Les Cahiers du GERAD ISSN: 0711–2440

On the clique decomposition impact to the optimal power flow semidefinite relaxation solve time

V. Ricaux, A. Legrain, A. Lesage-Landry

G–2024–77

November 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

On the clique decomposition impact to the optimal power flow semidefinite relaxation solve time

Victor Ricaux [a, b]

Antoine Legrain [c, e, f]

Antoine Lesage-Landry [c, d, f]

[a] *University of California, Berkeley*

[b] *Institut Polytechnique de Paris, France*

[c] *Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4*

[d] *Mila, Montréal, (Qc), Canada, H2S 3H1*

[e] *CIRRELT, Montréal, (Qc), Canada, H3T 2B2*

[f] *GERAD, Montréal (Qc), Canada, H3T 1J4*

victor_ricaux@berkeley.edu
a.legrain@polymtl.ca
antoine.lesage-landry@polymtl.ca

November 2024
Les Cahiers du GERAD
G–2024–77

Abstract : Managing intermittent generation in electric power systems with high penetration of renewable sources of energy presents major operational challenges. Faster, more efficient optimization techniques are essential to mitigate this intermittency and ensure grid reliability. Convex relaxations of optimal power flow (OPF) problem offer tractable mean of solving the non-linear, non-convex OPF problem. Specifically, the semidefinite relaxation yields the tightest lower bounds for the OPF but require careful exploitation of sparsity to remain computationally viable when scaling to large problem instances. This exploitation can be achieved through clique decomposition of the semidefinite constraint. In this paper, we experiment with various clique decomposition algorithms and demonstrate that the resulting OPF solve time is highly sensitive to the choice of decomposition. Our main contribution is showing that the optimal decomposition depends on both the network topology and the demand profile. We categorize networks into two types: those with a preferred decomposition that performs well regardless of demand, and those where demand significantly impacts the optimal decomposition choice. This insight opens the possibility of using a learning-based approach to predict the best decomposition for minimizing OPF solve time, tailored to the network demand.

Keywords: Optimal power flow, semidefinite programming, clique decomposition, chordal extension heuristics

Résumé : Pour les réseaux à forte pénétration des renouvelables, la gestion de la génération intermittente est un défi opérationnel majeur. Des techniques d'optimisation plus rapides et plus efficaces sont essentielles pour atténuer cette intermittence et garantir la fiabilité du réseau. Les relaxations convexes du problème de l'Optimal Power Flow (OPF) offrent un moyen tractable de résoudre ce problème non linéaire et non convexe. En particulier, la relaxation semi-définie fournit de très bonnes bornes inférieures pour l'OPF, mais nécessite une exploitation rigoureuse de la parcimonie pour rester viable en termes de temps de calcul lors du passage à des instances de problèmes de grande taille. Cette exploitation peut être réalisée grâce à la décomposition en cliques de la contrainte semi-définie. Dans cet article, nous expérimentons différentes méthodes de décomposition en cliques et démontrons que le temps de résolution de l'OPF est fortement influencé par le choix de la décomposition. Notre principale contribution est de montrer que la décomposition optimale dépend à la fois de la topologie du réseau et du profil de la demande. Nous classons les réseaux en deux types : ceux avec une décomposition préférée qui performe bien indépendamment de la demande, et ceux où la demande impacte de manière significative le choix de la décomposition optimale. Cette observation ouvre la possibilité d'utiliser de l'apprentissage pour prédire la meilleure décomposition afin de minimiser le temps de résolution de l'OPF, en fonction du profil de la demande du réseau.

Mots-clés : Flux de puissance optimal, programmation semi-définie, décomposition en cliques, heuristiques d'extension cordale

1 Introduction

Transitioning from today's fuel-based power grids to renewable-based systems poses significant operational and economic challenges. This is largely due to the intermittent nature of non-dispatchable renewable energy sources, such as wind and solar, which depend on weather conditions. This intermittency introduces uncertainties and results in two fundamental differences from traditional power systems: faster timescales for power system operations and a shift from large, centralized power plants to numerous smaller distributed generation units [24]. To address these changes, more efficient optimization methods are essential to mitigate the variability of renewables via the flexibility of distributed generation infrastructure that can be rapidly re-dispatched.

The optimal power flow (OPF) is a key optimization problem that determines the optimal power dispatch to minimize generation costs while adhering to grid operational constraints, like power line and bus voltage bounds [25]. However, due to the nature of power flow equations, the OPF is a non-convex quadratic problem and is thus NP-hard [6]. Various approaches have been proposed to solve the OPF to near optimally, including (linear) approximations (e.g., `DC-OPF`) and convex relaxations [16,25]. The semidefinite relaxation (`SDR-OPF`) is one of the tightest convex relaxations but is computationally expensive. We refer to the chordal relaxation (`CR-OPF`) the equivalent form to `SDR-OPF` that leverages the sparsity of the grid to accelerate computations in semidefinite programming (SDP) [3,13]. However, selecting the optimal chordal decomposition remains challenging as the solution time of the resolution is highly sensitive to the selected decomposition. Current methods typically involve heuristics to minimize the edges added to obtain the chordal extension, followed by clique merging strategies to balance clique size and count.

Alternative strategies, such as genetic algorithms [9] and end-to-end learning models [19, 28], have also been explored for rapid resolution of the OPF. These methods often lack convergence guarantees and/or their approximation nature compromises more on feasibility. Hybrid approaches that combine optimization and learning, such as [4], use machine learning predictions to accelerate traditional solvers. Recent research has explored clique decomposition learning methods, where [1] predicts the computational efficiency of decompositions, and [15] applies imitation learning to mimic chordal extension heuristics.

Contribution. Unlike previous studies that focus on optimizing the overall performance of chordal decomposition heuristics across different networks, this paper examines how loading patterns influence decomposition quality in terms of OPF solve time for a fixed network topology. We demonstrate that for specific grids, the effectiveness of decomposition heuristics can significantly vary based on demand profiles. This approach is motivated by the fact that the topology of networks remains relatively stable over time, whereas demand conditions fluctuates on much shorter time scales.

2 Optimal power flow and `SDR-OPF`

The OPF involves determining the optimal operating conditions of an electric power system to minimize generation costs while satisfying network constraints. Formally, the power system is represented as a graph $(\mathcal{N}, \mathcal{L})$, where $\mathcal{N} \subset \mathbb{N}$ is the set of vertices, i.e., buses, and $\mathcal{L} \subset \mathbb{N} \times \mathbb{N}$ is the set of edges, i.e., power lines. Let the admittance of line $ij \in \mathcal{L}$ be denoted by $y_{ij} \in \mathbb{C}$. Let $p_i \in \mathbb{R}$ and $q_i \in \mathbb{R}$ be, respectively, the active and reactive power injection/absorption at bus $i \in \mathcal{N}$. Similarly, let $p_{ij} \in \mathbb{R}$ and $q_{ij} \in \mathbb{R}$ be, respectively, the active and reactive power flow in line $ij \in \mathcal{L}$. Lastly, let $v_i \in \mathbb{C}$ be the complex voltage phasor at bus i. The non-linear, non-convex OPF (`AC-OPF`) problem is expressed as:

$$\min_{v,p,q} \quad \sum_{i\in\mathcal{N}} c_{i,2}p_i^2 + c_{i,1}p_i + c_{i,0} \tag{1a}$$

$$\text{s.t.} \quad p_{ij} + \mathbf{j}q_{ij} = v_i(v_i^* - v_j^*)y_{ij}^*, \quad ij \in \mathcal{L} \tag{1b}$$

$$\underline{v} \le |v_i| \le \overline{v}, \quad i \in \mathcal{N} \tag{1c}$$

$$\sum_{j:ij\in\mathcal{L}} p_{ij} = p_i, \quad \sum_{j:ij\in\mathcal{L}} q_{ij} = q_i, \quad i \in \mathcal{N} \tag{1d}$$

$$\underline{p}_i \le p_i \le \overline{p}_i, \quad \underline{q}_i \le q_i \le \overline{q}_i, \quad i \in \mathcal{N} \tag{1e}$$

$$\sqrt{p_{ij}^2 + q_{ij}^2} \le \overline{s}_{ij}, \quad ij \in \mathcal{L} \tag{1f}$$

where (1a) is a convex generation cost function commonly set as a convex quadratic function [25] with coefficients denoted $c_{i,0}$, $c_{i,1}$ and $c_{i,2}$. (1b) is the power flow constraint, (1c) imposes the minimum/maximum ($\underline{v}$ and $\overline{v}$, respectively) voltage magnitude limits, (1d) represents the active and reactive nodal power balance, (1e) imposes the minimum and maximum active and reactive power limits, and (1f) enforces the apparent power limit $\overline{s}_{ij}$ for each line.

A common set of OPF convex relaxations lifts the dimensionality of the voltage variable through the introduction of a Hermitian matrix $\mathbf{W} \in \mathbb{C}^{n\times n}$, defined as $\mathbf{W} = \mathbf{v}\mathbf{v}^{\mathrm{H}}$, where $\mathbf{v}$ is the voltage vector. This allows us to rewrite (1b) as a linear constraint, in addition to rewriting (1c), yielding:

$$p_{ij} + \mathbf{j}q_{ij} = (\mathbf{W}_{ii} - \mathbf{W}_{ij})y_{ij}^*, \quad ij \in \mathcal{L}, \tag{2a}$$

$$\underline{v}^2 \le \mathbf{W}_{ii} \le \overline{v}^2, \tag{2b}$$

$$\mathbf{W} \succeq 0, \tag{2c}$$

$$\mathrm{rank}(\mathbf{W}) = 1 \tag{2d}$$

where (2c), viz., a positive semidefinite (PSD) constraint, and (2d) are a reformulation of the equality $\mathbf{W} = \mathbf{v}\mathbf{v}^{\mathrm{H}}$. The resulting problem, i.e., (1a) subject to (1d)–(1f), (2a)–(2d), is equivalent to the original formulation [25]. Finally, omitting (2d) leads to the semidefinite relaxation (`SDR-OPF`).

3 Sparsity exploitation and clique decompositions

The `SDR-OPF` is usually a fairly sparse optimization problem and only the coefficients $\mathbf{W}_{ij}$ for which $ij \in \mathcal{L}$ are involved in the problem. This results in a PSD constraint on a large matrix where many of its coefficients are free. Under certain assumptions, which we will detail in this section, it is possible to transform this constraint into several PSD constraints on smaller matrices, making the problem less computationally expensive. Understanding this transformation requires introducing chordal graphs and clique decompositions. This approach is rooted in the theory of matrix completion, which aims to fill in the missing entries of a partially observed matrix. Decomposing the PSD contraint on $\mathbf{W}$ is feasible under specific conditions, namely the chordality of the graph but comes at the cost of adding linking constraints between the submatrices obtained.

3.1 Chordal graphs and sparsity patterns

Consider a symmetric adjacency pattern of order $n \in \mathbb{N}$, defined as a set of index pairs $\mathcal{E} \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$, where each pair $(i, j) \in \mathcal{E}$ represents a non-zero element in an $n \times n$ adjacency matrix. We can associate this pattern $\mathcal{E}$ with an undirected graph $\mathcal{G}$ consisting of n nodes, where an edge connects nodes i and j if $(i, j) \in \mathcal{E}$.

A graph is called chordal if every cycle of length four or more has a chord, which is an edge connecting two non-adjacent nodes in the cycle [26]. This property ensures that certain sparse matrix structures can be decomposed into smaller, more manageable components.

3.2 Clique decomposition steps

For a sparse SDP problem, e.g., `SDR-OPF`, the clique decomposition process involves several steps explained next.

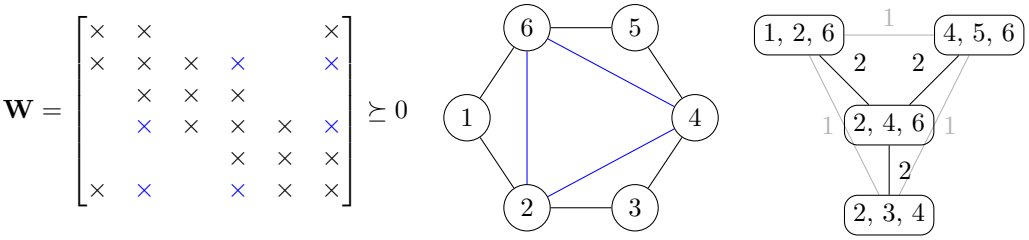

Figure 1: Adjacency matrix of W and the corresponding chordal graph and clique tree

3.2.1 Sparsity pattern

The sparsity pattern summarizes all non-zero entries across the problem's data matrices and can be viewed as an adjacency matrix. This is illustrated in Figure 1, were we consider that the original network is a cycle with six nodes. The black entries of $\mathbf{W}$ then represent the sparsity pattern of that network.

3.2.2 Chordal extension

Because not all graphs are naturally chordal, we extend the original network to a chordal graph by adding edges. This is shown in Figure 1 where we add the blue edges to the original graph and the corresponding blue entries in $\mathbf{W}$. While finding the minimal chordal extension is NP-complete, effective heuristics, such as the approximate minimum degree (AMD) ordering followed by a Cholesky factorization, are commonly used [2].

3.2.3 Maximal clique identification

A clique is a subset of nodes where every pair is connected. A maximal clique is a clique not entirely contained within another. Listing these cliques from a chordal graph can be done in linear time with Bron and Kerbosh's algorithm [7]. Now that we have a chordal graph and the set of its maximal cliques, we can rewrite the constraint (2c) as multiple PSD constraints on each submatrix associated with a clique. To ensure that these submatrices reconstruct the original matrix $\mathbf{W}$ correctly, we must add linking constraints between these submatrices.

3.2.4 Linking constraints

As the cliques may overlap, linking constraints ensure consistency between intersecting submatrices. To achieve this, we first construct the clique graph of our chordal graph. This graph has nodes representing the cliques of the original graph, see Figure 1. In this graph, two nodes, i.e., two cliques, are connected if their corresponding cliques share common nodes. We can assign a weight to these edges, which represents the number of shared nodes. These edges describe the linking constraints between the submatrices. Choosing all the edges results in many redundant constraints due to cycles in the clique graph. To minimize these linking constraints, we seek a clique graph covering that satisfies two properties: it is acyclic and satisfies the *running intersection property* [26]. The latter ensures that for an entry of $\mathbf{W}$, all the submatrices that contain it are equal at that location, see Figure 2. The covering that meets these properties is the maximum spanning tree [26], and it can be obtained using Prim's algorithm [21]. We refer to it as the clique tree, see Figure 1.

3.3 Clique merging

In practice, especially in cases with extremely sparse problems like those in the OPF, the decomposition may yield numerous small cliques. To address this, a post-processing step called clique merging is often employed [17, 22]. This process involves adding edges to the chordal graph to combine adjacent cliques,

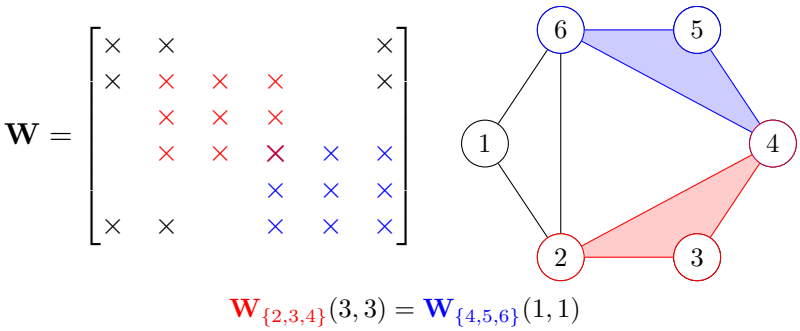

Figure 2: Linking constraints between submatrices corresponding to two overlapping cliques

thereby reducing the number of linking constraints, provided that the chordal property is preserved. This formulation is particularly relevant when handling complex SDPs, where converting complex variables into real variables may necessitate doubling the clique sizes.

Clique merging algorithms can be based on the clique tree [23] or on the clique graph [12]. In the remainder of this paper, we consider the latter and use [12]'s clique graph-based merging strategy provided in Algorithm 1. As show in Figure 1, the edge weights of the clique graph are usually given by the number of nodes common to the cliques at both ends of the edge. In [12]'s clique merging algorithm, a new edge weighting function is introduced: $w_{ij} = |\gamma_i|^3 + |\gamma_j|^3 - |\gamma_i \cup \gamma_j|^3$, where $|\gamma_i|$ and $|\gamma_j|$ are two cliques, i.e., two sets of nodes of the original graph. This function is typically negative, except in cases where there is significant overlap between cliques. Algorithm 1 iteratively merges the cliques i and j for which $w_{ij} \geq 0$ and then recomputes the weights w_{ij} of the updated clique graph until $w_{ij} < 0$ for all i and j.

Algorithm 1: Clique merging [12]

Input: A chordal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{1, 2, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
Output: An updated clique graph
Construct the intersection graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ with $\mathcal{V}_c = \{\gamma_1, \ldots, \gamma_m\}$ and $\mathcal{E}_c = \{\{\gamma_i, \gamma_j\} \mid \gamma_i \cap \gamma_j \neq \emptyset\}$
Set $w_{ij} = |\gamma_i|^3 + |\gamma_j|^3 - |\gamma_i \cup \gamma_j|^3$ for each $\{\gamma_i, \gamma_j\} \in \mathcal{E}_c$
While the clique graph $\mathcal{G}_c$ contains positive weights **do**
  Select two admissible cliques γ_i and γ_j with the highest w_{ij}
  Merge the cliques γ_i and γ_j
  Update the clique graph $\mathcal{G}_c$
  Recompute the weights w_{ij} for the updated clique graph $\mathcal{G}_c$
End while

3.4 Node ordering heuristics for the chordal extension

The Elimination Game [20] given in Algorithm 2 simulates Gaussian elimination on graphs. Fulkerson [11] showed that the class of graphs produced by Elimination Game is exactly the class of chordal graphs, making it a foundational method for obtaining chordal extensions. Algorithm 2 operates on a graph $\mathcal{G}$ and a node ordering σ, systematically adding edges to transform neighbouring nodes into cliques. A chordal extension is therefore entirely determined by a node ordering of the original graph.

We define the Cholesky factorization of a PSD symmetric matrix $\mathbf{A}$ as a decomposition $\mathbf{P}_\sigma \mathbf{A} \mathbf{P}_\sigma^\top = \mathbf{L}\mathbf{D}\mathbf{L}^\top$, where $\mathbf{L}$ is a lower triangular matrix, $\mathbf{D}$ is a positive diagonal matrix, and $\sigma = (\sigma(1), \ldots, \sigma(n))$ is a permutation of $\{1, 2, \ldots, n\}$.

The adjacency matrix $\mathbf{A}$ of a graph $\mathcal{G}$ is by definition symmetric. If self-loops are not allowed in the graph, the diagonal of the adjacency matrix can be inflated arbitrarily without changing the graph's structure. This process can continue until the matrix becomes diagonally dominant and therefore PSD. If we let the matrix $\mathbf{A}$ be the PSD adjacency matrix of a graph $\mathcal{G}$, then the chordal extension $\mathcal{G}_\sigma^+$

Algorithm 2: Elimination game [20]

Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and σ, an ordering of the nodes
Output: A chordal extension $\mathcal{G}_\sigma^+$ of the graph $\mathcal{G}$
$\mathcal{G}_\sigma^1 \leftarrow \mathcal{G}$
$\mathcal{G}_\sigma^+ \leftarrow \mathcal{G}$
For $k = 1$ to n **do**
  Let $\mathcal{F}$ be the set of edges needed to turn the neighbourhood of node v_k into a clique in $\mathcal{G}_\sigma^k$
  $\mathcal{G}_\sigma^{k+1} \leftarrow \mathcal{G}_\sigma^k + \mathcal{F} - \{v_k\}$
  $\mathcal{G}_\sigma^+ \leftarrow \mathcal{G}_\sigma^+ + \mathcal{F}$
End for

obtained after doing the graph elimination of $\mathcal{G}$ is exactly the graph described by the sparsity pattern of $\mathbf{L} + \mathbf{L}^T$ where $\mathbf{L}$ is the Cholesky factor of $\mathbf{A}$ with permutation σ . Indeed, it can be shown [26] that if $\mathbf{A} \in \mathbb{S}^n_\mathcal{E}$, then $\mathbf{P}_\sigma(\mathbf{L} + \mathbf{L}^\top)\mathbf{P}_\sigma^\top \in \mathbb{S}^n_{\mathcal{E}'}$, where $\mathcal{E}'$ is the chordal extension of $\mathcal{E}$ and $\mathbb{S}^n_\mathcal{E}$ is the set of symmetric matrices of size n with adjacency structure $\mathcal{E}$. The adjacency structure of $\mathbf{L} + \mathbf{L}^\top$ is thus the chordal extension of the adjacency structure of $\mathbf{A}$.

We therefore aim to find a permutation σ that minimizes the fill-in [10], meaning that we want the Cholesky factor $\mathbf{L}$ of the adjacency matrix $\mathbf{A}$ of the network to be as sparse as possible. The question of optimal chordal extension in terms of OPF solve time thus translates to finding the best node ordering. We explore various heuristics to determine σ. Although the problem of finding the ordering the minimizes the number of added edges in the chordal extension is NP-complete [27], there are several efficient heuristics available. In this work, we implement the following heuristics:

Minimum degree (`MD`): at each iteration, the node with the smallest degree is chosen.

Approximate minimum degree (`AMD`): employs the same algorithm as `MD` but with bounds on the node degrees.

Minimum fill-in (`MFI`): at each iteration, the node that minimizes the number of edges to be added among its neighbours so that they form a clique is chosen.

These heuristics were selected because they provided the best results in terms of solve time. Other heuristics such as the *Maximum Cardinality Search (`MCS-M`)* [5] were tested but were set aside from the final results due to their significantly poorer performance compared to the above three algorithms.

4 Numerical tests

To test the chordal extension heuristics, we solve `CR-OPF` on the RTE [14] and MATPOWER [29] benchmarks with `MOSEK` [18]. To formulate this problem, we use the Julia library `PowerModels.jl` [8]. We note that chordal extension processes are completed in negligible time compared to the solving times with `MOSEK` and are therefore neglected. Tests are conducted on an 11th Gen i7-11700F processor.

4.1 Test system results

First, we compare the performance of the heuristics on about ten cases with over 1000 buses. For each chordal extension heuristic (`AMD`, `MD`, and `MFI`), we conduct tests both with and without clique merging. The resulting `CR-OPF` solve times are given in Table 1. We observe that, for a given case, the different heuristics provide distinct decompositions with varying performance. We also note that: (i) networks of similar size (in terms of the number of buses) can have very different solve times; (ii) decompositions without clique merging are on average less efficient than those with clique merging for the considered test systems; (iii) there is no single decomposition that consistently outperforms all others across all networks: this suggests that certain network topologies favour specific decompositions; (iv) in some cases, the `AMD` decomposition without clique merging offers better solve times than all decompositions with clique merging; (v) `MFI` with clique merging generally provides the best results on average, closely followed by the `AMD` heuristic with clique merging.

Table 1: Relative resolution time difference for different test cases

Case	With merge			Without merging		
	AMD	MD	MFI	AMD	MD	MFI
case1888rte	**9.96**	+20%	+9%	+38%	+65%	+41%
case1951rte	+4%	+6%	**11.23**	+34%	+37%	+43%
case2736sp	+30%	+37%	+5%	**27.56**	+27%	+7%
case2737sop	**23.33**	+45%	+18%	+11%	+29%	+19%
case2746wop	+15%	+29%	**33.27**	+18%	+24%	+13%
case2746wp	+4%	+45%	+3%	**28.05**	+22%	+2%
case2848rte	**17.60**	+26%	+6%	+36%	+35%	+22%
case2868rte	**20.17**	+2%	+2%	+24%	+14%	+32%
Mean	+8%	+26%	**+5%**	+20%	+32%	+22%

4.2 Sensitivity to the demand profile

To highlight the impact of loads on chordal decomposition heuristics, we vary the demand profile for a given network. For each bus i, we perturb its active power p_i and reactive power q_i by adding Gaussian noise $\mathcal{N}(0, \Sigma_i)$. The standard deviation Σ_i is set as $\Sigma_i = \alpha p_i$ for active power and $\Sigma_i = \alpha q_i$ for reactive power, where α is a constant. This ensures perturbations are proportional to the original demand. We use $\alpha = 0.5$ to create significant perturbations without making the problem too frequently infeasible. We compare performance on the `case1951rte` and `case2746wop` networks, with 100 perturbed demand profiles each. For each perturbation, we measure each heuristic's performance based on the solve time of the resulting `CR-OPF`.

4.2.1 `case1951rte`

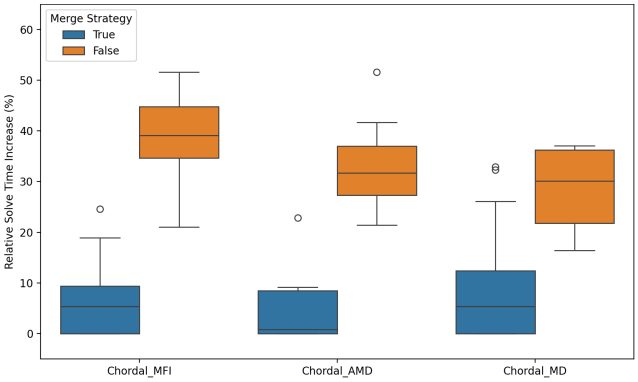

Figure 3: Relative solve times (%) for `case1951rte` (100 demand profiles)

We consider the `case1951rte` test system for 100 loading profiles. The performances of each heuristics are shown in Figure 3. For this network, the optimal decomposition varies depending on the demand. The heuristics without clique merging consistently perform worse than those with merging. We note that each clique merging algorithm provides the best results about one-third of the time. This points to an opportunity of using a learning-based approach to predict the best decomposition for minimizing `CR-OPF` solve time, tailored to the network demand.

4.2.2 `case2746wop`

We now move to the `case2746wop` test system. Figure 4 shows that `AMD` with clique merging outperforms other approaches in almost all considered demand profiles. We also observe that the ranking of heuristics is not the same as with the previous network: each heuristic responds differently to each

topology, and there is no clear link between topology type and the preferred heuristic. Similarly, demand profile perturbations corresponding to a large increase in demand would favour one heuristic over another. However, we could not establish such a link. This further supports using learning-based methods to predict the best heuristic based on network topology and demand loading.

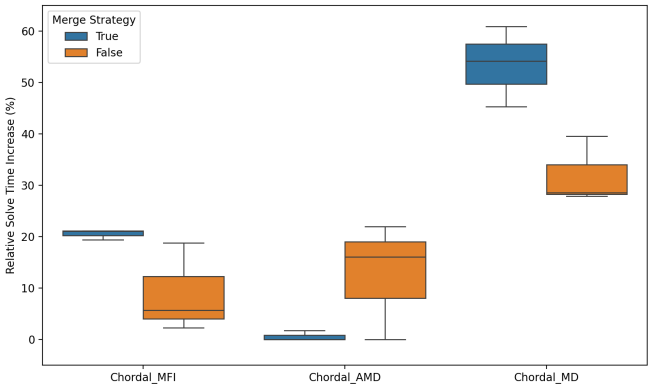

Figure 4: Relative solve times (%) for `case2746wop` (100 demand profiles)

5 Conclusion

In this paper, we explore ways of accelerating the Optimal Power Flow (OPF) by applying chordal relaxation techniques and improving the choice of decomposition. Our implementation of `CR-OPF` reveals that different test systems respond uniquely to chordal extension heuristics, highlighting the potential benefits of using machine learning to select the best relaxation method based on demand profiles. Additionally, clique merging has proven to be an effective operation for reducing OPF resolution time, though its effectiveness varies depending on the case and demand fluctuations. Moving forward, optimizing the weight function for clique merging and employing machine learning techniques to select the best heuristic could improve solve time on specific grids.

References

[1] Charly Alizadeh, Pegah Alizadeh, Miguel F Anjos, Lucas Létocart, and Emiliano Traversi. How to learn the optimal clique decompositions in solving semidefinite relaxations for OPF. In 2022 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2022.

[2] Patrick R Amestoy, Timothy A Davis, and Iain S Duff. An approximate minimum degree ordering algorithm. SIAM Journal on Matrix Analysis and Applications, 17(4):886–905, 1996.

[3] Xiaoqing Bai and Hua Wei. A semidefinite programming method with graph partitioning technique for optimal power flow problems. International Journal of Electrical Power & Energy Systems, 33(7):1309–1314, 2011.

[4] Kyri Baker. Learning warm-start points for AC optimal power flow. In 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), pages 1–6, 2019.

[5] Anne Berry, Jean RS Blair, and Pinar Heggernes. Maximum cardinality search for computing minimal triangulations. In Graph-Theoretic Concepts in Computer Science: 28th International Workshop, pages 1–12. Springer, 2002.

[6] Daniel Bienstock and Abhinav Verma. Strong NP-hardness of AC power flows feasibility. Operations Research Letters, 47(6):494–501, 2019.

[7] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM, 16(9):575–577, 1973.

[8] Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. Powermodels.jl: An open-source framework for exploring power flow formulations. In Power Systems Computation Conference (PSCC), pages 1–8. IEEE, 2018.

[9] Stephen Frank, Ingrida Steponavice, and Steffen Rebennack. Optimal power flow: A bibliographic survey II: Non-deterministic and hybrid methods. Energy Systems, 3:259–289, 2012.

[10] Mituhiro Fukuda, Masakazu Kojima, Kazuo Murota, and Kazuhide Nakata. Exploiting sparsity in semidefinite programming via matrix completion i: General framework. SIAM Journal on optimization, 11(3):647–674, 2001.

[11] Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. Pacific journal of mathematics, 15(3):835–855, 1965.

[12] Michael Garstka, Mark Cannon, and Paul Goulart. A clique graph based merging strategy for decomposable sdps. IFAC-PapersOnLine, 53(2):7355–7361, 2020.

[13] Rabih A Jabr. Exploiting sparsity in SDP relaxations of the OPF problem. IEEE Transactions on Power Systems, 27(2):1138–1139, 2011.

[14] Cédric Josz, Stéphane Fliscounakis, Jean Maeght, and Patrick Panciatici. AC power flow data in MATPOWER and QCQP format: itesla, rte snapshots, and pegase. arXiv preprint arXiv:1603.01533, 2016.

[15] Defeng Liu, Andrea Lodi, and Mathieu Tanneau. Learning chordal extensions. Journal of Global Optimization, 81(1):3–22, 2021.

[16] Daniel K Molzahn, Ian A Hiskens, et al. A survey of relaxations and approximations of the power flow equations. Foundations and Trends® in Electric Energy Systems, 4(1-2):1–221, 2019.

[17] Daniel K Molzahn, Jesse T Holzer, Bernard C Lesieutre, and Christopher L DeMarco. Implementation of a large-scale optimal power flow solver based on semidefinite programming. IEEE Transactions on Power Systems, 28(4):3987–3998, 2013.

[18] APS Mosek. The MOSEK optimization software. Online at http://www.mosek.com, 54(2-1):5, 2010.

[19] Rahul Nellikkath and Spyros Chatzivasileiadis. Physics-informed neural networks for AC optimal power flow. Electric Power Systems Research, 212:108412, 2022.

[20] Seymour Parter. The use of linear graphs in Gauss elimination. SIAM review, 3(2):119–130, 1961.

[21] Robert Clay Prim. Shortest connection networks and some generalizations. The Bell System Technical Journal, 36(6):1389–1401, 1957.

[22] Julie Sliwak, Erling D Andersen, Miguel F Anjos, Lucas Létocart, and Emiliano Traversi. A clique merging algorithm to solve semidefinite relaxations of optimal power flow problems. IEEE Transactions on Power Systems, 36(2):1641–1644, 2020.

[23] Yifan Sun, Martin S Andersen, and Lieven Vandenberghe. Decomposition in conic optimization with partially separable structure. SIAM Journal on Optimization, 24(2):873–897, 2014.

[24] Josh A Taylor, Sairaj V Dhople, and Duncan S Callaway. Power systems without fuel. Renewable and Sustainable Energy Reviews, 57:1322–1336, 2016.

[25] Joshua Adam Taylor. Convex Optimization of Power Systems. Cambridge University Press, 2015.

[26] Lieven Vandenberghe, Martin S Andersen, et al. Chordal graphs and semidefinite optimization. Foundations and Trends® in Optimization, 1(4):241–433, 2015.

[27] Mihalis Yannakakis. Computing the minimum fill-in is NP-complete. SIAM Journal on Algebraic Discrete Methods, 2(1):77–79, 1981.

[28] Ahmed S Zamzam and Kyri Baker. Learning optimal solutions for extremely fast AC optimal power flow. In 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pages 1–6, 2020.

[29] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Transactions on Power Systems, 26(1):12–19, 2010.