michael Kyesswa[1] ✉, Alexander Murray[1], Philipp Schmurr[1], Hüseyin Çakmak[1], Uwe Kühnapfel[1], Veit Hagenmeyer[1]

[1]*Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany*
✉ *E-mail: michael.kyesswa@kit.edu*

Abstract: The complexity of most power grid simulation algorithms scales with the network size, which corresponds to the number of buses and branches in the grid. Parallel and distributed computing is one approach that can be used to achieve improved scalability. However, the efficiency of these algorithms requires an optimal grid partitioning strategy. To obtain the requisite power grid partitionings, the authors first apply several graph theory based partitioning algorithms, such as the Karlsruhe fast flow partitioner (KaFFPa), spectral clustering, and METIS. The goal of this study is an examination and evaluation of the impact of grid partitioning on power system problems. To this end, the computational performance of AC optimal power flow (OPF) and dynamic power grid simulation are tested. The partitioned OPF-problem is solved using the augmented Lagrangian based alternating direction inexact Newton method, whose solution is the basis for the initialisation step in the partitioned dynamic simulation problem. The computational performance of the partitioned systems in the implemented parallel and distributed algorithms is tested using various IEEE standard benchmark test networks. KaFFPa not only outperforms other partitioning algorithms for the AC OPF problem, but also for dynamic power grid simulation with respect to computational speed and scalability.

1 Introduction

Today, human life relies on power in almost every facet of life. To ensure reliable supply of power and system stability, simulation studies, such as optimal power flow (OPF) and transient stability analysis can be used to derive system settings and operation limits. OPF is typically cast as a numerical optimisation problem which seeks to minimise the cost of power generation while ensuring continuous grid operation. A survey of OPF is given in [1, 2]. Transient stability analysis studies the behaviour of the grid following network disturbances to derive measures to mitigate the effects of such an occurrence in real operation [3]. However, with ongoing changes in power systems towards renewable energies and microgrids, analysing and maintaining stability of the power grid becomes more challenging [4]. From a control point of view, the challenge lies within managing increasingly many distributed power sources, which requires more data to be transferred to the grid management centre in the context of smart grids. From a power system simulation perspective, an increase in distributed generation results in a considerable increase in computational complexity. Therefore, it is necessary to devise improved measures to cope with the introduced complexity in the analysis methods applied for planning and operation of networks in the smart grids context.

In recent studies, parallel and distributed computing approaches are shown to offer great potential for computational speed up and efficiency in grid simulations [5–8]. The approaches used to parallelise and distribute the simulations mainly rely on spatial decomposition as applied in [8, 9] for OPF problems and in [6, 10] for transient stability analysis. Other approaches applied to transient stability analysis are parallelisation in-time [11] and waveform relaxation methods [12, 13]. However, in order to develop efficient parallel and distributed solutions for grid simulation algorithms, appropriate partitioning of the grid is required. Power grids are often represented as graphs, where the network nodes and branches are represented by the graph vertices and edges, respectively. Most graph partitioning approaches are hierarchical and split a graph into two partitions per recursion level [14, 15]. Such an approach is called bi-partitioning, and recursive bi-partitioning can create a k-partitioning, but it is not a computationally efficient means of doing so [16] [While not computationally efficient, recursive bi-partitioning has been shown to make more efficient use of memory resources [17]]. Some specialised partitioning methods have been developed for use in distributed OPF, such as the spectral method of [15]. Such an approach seeks to create partitions based on information gleaned from the Hessian of the Lagrangrian of the OPF problem, however this requires a priori knowledge of the optimal operating point. Furthermore, both bi-partitioning and spectral clustering are known to scale poorly, and other techniques – such as multilevel graph partitioning – are required for large graphs [18].

In this paper, a variety of partitioning strategies are applied to standard test networks of varying complexity. Two distinct power system simulation algorithms are then tested on these partitioned grids to observe the impact of proper spatial partitioning on computational efficiency. The two algorithms are the augmented Lagrangian based alternating direction inexact Newton (ALADIN) algorithm [19] for AC OPF and a transient stability analysis algorithm [20] for the simulation of power grid dynamics. The transient stability analysis algorithm is reformulated using a block bordered diagonal form (BBDF) [10] of the network equation, which is used for the parallelisation of the analysis process. The primary benefit of the presented concepts is the fact that they allow for distributed computation, which allows for optimisation without the full problem description being stored in any one place. Hence, a distributed approach can reduce the amount of data transferred in a future power grid, minimising expensive communication hardware requirements in smart grids. The main contributions of this paper are an examination of how grid partitioning strategies affect the results of the OPF problem, and an evaluation of the same partitions for the transient stability analysis. This paper provides a proof of concept for the parallelisation of both the OPF computation as well as the dynamic simulation. Furthermore, the results presented in this paper offer some first steps towards optimal power system partitioning strategies. It shall be shown that one such partitioning strategy yields some very promising results

despite the resulting partitions being used for two very different applications.

The rest of this paper is organised as follows: Section 2 describes the formulation of the OPF and dynamic simulation problems. This is followed in Section 3 by a description of the algorithms that are applied to the problems stated in Section 2, as well as the parallel partitioning method that is used to create the required subsystems for the two application cases. In Section 4, results from the two application cases are described regarding the speedup and efficiency. Section 5 concludes the paper, including a discussion regarding the performance and proposition of how this performance can be improved in future implementations.

2 Problem formulation

The current section gives an overview of the general formulation of the two power system simulation studies applied in this paper: the AC OPF problem, and the power system dynamic simulation.

2.1 AC OPF problem

The AC OPF problem is a constrained non-convex optimisation problem that involves some form of Kirchhoff's voltage and current laws. While OPF is sometimes used to refer to any problem with this particular structure, the AC OPF problems considered in the following are those where the objective to be minimised is the generator cost, whose form is given in (1). The box constraints for the decision variables are given in (2), and the AC power flow constraints are shown in (3) and (4).

$$f(x) = \sum_{i \in G} f_i(x) \text{ with } f_i(x) = a_i p_i^2 + b_i p_i + c_i \tag{1}$$

$$\underline{p_i} \leq p_i \leq \bar{p}_i, \underline{q_i} \leq q_i \leq \bar{q}_i, \underline{v_i} \leq v_i \leq \bar{v}_i \tag{2}$$

$$v_i \sum_{k \in N_i} v_k (g_{ik} \cos\theta_{ik} + b_{ik} \sin\theta_{ik}) + p_{d_i} - p_i = 0 \tag{3}$$

$$v_i \sum_{k \in N_i} v_k (g_{ik} \sin\theta_{ik} - b_{ik} \cos\theta_{ik}) + q_{d_i} - q_i = 0 \tag{4}$$

Thereby, $\boldsymbol{x}$ is the vector of the physical state variables of the power system. This vector includes the voltage angle θ_i, the voltage magnitude v_i, active injected power p_i and reactive injected power q_i for every bus i within the power system. The constants g_{ik} and b_{ik} correspond to the conductance and susceptance of the branch between bus i and its neighbour k. All generator buses are combined in the set G and $f_i(x)$ is the cost function for a single generator bus. It contains three cost parameters a_i, b_i and c_i which are defined for every generator. Finally, note that as voltage angle and magnitude are relative, one of the buses needs to be defined as the reference bus, or slack bus. The slack bus has the equality constraints $\theta_r = 0$ and $v_r = const$.

2.2 Dynamic power grid simulation

Transient stability analysis in power systems is used to study system response and its capabilities to regain a steady state after it is exposed to disturbances like network faults, load variations and line parameters. As a general description, the power system is modelled using a set of differential and algebraic equations of the form given in (5), where x are the state variables which include the generator rotor speed, rotor angle, and generator internal voltage; y are the algebraic variables including the injected node currents, node voltages, and injected active and reactive power; and u are the system parameters. The differential equations are used to model the dynamic behaviour of generators and the connected controllers, which include excitation control systems for regulating the generator terminal voltage [21] and turbine-governor systems for controlling the rotational speed and input mechanical power [22]. The algebraic equations are used to represent the generator-stator and transmission network. Details of the component models that comprise the differential and algebraic equations in (5) are given in [20, 23]. The network equation constitutes the largest part of the algebraic equations and is derived from the current injection I_i at each node as given in (6). Combining the current injections at each node results into the nodal current equation of the form given in (7) which is used to describe the network equation, where $\boldsymbol{I}$ is a vector of current injections (I_i), $\boldsymbol{V}$ is a vector of node voltages (V_i) and $\boldsymbol{Y}_{\text{Bus}}$ is the nodal admittance matrix.

$$\dot{x} = f(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}); 0 = g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}) \tag{5}$$

$$I_i = (g_{ii} + \mathrm{j}b_{ii})v_i + \sum_{k \in N_i, k \neq i} v_k (g_{ik} + \mathrm{j}b_{ik})(\cos\theta_k + \mathrm{j}\sin\theta_k) \tag{6}$$

$$\boldsymbol{I} = \boldsymbol{Y}_{\text{Bus}} \boldsymbol{V} \tag{7}$$

It is important to note that the formulation of the dynamic simulation problem in this paper does not include static var compensators (SVCs), dynamic loads, HVDC, and FACTS devices. In addition, the domain of analysis in this paper is limited to electromechanical transient simulations that can be applied to large networks, unlike the computationally complex electromagnetic transient simulations limited to small network sections.

3 Methodology

Power grid simulations can benefit from advances in computational technology if the algorithms are modified to run on parallel and distributed computing hardware. This can be achieved by either devising alternative algorithms which give similar results, but offer more parallelisation potential, or decomposing the algorithms into subsystems that can be parallelised. Either way, such algorithms first need the problem to be reformulated such that its components can easily be distributed to several computing nodes and parallelised. The reformulation of the OPF problem and parallel dynamic simulation are described in the following sections.

3.1 Distributed AC OPF with ALADIN

While many algorithms have been developed for non-linear programmes (NLPs), much less are available which fit a distributed computing context. The ALADIN algorithm is one such method that also guarantees convergence to local optimality for non-convex NLPs details of which can be found in [19]. As the AC-power flow manifold constitutes a highly non-convex constraint in the AC OPF problems (1)–(4), ALADIN is a natural choice of distributed optimisation algorithm. Furthermore, it should also be noted that recent work has shown faster convergence of ALADIN for the AC OPF compared to other methods, like ADMM [24].

The steps of the ALADIN algorithm are summarised as follows: The algorithm is initialised with an initial guess of the primal solution $\boldsymbol{x}_0$, an initial vector of the dual variable of the coupling problem $\boldsymbol{\lambda}_0$. The penalty parameters ρ and μ as well as the weighting matrix $\sum_i$ must also be given. The algorithm works iteratively until the coupling constraint violation and the relative change per iteration is smaller than a given $\in$.

In the first step, the decoupled sub-problems are solved. Note that the local sub-problems do not use their original objective function but rather the so-called augmented Lagrangian; a crucial requirement for ALADIN's convergence properties. Semantically put, the updated objective function allows the algorithm to modify the local sub-problems in a way that it can control the progress towards a feasible solution of the original problem.

After solving the local subproblems in Step 1, a quadratic approximation of the objective function at the current solution, with linearised active sets C^*, is constructed in Step 2. The quadratic approximation with coupling constraints is then solved in Step 3. Note the subproblem solved in Step 3 is an equality constrained quadratic programme and thus is solvable, under certain regularity conditions, as a linear system of equations via the

Input: $y,\ \lambda > 0,\ \rho > 0,\ \mu > 0, \delta > 0$

1. Solve local NLPs

$$x_i^* = \underset{x_i}{\operatorname{argmin}}\ f_i(x_i) + \lambda^\top A_i x_i + \frac{\rho}{2}\left\|(x_i - y_i)\right\|_{\Sigma_i}^2$$
$$\text{s.t.}\quad h_i(x_i) \le 0 \mid k_i$$

2. Compute local gradients & Hessians for QP

$$g_i = \nabla_{x_i} f_i(x_i^*)$$
$$H_i = \nabla_{x_i}^2 \left(f_i(x_i^*) + k_i^\top h_i(x_i^*)\right)$$
$$C_{i,j}^* = \begin{cases} \nabla\left(h_i(x_i^*)\right)_j & \text{if } (h_i(x_i^*))_j = 0 \\ 0 & \text{otherwise} \end{cases}$$

3. Solve centralized QP

$$\Delta x^* = \underset{\Delta x, s}{\operatorname{argmin}}\ \tfrac{1}{2}\Delta x^\top H \Delta x + g^\top \Delta x + \lambda^\top s + \tfrac{\mu}{2}\|s\|_2^2$$
$$\text{s.t.}\quad A(x + \Delta x) - s = b \mid \lambda_{QP}$$
$$C^* \Delta x = 0$$

4. Line search (optional)
   Implement according to Algorithm 3 in Houska et al. (2016) to obtain $\alpha_1, \alpha_2, \alpha_3$; otherwise let $\alpha_1 = \alpha_2 = \alpha_3 = 1$
5. Update
   $y \leftarrow y + \alpha_1(x^* - y) + \alpha_2 \Delta x^*,$
   $\lambda \leftarrow \lambda + \alpha_3(\lambda_{QP} - \lambda)$
6. Terminate check
   If $\left\|Ax^* - b\right\|_1 > \delta$ and $\sum_{i=1}^n \rho \left\|\sum_i (x_i^* - y_i)_1 > \delta\right\|$ then terminate. Else go to Step 1

Fig. 1 *Algorithm 1: Mixed-integer extension of ALADIN [19]*

linear-independence Karush–Kuhn–Tucker conditions [25]. In Step 4, a line search may be performed. Doing so is crucial for guaranteeing convergence and can reduce the number of iterations until convergence, but in practice this step is often ignored or replaced with a heuristic approach. Finally, the primal and dual variables are updated according to the solutions of Steps 1 and 3. The algorithm then proceeds to the next iteration and continues until the termination criteria are met. Algorithm 1 (see Fig. 1) summarises the steps of the ALADIN algorithm.

3.2 Parallel dynamic simulation based on BBDF

Dynamic simulations are used for studying relatively slow responses as electromechanical transients and dynamics due to control devices based on time-domain simulations. This kind of analysis mainly focuses on the transient response of interconnected systems to disturbances in the electrical network. The workflow in the dynamic simulation starts with the calculation of the steady-state operating conditions in terms of bus voltage and power. The state variables of the dynamic models for generator, governor and exciter are initialised from the solution of (5) based on the results of the OPF in Section 3.1 and setting the time derivatives equal to zero at the initialisation point. The initial variables are denoted as y_0 and x_0. The augmented $\boldsymbol{Y}_{\text{Bus}}$ matrix is created by adding the generator internal transient reactance and loads represented by constant admittances to the network admittance matrix. The main algorithm loop solves the differential equations and algebraic equations in (5) at every time step. In this algorithm, the numerical integration is based on the fourth-order Runge-Kutta [26] method while the network equation is solved using the direct LU factorisation method [26]. The matrix is only updated in case of an event. An example of an event is a short-circuit fault that is simulated by inserting a high shunt admittance on the bus where the fault is located. Thereby, the shunt admittances of the network are modified during the fault period in the event handling process.

The solution of the network equation in (7) for the node voltages is the most time consuming task in dynamic simulations. An alternative formulation for the network equation, based on the BBDF, is a faster solution [10], which is an in-space parallelisation approach that can be used for the parallelisation of the network solution step. The basic concept is to rearrange the network admittance matrix such that the equation system given in (8) has a preprocessing step and one step for every sub block of the matrix.

Inputs: Network casefile and partitions

1. Initialization: y_0, x_0
2. Precomputation:
   Form subsystem matrices Y_i, $\bar{Y}_i$ and Y_s
   Compute $\hat{Y}_s$ and product $\bar{Y}_i^T \cdot Y_i^{-1}$
   LU factorize Y_i and $\hat{Y}_s$
3. Compute subsystem state variables
   for *each partition* **do**
      Calculate machine state variables X_i
      Compute current injection in each partition I_i
   end
4. Compute interconnect system variables
   Compute link currents $\hat{I}_s$
   Solve for the interconnect subnetwork voltages V_s
5. Compute subsystem internal variables
   for *each partition* **do**
      Solve for subnetwork node voltages V_i
   end

Return: State and algebraic variables at each time step

Fig. 2 *Algorithm 2: Parallel computation procedure*

The sub blocks, which correspond to the subnetworks, are created from partitioning of the power grid and can be solved in parallel.

Equation (8) shows the restructured $\boldsymbol{Y}_{\text{Bus}}$ matrix in the block bordered diagonal form. The Y_i elements represent the elements of the basic $\boldsymbol{Y}_{\text{Bus}}$ matrix within partition i. Y_s represents the properties of the buses of the interconnect partition as well as all branches that are connecting to interconnect buses. All $\bar{Y}_i$ elements consist of data regarding the branches that connect a normal partition with the interconnect partition. The voltage and current vectors must be reordered accordingly.

$$\begin{bmatrix} Y_1 & & & & \bar{Y}_1 \\ & Y_2 & & & \bar{Y}_2 \\ & & \ddots & & \vdots \\ & & & Y_p & \bar{Y}_p \\ \bar{Y}_1^T & \bar{Y}_2^T & \dots & \bar{Y}_p^T & Y_s \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_p \\ V_s \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_p \\ \boldsymbol{I}_s \end{bmatrix} \tag{8}$$

$$\hat{\boldsymbol{Y}}_s \boldsymbol{V}_s = \hat{\boldsymbol{I}}_s \tag{9}$$

$$\hat{\boldsymbol{Y}}_s = Y_s - \sum_{i=1}^{p} \bar{\boldsymbol{Y}}_i^T Y_i^{-1} \bar{Y}_i \tag{10}$$

$$\hat{\boldsymbol{I}}_s = \boldsymbol{I}_s - \sum_{i=1}^{p} \bar{Y}_i^T Y_i^{-1} I_i \tag{11}$$

$$\boldsymbol{Y}_i V_i = I_i - \bar{Y}_i \boldsymbol{V}_s \tag{12}$$

Equations (9)–(11) describe the preprocessing steps that enable (8) to be solved in blocks of the form seen in (12). Equation (9) shows how the interconnect partition voltage vector can be pre-calculated. This requires the matrix $\hat{\boldsymbol{Y}}_s$ and the vector $\hat{\boldsymbol{I}}_s$, which can be calculated with (10) and (11). To speed up the simulation, (10) can be completely calculated at the beginning of the simulation. Equation (11) needs to be calculated every iteration of the simulation since it depends on the current vector $\boldsymbol{I}_s$, which changes throughout the simulation. However, the $\bar{Y}_i^T \cdot Y_i^{-1}$ part can be pre-computed as well. In addition, the two matrices $\boldsymbol{Y}_i$ and $\hat{\boldsymbol{Y}}_s$ can be factorised by LU decomposition beforehand to speed up the calculation during the simulation loop. The speedup for this approach depends on the runtime growth from larger $\boldsymbol{Y}_{\text{Bus}}$ matrices for solving the original equation system with the LU factorised $\boldsymbol{Y}_{\text{Bus}}$ matrix. The parallel solution process is summarised in Algorithm 2 (see Fig. 2). Further details of the algorithm and the

communication aspects during the parallel simulation process are described in [27].

3.3 Power grid partitioning

In both algorithms of Sections 3.1 and 3.2, the main criterion for the speed is the number of branches that connect partitions, where fewer connections typically result in better performance. A secondary criterion is the equality of partition sizes, as the result gathering takes as long as the slowest parallel processor. These two criteria perfectly fit the common partitioning problem in graph theory described by Buluç *et al.* [18]. Since graph partitioning takes different forms throughout the literature, the following definition is adopted in this paper:

Definition 1: For a weighted graph $G = (V, E, w)$, where V is the vertex set, E is the edge set, and $w: E \to \mathbb{R}$ represents the edge weights, a *k-partitioning* of G is formally defined as the set

$$\{G_1(V_1, E_1, w_1), \ldots, G_k(V_k, E_k, w_k)\}$$

such that

- $V_1 \cup \cdots \cup V_k = V$
- $V_i \neq V_j, \forall i \neq j$
- $|V_i| \leq (1 + \varepsilon)\lceil |V|/k \rceil$, for some parameter $\varepsilon > 0$.

In the definition above, if $\mathbb{E}$ is the set of edges in $\sum[E - \{E_1 \cup \cdots \cup E_k\}]$, the partitioning is considered to be optimal if $\sum_{e \in \mathbb{E}} w(e)$ is minimised. The third condition defines the balance constraint for balancing the number of vertices in each partition.

In the same way, the problem of graph clustering is defined as follows:

Definition 2: For a weighted graph $G = (V, E, w)$, a clustering of G given $k > 1$ is defined as the set

$$\{G_1(V_1, E_1, w_1), \ldots, G_k(V_k, E_k, w_k)\}$$

such that

- $V_1 \cup \cdots \cup V_k = V$
- $V_i \neq V_j, \forall i \neq j$

In the above definition, the clustering is considered to be optimal if $\sum_{i=1}^{k} \sum_{e \in E_i} w_i(e)$ is maximised.

It is important to note that the edge weights w in graph clustering correspond to the affinity between pairs of graph vertices. The pairwise affinity between vertices in G is then defined within an affinity matrix. Thereby, if an affinity matrix is defined, graph clustering can be applied to the partitioning problem.

From the above definitions, it is worth noting that graph partitioning aims to minimise the number of edges between partitions, whereas graph clustering seeks to maximise the affinity of elements within each cluster. With this in mind, the following partitioning algorithms are compared for the AC OPF in Section 3.1 and dynamic simulation problems in Section 3.2:

3.3.1 Karlsruhe Fast Flow Partitioner: The KaFFPa from the 'Karlsruhe High Quality Partitioning' (KaHIP) project is a multi-level graph partitioning algorithm that generates equally sized partitions that have a minimal number of cut branches. KaFFPa consists of three main steps: contraction, initial partitioning, and refinement [28, 29]. In the first step, the algorithm contracts the input graph to create a smaller representation until it is small enough to be partitioned with a global algorithm. To find edges to contract, the algorithm creates a maximum matching using the global paths algorithm (GPA) which was presented in [30]. A matching is a set of edges where no two edges in the set have a common endpoint (vertex). Different weight functions influencing the construction of these matchings are evaluated by Holtgrewe *et al.* [31]. A matching is then contracted by combining the start and endpoint of every edge in the set. This contraction process quickly decreases the size of the input graph. As soon as the graph is small enough, a global partitioning algorithm is applied. Once the global partitioning algorithm has finished, the contraction is then undone step by step, applying local refinement strategies. Essentially, this involves checking whether moving some vertices that lie at the partition boundary to the neighbouring partition would improve the partition balance or the minimum number of edges cut. A detailed description of the KaFFPa algorithm is given in [29, 32]. The result is a partitioning with evenly sized partitions and a minimal number of edges that are cut in between the partitions.

Input: Graph $G(V, E, w)$, number of clusters k
1. Derive an affinity matrix A from graph G
2. Define diagonal matrix D where $D_{i,i} = \sum_{n=1}^{B} A_{i,n}$ and construct the matrix $P = D^{1/2} A D^{1/2}$
3. Find the k largest eigenvalues of P and form the matrix V by stacking the corresponding eigenvectors as columns. Then renormalize each row of V
4. Treat each row in V as a data point and cluster these data points into k clusters, e.g. using the k-means algorithm
5. Assign bus i to cluster G_i if row i of V was assigned to cluster G_i

Return: $G_1(V_1, E_1, w_1), \ldots, G_k(V_k, E_k, w_k)$

Fig. 3 *Algorithm 3: Spectral clustering algorithm*

3.3.2 METIS: METIS is another example of a multi-level graph partitioning algorithm that is used for partitioning large unstructured graphs. The approach uses undirected graphs as inputs, which are then split into the required number of subsystems [33]. Like KaFFPa, it is a multi-level partitioning method where the main requirement is an equal number of nodes in each subsystem and a minimum number of interconnections between the partitions. However, METIS uses a Kernigan-Lin approach in the uncoarsening phase as opposed to the local search method used by KaFFPa. This algorithmic difference can lead to more edges between partitions despite the overall runtime still being comparable to KaFFPa.

3.3.3 Spectral clustering: Spectral clustreing is another method for partitioning graphs designed for the related problem of graph clustreing. The partitions are formed based on the consideration of the affinity between elements in the system. In other words, if many elements are more similar to each other than the rest of the data set, they are grouped together. In the power system applications considered in this paper, this corresponds to the connectedness between the system nodes. The partitioning is therefore performed based on the definition of an affinity matrix $\boldsymbol{A}$ and groups the nodes according to their corresponding levels of connectedness using a clustering algorithm. The spectral clustering algorithm can be summarised according to Algorithm 3 (see Fig. 3) as described in [15].

Two spectral clustering algorithms are applied in this paper to produce additional partitions. In the rest of the paper, the approaches are referred to as 'Spectral-Hessian' and 'Spectral-Ybus'. The difference between the two algorithms is how the affinity matrix is defined. The Spectral-Hessian approach combines the Hessian matrix of the optimisation problem and the admittance matrix to form the affinity matrix as proposed in [15]. Thereby, the (i, j)th entry of the affinity matrix is given by

$$A_{i,j} = (1 - \tilde{w}) \cdot \sum_{k=1}^{m} \sum_{l=1}^{n} \left| H_{k,l} \right| + \tilde{w} \cdot Y_{i,j} \tag{13}$$

where $Y_{i,j}$ is the (i, j)th element in the admittance matrix and $0 \leq \tilde{w} \leq 1$ is its weight in the affinity matrix. The variables $k = 1, \ldots, m$ and $l = 1, \ldots, n$ denote the indices of entries $H_{k,l}$ in the

Table 1 Size of test networks

Case	Number of components				Load data	
	Gens	Nodes	Branches	Loads	*P*, MW	*Q*, MVar
case9	3	9	9	3	315.0	115.0
case14	5	14	20	11	259.0	73.5
case30	6	30	41	20	189.2	107.2
case39	10	39	46	21	6254.2	1387.1
case57	7	57	80	42	1250.8	336.4
case118	54	118	186	99	4242.0	1438.0
case300	69	300	411	201	23525.8	7788.0
case1354	260	1354	1991	673	73059.7	13401.4
case9241	1445	9241	16049	4895	312354.1	73581.6
case13659	4092	13659	20467	5544	381431.9	98523.4

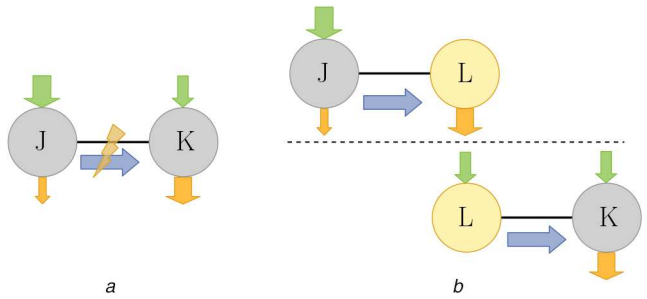

Fig. 4 *Construction of an auxiliary bus for a cut-branch*
(a) Branch cutting, ***(b)*** Formation of an auxiliary bus in each subsystem

Table 2 ALADIN iterations for partitioned IEEE test cases

Case – partitions	KAFFPa	Spectral-Hessian	Spectral-Ybus	METIS
case 9 – 2	3	3	3	3
case 14 – 2	17	29	29	17
case 30 – 2	20	15	15	23
case 39 – 2	4	8	8	4
case 57 – 2	16	14	20	41
case 57 – 3	22	24	29	71
case 118 – 2	20	23	23	22
case 118 – 3	28	34	40	37
case 118 – 4	28	29	25	22
case 118 – 5	23	42	44	44
case 300 – 3	51	28	> 500	102
case 300 – 5	97	> 500	54	103

Hessian matrix $\boldsymbol{H}^*_{\text{sys}}$ associated with the two buses i and j. A detailed description of the approach is given in [15].

On the other hand, the affinity matrix in the Spectral-Ybus approach is formed using the admittance matrix according to the algorithm described in [14]. The first term on the right hand side in (13) is eliminated in the formulation of the affinity matrix. Further details of the algorithm are given in [14].

4 Application case studies

This section presents two application cases to evaluate the partitioning modes in power grid analysis problems. The first application is the solution of the OPF problem given in Section 2.1 and is implemented in the Matlab environment. In the second application case, the partitioning modes are evaluated in a dynamic simulation problem context given in Section 2.2 and implemented in the Julia programming environment [34]. In both application cases, the partitioning strategies are applied to varying network sizes to analyse the variation of simulation runtime with partition sizes.

4.1 Simulation setup

The input network case files used in this work are standard IEEE test networks [35] and larger networks representing the structure of the European grid from the Pegase project [36]. These network case files are part of the MATPOWER package [37]. Table 1 gives a summarised description of the considered networks in terms of number of generators, buses, branches and load data. In this paper, the networks are considered to be unweighted. Thereby, all branches are defined with an equal weight function of one.

4.2 Application case 1: distributed AC OPF

Recall from Section 3.1 that the ALADIN algorithm requires a problem to be given in a separable, but linearly coupled form. Thus, to apply this algorithm to the OPF problem, the original problem definition needs to be partitioned into subproblems. Therefore, the partitionings described in Section 3.3 are used and in between every cut-branch a so-called auxiliary bus is created. Fig. 4 demonstrates this process.

The cut-branch between buses J and K gets split and an auxiliary bus L is inserted in both partitions (separated by the dashed line). The coloured arrows represent power flowing in the given direction and their width represents the amount of power. Loads are coloured orange, injection is coloured green and power transferred on a branch is coloured blue. The appropriate coupling constraints for all four of the physical state variables mentioned in Section 3.1 are created. This can be seen in Fig. 4 with the green and orange arrows, which indicate power inflow and outflow, respectively. These constraints are then included in the coupling QP step of the ALADIN algorithm. Afterwards, the objective functions as well as the constraints (including the power flow equations) are created for every partition separately (they are uncoupled). Each partition uses the modified objective function as described in the previous subsection. The algorithm then progresses as described in the previous section until the termination criteria are met. Further details on this process are provided in [38].

In both versions, a smaller number of interconnecting branches results in a smaller coupling subproblem for ALADIN. This also means less auxiliary buses and therefore less runtime for the local NLPs per iteration. In the parallel implementation, this effect is even stronger, because the QP step is computed sequentially and according to Amdahl's law, a smaller sequential runtime increases the upper bound of the speedup. Additionally, the parallel version benefits from equally sized partitions since this reduces waiting times. Some recent research has shown that the coupling QP of ALADIN can be solved efficiently in a decentralised manner [39], however, such results are preliminary and such an implementation of ALADIN has been left to future work. Interestingly, the relative sizes of the partitions did not play a large role. The effect of equally sized partitions can only be seen using parallelisation.

Table 2 shows the number of iterations required by ALADIN to converge to a locally optimal solution when applied to each of the partitioned OPF problems. Each of the local subproblems are solved with IPOPT [40] and the equality constrained QPs are solved as a linear system via the KKT conditions. In all cases, the termination threshold is 10^{-3}. An iteration limit of 500 is set for Algorithm 1 (see Fig. 1), which is reached in the 300-bus case for several partitionings.

ALADIN requires certain parameters to be given, which not only can be difficult to choose a priori, but also have a large effect on the convergence rate. A number of different combinations of parameters are tested for each of the partitionings and only the best results are shown in Table 2. Interestingly, the best parameters seem to vary from partitioning to partitioning. Table 3 shows the parameters used to generate the results of Table 2. All parameters are given in the form $(\rho, \mu, \dot{\rho}, \dot{\mu})$, where ρ and μ are the initial values of the parameters described in Section 3.1, and, where $\dot{\rho}$ and $\dot{\mu}$ are the factors by which ρ and μ are updated at every iteration. It should be noted that this updating method is a heuristic means of replacing the line search step as discussed in Section 3.1.

While there is some variability, the results shown in Table 2 are relatively good if one considers a worst-case partitioning strategy. For a worst-case partitioning, we use the singleton partitioning,

Table 3 Best ALADIN parameters for each partitioning

Case – partitions	KAFFPa	Spectral-Hessian	Spectral-Ybus	METIS
case 9 – 2	500, 500, 1.05, 2	500, 500, 1.05, 2	500, 500, 1.05, 2	500, 500, 1.05, 2
case 14 – 2	500, 500, 1.05, 2	500, 500, 1.05, 2	500, 500, 1.05, 2	500, 500, 1.05, 2
case 30 – 2	500, 1000, 1.05, 2	500, 500, 1.05, 2	500, 500, 1.05, 2	500, 1000, 1.05, 2
case 39 – 2	20, 2000, 1.15, 1.15	500, 2000, 1.15, 2	500, 2000, 1.15, 2	500, 2000, 1.15, 2
case 57 – 2	1000, 2000, 1.05, 2	1000, 2000, 1.05, 2	1000, 2000, 1.05, 2	100, 100, 1.2, 2
case 57 – 3	100, 2000, 1.2, 2	100, 100, 1.2, 2	100, 2000, 1.2, 2	500, 2500, 1.05, 1.15
case 118 – 2	100, 100, 1.2, 2	100, 100, 1.2, 2	100, 100, 1.2, 2	100, 100, 1.2, 2
case 118 – 3	100, 1000, 1.05, 1.5	100, 1000, 1.05, 1.5	100, 1000, 1.05, 1.5	100, 1000, 1.05, 1.5
case 118 – 4	500, 1000, 1.05, 1.5	500, 1000, 1.2, 1.5	500, 1000, 1.1, 1.5	500, 1000, 1.1, 2
case 118 – 5	500, 2000, 1.1, 2	500, 2000, 1.1, 1.2	500, 2000, 1.1, 1.2	500, 2000, 1.1, 1.2
case 300 – 3	100, 100, 1.1, 2	100, 100, 1.1, 2	N/A	100, 100, 1.05, 1.1
case 300 – 5	100, 100, 1.1, 2	N/A	100, 100, 1.1, 2	500, 500, 1.05, 2

Table 4 Results for the singleton partitioning

Case	Iterations	ALADIN parameters
case 9	3	(500, 500, 1.05, 2)
case 14	42	(500, 500, 1.05, 2)
case 30	162	(50, 250, 1.05, 1.05)
case 39	4	(20, 2000, 1.15, 1.15)
case 57	> 500	N/A
case 118	> 500	N/A
case 300	> 500	N/A

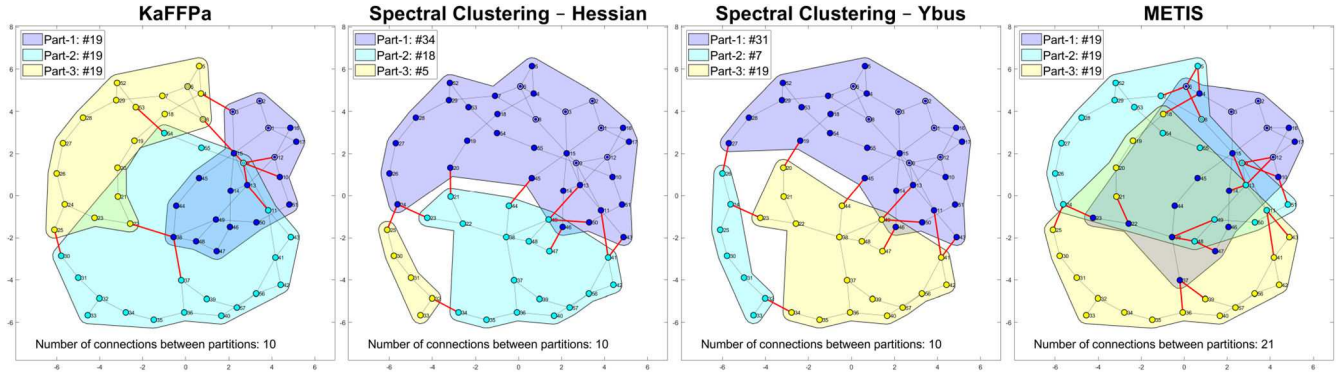

Fig. 5 *Partitions generated for the IEEE 57 bus case. Red lines indicate branches between partitions*

where every bus is in its own partition. While it may seem like there is a lot of potential for parallelisation with such an approach, it actually leads to a significantly larger problem due to the introduction of so many auxiliary buses and a very large QP to solve in Step 3 of ALADIN. The results of this partitioning are shown in Table 4 to converge within 500 iterations.

The IEEE 57 bus case is the smallest example examined such that all of the partitioning algorithms provided different partitionings. Fig. 5 shows the partitionings of each algorithm. Observe that although the partitions generated by KaFFPa and METIS are quite similar, there is nonetheless a large difference in the convergence rate of ALADIN. As shown in Fig. 5, METIS generates many more branches between partitions than KaFFPa, which is likely the reason for its relatively poor performance in the results of Table 2. Also of note is the fact that spectral clustreing applied to the $\boldsymbol{Y}_{\text{Bus}}$ matrix consistently yields almost the exact same partitions as spectral clustreing of the Hessian of the Lagrangian at the optimal operating point. This seems to imply that the resistances and susceptances of the lines are dominant when considering a partitioning and that very little is gained by the a priori knowledge of the optimal operating point. However, as noted in [15], this may change when line limits are reached.

4.3 Application case 2: parallel dynamic simulation

The solution in the parallel dynamic simulation as described in Section 3.2 requires restructuring of the admittance matrix in the form such that no bus in a partition has a direct connection to a bus in another partition. For this, the basic partitions formed by the different partitioning modes are extended to create an interconnect partition under the condition that all branches leaving a partition must connect to a node in the interconnect partition. This is in contrast to ALADIN, where auxiliary buses with corresponding coupling constraints between their decision variables are added on the boundaries of each partitioned problem. Fig. 6 shows the required interconnection between subnetworks through an interconnect partition for an example system divided into three partitions.

The data exchanged between partitions during the parallel computation can be summarised as follows: According to the formulation in Section 3.2, the individual partitions compute their internal node current injections $\boldsymbol{I}_i$ and transfer them to the interconnect partition. The interconnect partition sequentially computes boundary node voltages $\boldsymbol{V}_s$, which are transferred to the separate partitions to complete the parallel solution of the internal node voltages in each partition. Therefore, exchanged variables are node voltages and current injections in terms of magnitude and angle.

The evaluation of the parallel dynamic simulation is performed using the standard test networks shown in Table 1. Each network is partitioned using KaFFPa, METIS, Spectral-Hessian, and Spectral-Ybus partitioning modes and adapted to the interconnect format. In each test case, the dynamic simulation is run for 10 s, with a step size of 1 ms. The simulations are run in the Julia environment on a single thin node of the ForHLR II computing cluster [https://www.scc.kit.edu/dienste/forhlr2.php] with two Intel Xeon E5-2660 v3 Deca-Core processors and 64 GByte of RAM per node.

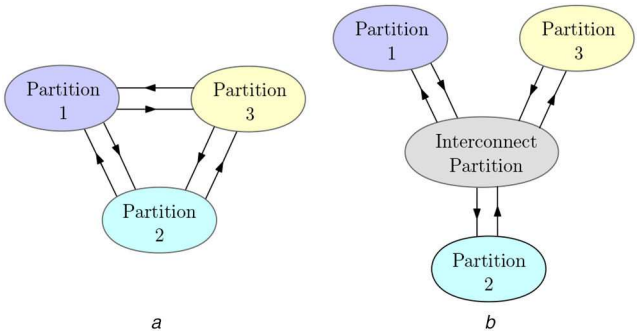

Fig. 6 *Formation of the interconnect partitioning format for parallel dynamic simulation*
(a) Initial partitioning output, ***(b)*** Connection between partitions via an interconnect partition

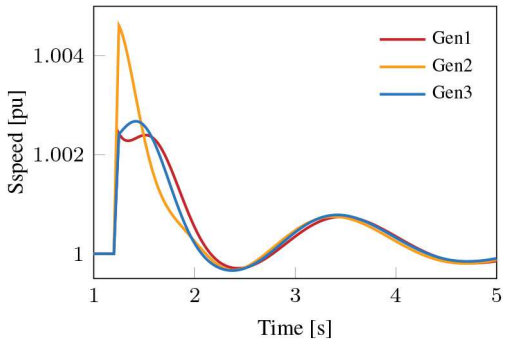

Fig. 7 *Response of generator rotational speed following a topology change due to a bus fault in the partitioned network*

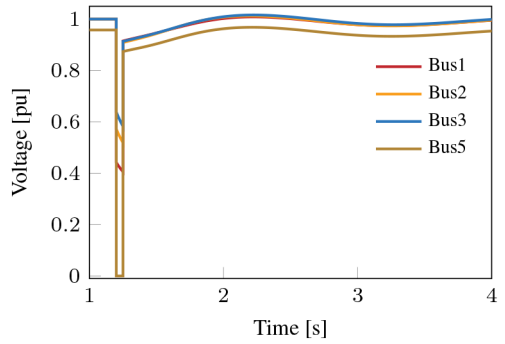

Fig. 8 *Response of bus voltage following a topology change due to a fault on bus 5 in the partitioned network*

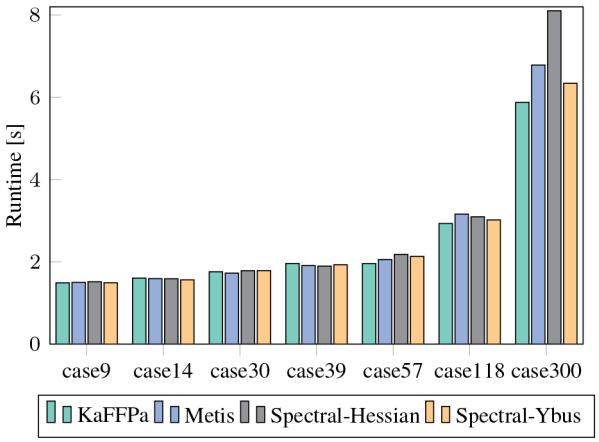

Fig. 9 *Comparison of runtime for dynamic simulations with networks partitioned using different partitioning strategies*

Initialisation of the system to the steady-state operating conditions is based on the results of the AC OPF solution as presented in Section 4.2. In each test case, an event – in form of a short-circuit fault – is triggered during the simulation. The fault is applied by inserting a high shunt value on a bus for a duration of 50 ms, and cleared by resetting the bus shunt to the original value. This results in two events in the process, which implies that the network topology changes twice during the simulation. As an illustration of the transient behaviour following a change in network topology, Figs. 7 and 8 show the response of the generator rotational speed and bus voltage magnitude, respectively, for the nine-bus system (case 9) partitioned into two subsystems. It is important to note that the results obtained using the partitioned system are similar to those with the original network as described in [27]. However, the variables in this case are positive sequence variables, since the analysis is based on a symmetrical transients' analysis algorithm. The algorithm will be extended to include analysis of unsymmetrical transients as part of future implementations according to the analysis approach presented in [41].

Fig. 9 shows the simulation runtime of the network structures partitioned using the different partitioning modes. Each test case is shown with a selected partition count as summarised in Table 5. For example, case 9 – 2 is the IEEE case 9 network with two partitions. From the results shown in Fig. 9, the simulation runtimes of the test cases partitioned using the KaFFPa partitioning mode are observed to be better than the runtimes with the other partitioning modes.

To explain the performance difference of the simulations using the different partitioning modes, Table 5 shows a summary of the partitioning information for the considered test cases. In Table 5, the first n digits are the sizes of the n main partitions, while the last digit is the size of the interconnect partition. For example, case 9 – 2 with the KaFFPa partitioning mode results in partitioning $4, 3 \sim 2$, which is interpreted as four buses in the first partition, three buses in the second partition, and interconnected by two buses. The good performance of the KaFFPa partitioning mode can be attributed to the optimised network partitioning resulting into a relatively balanced number of buses in each partition and a small size of the interconnect partition.

In a further step, the KaFFPa partitioning mode is tested with larger test cases. For this, standard test networks with the size and complexity of the European network – case 1354, 9241, 13,659 – as described in [36] are used. Fig. 10 shows the computational speedup for the large test cases for different number of partitions using KaFFPa.

The reference speedup of 1.0 is the point at which the computational runtime of the parallel algorithm is equal to that of the sequential algorithm. Important to observe is that there exists an optimal partitioning for each network according to the attained speedup. At the optimal partitioning, there is a balance between the main partition sizes – which creates balanced parallelisable tasks – and minimises the size of the interconnect partition in order to reduce the sequential simulation runtime. Therefore, the optimal partitioning of the network for dynamic simulations depends on the size of the interconnect partition and the relative size of the main partitions to the interconnect partition. In addition, it can be observed that the computational speedup increases with the network size. This implies that optimal network partitioning is potentially beneficial for dynamic simulations of large networks.

5 Discussion

In Section 4, an AC OPF problem and a dynamic power grid simulation are partitioned and their respective parallelised

Table 5 Partitioning information for partitioned IEEE test cases

Case – partitions	KAFFPa	Spectral-Hessian	Spectral-Ybus	METIS
case 9 – 2	4, 3 ~ 2	3, 3 ~ 3	3, 4 ~ 2	3, 3 ~ 3
case 14 – 2	6, 6 ~ 2	6, 6 ~ 2	8, 4 ~ 2	8, 4 ~ 2
case 30 – 2	14, 13 ~ 3	14, 13 ~ 3	16, 11 ~ 3	16, 11 ~ 3
case 39 – 2	18, 18 ~ 3	18, 18 ~ 3	21, 15 ~ 3	4, 3 ~ 5
case 57 – 3	18, 17, 16 ~ 6	18, 14, 14 ~ 11	9, 27, 10 ~ 11	27, 7, 15 ~ 8
case 118 – 4	28, 26, 30, 24 ~ 10	27, 26, 27, 25 ~ 13	36, 28, 28, 16 ~ 10	32, 16, 27, 33 ~ 10
case 300 – 5	56, 57, 57, 57, 58 ~ 15	58, 57, 56, 55, 56 ~ 18	88, 58, 18, 13, 100 ~ 23	63, 60, 83, 63, 18 ~ 13

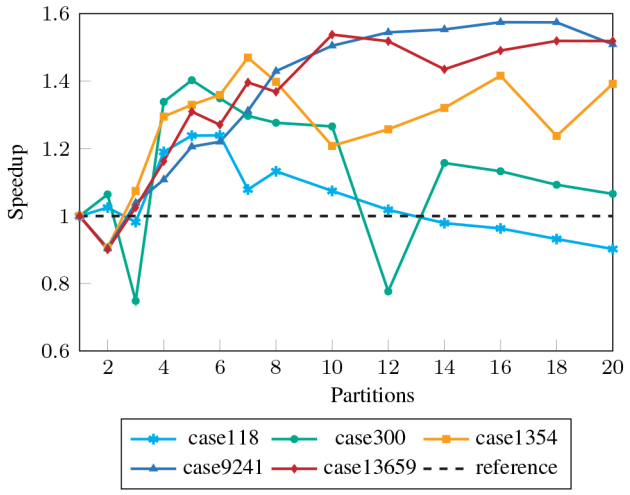

Fig. 10 *Computational speedup of dynamic simulations for large network cases partitioned using KaFFPa strategy*

algorithms are applied. The computational speed up of the implemented parallel and distributed algorithms are tested using various IEEE standard benchmark test networks. Overall, each partitioning algorithm showed fairly similar performance. However, for the larger grids KaFFPa is better in terms of number of iterations in the OPF problem and computation runtime in dynamic simulations.

Both ALADIN and the dynamic simulation are currently limited to execution on a single shared memory computer. This limits further scaling for larger input cases and more partitions. The results from both the dynamic simulation and ALADIN show that a proper partitioning improves the runtime and that there is potential for finding a partitioning that is tailored better for a given algorithm. This implies more focus on minimisation of the interconnect partition for the dynamic simulation and on having the amount of generation and loads balanced in each partition for the OPF algorithm.

Future implementations will consider other partitioning strategies, such as a means of performing dynamic problem partitioning within each ALADIN iteration. As the spectral clustering method requires almost the exact information that is computed in Step 2 of ALADIN, it may be possible to develop a variant of ALADIN that maintains its convergence and optimality guarantees but has the ability to dynamically adjust a given partitioning in order to improve performance. Such an algorithm would have the advantage of not having to rely on an optimally partitioned system to be given as input. The dynamically generated partitions will also be applied to the dynamic simulation problem to further compare their applicability to the two power system problems. Furthermore, new developments in ALADIN could be implemented to improve the efficiency of the coupling QP step.

6 Acknowledgments

This work was part of the Energy Systems 2050 project, an initiative of the Helmholtz Association. The work was conducted within the project E-GriCo using the supercomputer ForHLR funded by the Ministry of Science, Research and Arts Baden-Württemberg and by the Federal Ministry of Education and Research.

7 References

[1] Frank, S., Steponavice, I., Rebennack, S.: 'Optimal power flow: a bibliographic survey, I: formulations and deterministic methods', *Energy Syst.*, 2012, **3**, pp. 221–258

[2] Frank, S., Steponavice, I., Rebennack, S.: 'Optimal power flow: a bibliographic survey, II: nondeterministic and hybrid methods', *Energy Syst.*, 2013, **3**, pp. 259–289

[3] Koester, D., Ranka, S., Fox, G.: 'Power systems transient stability-a grand computing challenge', Northeast Parallel Architectures Center, Syracuse, NY, Tech. Rep. SCCS, vol. 549, 1992

[4] D'Arco, S., Suul, J.A., Fosso, O.B.: 'A virtual synchronous machine implementation for distributed control of power converters in smartgrids', *Electr. Power Syst. Res.*, 2015, **122**, pp. 180–197

[5] Pellegrini, F.: 'Distillating knowledge about SCOTCH', in Uwe, Naumann, Horst, D. Simon (Eds.): '*Combinatorial scientific computing* (Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009), pp. 1–12

[6] Tomim, M., Martí, J., Wang, L.: 'Parallel solution of large power system networks using the multi-area Thévenin equivalents (MATE) algorithm', *Int. J. Electr. Power Energy Syst.*, 2009, **31**, (9), pp. 497–503

[7] Liu, J., Benosman, M., Raghunathan, A.U.: 'Consensus-based distributed optimal power flow algorithm'. 2015 IEEE Power Energy Society Innovative Smart Grid Technologies Conf. (ISGT), Washington, DC, USA, February 2015

[8] Murray, A., Engelmann, A., Hagenmeyer, V.: 'Hierarchical distributed mixed-integer optimization for reactive power dispatch'. 10th Symp. on Control of Power and Energy Systems, Tokyo, Japan, 2018

[9] Low, S.H.: 'Convex relaxation of optimal power flow–part II: exactness', *IEEE Trans. Control Netw. Syst.*, 2014, **1**, (2), pp. 177–189

[10] Decker, I., Falcao, D., Kaszkurewicz, E.: 'An efficient parallel method for transient stability analysis'. Proc. of the Tenth Power Systems Computation Conf., Graz, Austria, 1990

[11] Scala, M.L., Sbrizzai, R., Torelli, F.: 'A pipelined-in-time parallel algorithm for transient stability analysis', *IEEE Trans. Power Syst.*, 1991, **6**, (2), pp. 715–722

[12] Ilic'-Spong, M., Crow, M.L., Pai, M.A.: 'Transient stability simulation by waveform relaxation methods', *IEEE Trans. Power Syst.*, 1987, **2**, (4), pp. 943–952

[13] Hou, L., Bose, A.: 'Implementation of the waveform relaxation algorithm on a shared memory computer for the transient stability problem', *IEEE Trans. Power Syst.*, 1997, **12**, (3), pp. 1053–1060

[14] Ng, A.Y., Jordan, M.I., Weiss, Y.: 'On spectral clustering: analysis and an algorithm'. 14th Int. Conf. on Neural Information Processing Systems: Natural and Synthetic, Vancouver, British Columbia, Canada, 2001

[15] Guo, J., Hug, G., Tonguz, O.K.: 'Intelligent partitioning in distributed optimization of electric power systems', *IEEE Trans. Smart Grid*, 2016, **7**, (3), pp. 1249–1258

[16] Schlag, S., Henne, V., Heuer, T., *et al.*: 'k-way hypergraph partitioning via n-level recursive bisection'. 2016 Proc. of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX), Arlington, Virginia, USA, 2016, pp. 53–67

[17] Drechsler, R., Gunther, W., Eschbach, T., *et al.*: 'Recursive bi-partitioning of netlists for large number of partitions', *Euromicro Symp. Digital Syst. Design*, 2002, **2002**, pp. 38–44

[18] Buluç, A., Meyerhenke, H., Safro, I., *et al.*: 'Recent advances in graph partitioning', in Lasse, Kliemann, Peter, Sanders (Eds.): '*Algorithm engineering*' (Springer, Cham, Switzerland, 2016), pp. 117–158

[19] Houska, B., Frasch, J., Diehl, M.: 'An augmented Lagrangian based algorithm for distributed nonConvex optimization', *SIAM J. Optim.*, 2016, **26**, (2), pp. 1101–1127

[20] Kyesswa, M., Çakmak, H., Kühnapfel, U., *et al.*: 'A matlab-based dynamic simulation module for power system transients analysis in the *e*ASiMOV framework'. European Modelling Symp., Manchester, UK, November 2017

[21] IEEEStd 421.5-2005: '*IEEE recommended practice for excitation system models for power system stability studies*' (IEEE, NewYork, 2006)

[22] Task Force on Turbine-Governor Modeling: '*Dynamic models for turbine-governors in power system studies*' (IEEE Power & Energy Society, USA, 2013)

[23] Kyesswa, M., Çakmak, H., Kühnapfel, U., *et al.*: 'Generator model extension for higher accuracy simulation of power system transients in OpenModelica'. MCSI, Corfu, Greece, 2017

[24] Engelmann, A., Mühlpfordt, T., Jiang, Y., *et al.*: 'Distributed AC optimal power flow using ALADIN', *IFAC-PapersOnLine*, 2017, **50**, (1), pp. 5536–5541
[25] Boyd, S., Vandenberghe, L.: '*Convex optimization*' (Cambridge university press, New York, USA, 2004)
[26] Crow, M.L.: '*Computational methods for electric power systems*' (CRC Press, Boca Raton, FL, USA, 2009)
[27] Kyesswa, M., Schmurr, P., Çakmak, H.K., *et al.*: 'A new julia-based parallel time-domain simulation algorithm for analysis of power system dynamics'. 2020 IEEE/ACM 24th Int. Symp. on Distributed Simulation and Real Time Applications (DS-RT), Prague, Czech Republic, September 2020
[28] Sanders, P., Schulz, C.: 'Think locally, act globally: highly balanced graph partitioning', in Vincenzo, Bonifaci, Camil, Demetrescu, Alberto, Marchetti-Spaccamela (Eds.): '*Experimental algorithms*' (Springer, Berlin Heidelberg, 2013), pp. 164–175
[29] Sanders, P., Schulz, C.: 'Engineering multilevel graph partitioning algorithms'. 19th European Symp. on Algorithms, Saarbrücken, Germany, 2011
[30] Maue, J., Sanders, P.: 'Engineering algorithms for approximate weighted matching'. Int. Workshop on Experimental and Efficient Algorithms, Rome, Italy, 2007
[31] Holtgrewe, M., Sanders, P., Schulz, C.: 'Engineering a scalable high quality graph partitioner'. 2010 IEEE Int. Symp. on Parallel & Distributed Processing, Atlanta, GA, USA, April 2010
[32] Sanders, P., Schulz, C.: 'High quality graph partitioning'. 10th DIMACS implementation challenge workshop: Graph Partitioning and Graph Clustering, Atlanta, GA, USA, 2013
[33] Karypis, G.: '*METIS: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 5.1.0*' (University of Minnesota, Department of Computer Science & Engineering, Minneapolis, MN, USA, March 2013)
[34] Bezanson, J., Edelman, A., Karpinski, S., *et al.*: 'Julia: a fresh approach to numerical computing', *SIAM Rev.*, 2017, **59**, (1), pp. 65–98
[35] 'Power systems test case archive - UWEE',. Available at https://www2.ee.washington.edu/research/pstca/, accessed 11 June 2018
[36] Josz, C., Fliscounakis, S., Maeght, J., *et al.*: 'AC power flow data in MATPOWER and QCQP format: iTesla, RTE Snapshots, and PEGASE', 2016. Available at http://arxiv.org/abs/1603.01533
[37] Zimmerman, R.D., Murillo-Sanchez, C.E., Thomas, R.J.: 'MATPOWER: steady-state operations, planning, and analysis tools for power systems research and education', *IEEE Trans. Power Syst.*, 2011, **26**, (1), pp. 12–19
[38] Murray, A., Kyesswa, M., Schmurr, P., *et al.*: 'On grid partitioning in AC optimal power flow'. Accepted at 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), The Hague, the Netherlands, October 2020
[39] Engelmann, A., Jiang, Y., Houska und T. Faulwasser, B.: 'Decomposition of non-convex optimization via bi-level distributed ALADIN', arXiv: Available at https://arxiv.org/abs/1903.11280
[40] Wächter, A., Biegler, L.: 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming', *Math. Program.*, 2005, **106**, (1), pp. 25–57
[41] Kyesswa, M., Çakmak, H.K., Kühnapfel, U., *et al.*: 'A matlab-based simulation tool for the analysis of unsymmetrical power system transients in large networks'. European Conf. on Modelling and Simulation (ECMS), Wilhelmshaven, Germany, May 2018