A sparse Convex AC OPF Solver and Convex Iteration Implementation Based on 3-Node Cycles

Minyue Ma, Lingling Fan*, Zhixin Miao, Bo Zeng, Hossein Ghassempour

**Department of Electrical Engineering, University of South Florida, Tampa FL USA 33620.*

Phone: 1(813)974-2031, Email: linglingfan@usf.edu.

Abstract

In this paper, we exploit sparsity technique and strengthen second-order cone programming (SOCP) relaxation of alternating current optimal power flow (AC OPF) through enforcing submatrices corresponding to maximal cliques and minimal chordless cycles in the cycle basis positive semi-definite (PSD). The proposed method adds virtual lines in minimal chordless cycles to decompose each of them into 3-node cycles. By enforcing the submatrices related to 3-node cycles PSD, the resulting convex relaxation has a tight gap. For majority of the test instances, the resulting gap is as tight as that of a semi-definite programming (SDP) relaxation, yet the computing efficiency is much higher. Further, convex iteration is implemented on the proposed solver to achieve exactness by enforcing all submatrices corresponding to lines and virtual lines rank-1. Test results from small size to large size with thousands of buses demonstrate the capability of the proposed solver and the feasibility of the convex iteration implementation.

Keywords: SOCP; SDP; AC OPF; Chordal Relaxation; Sparsity Technique; Convex Iteration

1. Introduction

SDP relaxation of AC OPF has shown to be a very strong convex relaxation to the original non-convex formulation [1, 2]. Nevertheless, the disadvantage of SDP relaxation is its expensive computational cost.

For that reason, sparse technique has been exploited for SDP relaxation [3, 4, 5]. The main theorem is the positive semidefinite (PSD) matrix completion theorem [6], which states that if every submatrix related to every maximal clique in a chordal graph is PSD, then the partial symmetric matrix X_{ch} corresponding to the chordal graph can be completed as a full PSD matrix $X \succeq 0$. The graph related to the topology of a power grid is usually not a chordal graph. To obtain a chordal graph, Cholesky factorization has been used to find chordal extension [3]. Detailed implementation procedure of Cholesky factorization and sparse SDP relaxation can be found in [4]. Instead of finding maximal cliques and further a clique tree through

Cholesky factorization, tree width decomposition can also be used to find a clique tree [5]. This method has been implemented in a software package for SDP relaxation of OPF [7].

The aforementioned researches focus on sparse SDP relaxation. On the other hand, there is a category of research focusing on strengthening SOCP relaxation [8, 9]. Compared to SDP relaxation, SOCP relaxation is computationally more efficient. Nevertheless, the feasible region of SOCP relaxation is less tight. Strengthening SOCP relaxation has been studied in [8, 9] by implementing cutting plane algorithms, i.e., iteratively adding valid inequalities (aka. cuts), including SDP based ones. The principle of the methods in [8, 9] is based on the fact that for a PSD matrix, its submatrices corresponding to cycles in a cycle basis are PSD. If a submatrix of the solution of the SOCP is not PSD, a valid inequality can be constructed to reduce the search region. The constraint can be constructed using duality concept in [8] and shortest Euclidean distance technique in [9].

In this paper, we explore an alternative computationally friendly method that can strengthen SOCP relaxation. Instead of iteratively solving and strengthening the SOCP relaxation by cutting plane algorithms, we propose to directly add maximal clique-based and cycle-based SDP feasibility constraints in the SOCP relaxation. Those added constraints enforce the submatrices related to maximal cliques and cycles to be PSD.

Further, we conduct chordal relaxation for chordless cycles. A chordless cycle of size n can be decomposed into 3-node cycles or cliques by adding $(n-3)$ virtual edges. Adding virtual lines has also been adopted by other researchers. For example, [10] proposed to add virtual lines between the reference bus and all its non-adjacent buses. By enforcing all submatrices related to the virtual lines PSD, the resulting convex relaxation in [10] is stronger than the original SOCP.

Compared to [10], our method of virtual line addition based on chrodless cycle 3-node decomposition results in less virtual lines and thus is more efficient. Overall, this method results in a stronger convex relaxation compare to SOCP relaxation. All 2×2 principal submatrices of the full matrix are guaranteed to be PSD. Further, all maximal cliques with size greater than 2 in the original power grid graph, and the 3-node cycles constructed from chordless cycles are PSD. The computing efficiency has been compared with sparse SDP methods [4] [5] and is found to be higher. Though the graph after chordal extension is not guaranteed to be a chordal graph, the resulting formulations in many cases are as strong as SDP relaxation.

Finally, based on the proposed convex relaxation, convex iteration is carried out based on 3-node cycles. Convex iteration based on SDP OPF has been implemented in [11][12]. SOCP exactness condition [13] states that the exactness condition is for the submatrices related to two nodes of a line PSD and rank-1, and cycle constraints (sum of the angles across a cycle is zero) satisfied. With every cycle in a cycle basis has been decomposed into 3-node cycles, the cycle constraints can be replaced by the cycle constraints of 3-node cycles. The exactness condition thus requires that every submatrix corresponding to every 3-node cycle is PSD and rank-1. This requirement is further translated to an equivalent requirement: all 3×3 submarices corresponding to 3-node cycles should be PSD and every 2×2 submatrix corresponding to lines and virtual lines should be rank-1. Convex iteration is then implemented to enforce those 2×2 submatrices rank-1. The resulting solution should be a feasible solution.

Our contribution is threefold. *First*, we answer a question naturally arise from the research results from [8] and [9]: Will a PSD solution be found if its SOCP solution's submatrices related to cycles in a cycle basis are PSD? We demonstrate that chordal relaxation for every cycle in a cycle basis cannot result in a chordal graph. Hence, there is no guarantee that the strengthened SOCP in [8] and [9] can lead to SDP solution eventually. *Second*, we propose a stronger convex relaxation compared to SOCP by enforcing minimal cycles of a cycle basis PSD. This enforcement can be further replaced by 3-node cycle PSD enforcement. The proposed solver is implemented in CVX platform and shows higher computing efficiency compared with sparse SDP methods [4] [5]. *Third*, based on the 3-node cycles, we implement convex iteration to enforce the submatrices related to the 3-node cycles PSD and rank-1. An even more efficient rank-1 enforcement is then derived. With the proposed sparse solver, enforcing all 2×2 submatrices related to lines and virtual lines rank-1 will lead to a feasible solution.

The rest of the paper is organized as follows. Section II presents SOCP and SDP relaxations of AC OPF. Section III presents the maximal clique- and cycle-based SOCP formulations. Section IV presents the convex iteration method. Numerical results are presented in Section V. Section VI concludes the paper.

2. SOCP and SDP Relaxations of AC OPF

2.1. SOCP and SDP relaxations

The decision variables of AC OPF are voltage magnitudes V, phase angles θ of buses with the bus set notated as $\mathcal{B}$, and generators' real and reactive power outputs, notated as P_i^g, Q_i^g, where $i \in \mathcal{G}$, and $\mathcal{G} \subseteq \mathcal{B}$ is the subset of the buses. The set of the branches is notated as $\mathcal{L}$.

The AC OPF formulation is a nonconvex optimization problem. This can be shown by the power injection equality constraints. Given the system admittance matrix $Y = G + jB$, the power injection at every node can be expressed by V and θ.

$$\begin{aligned}
P_i^g - P_i^d &= \sum_{j \in \delta_i} V_i V_j (G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)) \\
Q_i^g - Q_i^d &= \sum_{j \in \delta_i} V_i V_j (G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j))
\end{aligned} \tag{1}$$

where superscript g notates generator's output and d notates load consumption, δ_i is the set of the buses that are directly connected to Bus i.

Note that the equality constraints of power injections are nonlinear in terms of V and θ. This yields the AC OPF problem nonconvex. Relaxations have been developed in the literature to have a convex feasible region. These methods deal with new sets of decision variables to replace V and θ.

In SOCP relaxation [14], a new set of variables c_{ij} and s_{ij} is used to replace the voltage phasors $V_i \angle \theta_i, i \in \mathcal{B}$.

$$\begin{aligned}
c_{ii} &= V_i^2, \ i \in \mathcal{B} \\
c_{ij} &= V_i V_j \cos(\theta_i - \theta_j), \ (i,j) \in \mathcal{L} \\
s_{ij} &= -V_i V_j \sin(\theta_i - \theta_j), \ (i,j) \in \mathcal{L}
\end{aligned}$$

where $c_{ij} = c_{ji}$ and $s_{ij} = -s_{ji}$.

It is easy to find the following relationship:

$$c_{ij}^2 + s_{ij}^2 = V_i^2 V_j^2 = c_{ii} c_{jj}. \tag{2}$$

There will be $|\mathcal{L}|$ number c_{ij} and s_{ij}. If there is no direct connection between Bus i and Bus j, the power injection equations will not contain c_{ij} nor s_{ij}. The decision variables V and θ are replaced by $c_{ii}, i \in \mathcal{B}$, and $c_{ij}, s_{ij}, (i,j) \in \mathcal{L}$. The dimension of the new set of the variables is $2|\mathcal{G}| + |\mathcal{B}| + 2|\mathcal{L}|$.

With the new set of variables, power injection equations (1) are now linear. The line flow limit constraints become second-order cone constraints. The only constraint that makes the program nonconvex is (2). This constraint can be relaxed as a second-order cone:

$$c_{ij}^2 + s_{ij}^2 \leq c_{ii} c_{jj}, \quad (i,j) \in \mathcal{L} \tag{3}$$

This relaxation was first proposed in [14] for AC OPF and named as SOCP relaxation.

In SDP relaxation proposed by [15], the decision variables related to voltage phasors are replaced by a matrix $X = \overline{V}\overline{V}^H$, where $X_{ij} = \overline{V}_i\overline{V}_j^* = c_{ij} - js_{ij}$, superscript H denotes Hermitian transpose for a vector and superscript $*$ means complex conjugate for a scalar.

(3) can be rewritten as

$$\begin{vmatrix} X_{ii} & X_{ij} \\ X_{ji} & X_{jj} \end{vmatrix} = X_{ii}X_{jj} - X_{ij}X_{ji} \geq 0. \tag{4}$$

Based on the definition $X = \overline{V}\overline{V}^H$, it is obvious that X is PSD and rank-1.

$$X = X^H, \quad X \succeq 0, \text{and } \text{rank}(X) = 1 \tag{5}$$

$X \succeq 0$ means that this matrix is PSD.

The power injection constraints are linear with the elements of X. With the rank-1 constraint relaxed, the problem is a convex problem: SDP relaxation of AC OPF. For tree networks, SOCP relaxation and SDP relaxation are equivalent [2]. For meshed network, the SOCP constraint (3) or (4) enforces only 2×2 principal submatrices related to lines PSD. For cliques with sizes greater than 2 and cycles, SOCP relaxation does not guarantee the related submatrices PSD.

3. Prposed Sparse Convex Relaxation Formulation

Instead of dealing with a full matrix X for the entire grid, for each maximal clique and each cycle in the cycle basis of the network, we impose the PSD constraint for the corresponding submatrix $\widetilde{X}^{(i)}$. This is equivalent to first conduct chordal extension to make a chordless cycle of size n a clique and then enforce the related $n \times n$ submatrix PSD. On the other hand, there is a more efficient way of chordal extension for a chordless cycle: A cycle can be decomposed into 3-node cycles. This approach can save computing cost due to the reduction of the size of the PSD matrices (all 3×3). We further examine if such chordal relaxation can lead to a chordal graph for the entire power grid. If so, we have a sparse SDP relaxation. If not, we have a stronger convex relaxation compared to SOCP.

In this section, we first review a few graph theory techniques that will be used to identify maximal cliques and chordless cycles. We then examine the proposed chordal extension for various examples and check if they can lead to a chordal graph. Finally, we give the proposed sparse convex relaxation formulation.

3.1. Maximal Cliques Identification

Given a graph's boolean adjacency matrix, all maximal cliques can be identified using Bron-Kerbosch algorithm [16]. In this project, a MATLAB toolbox [17] based on Bron-Kerbosch algorithm has been used to identify maximal cliques. Table 1 presents the size of the largest maximal cliques in test instances. We may observe that all grids have largest maximal cliques with size 3 or less, except IEEE 118-bus system. This system has a maximal clique of size 4.

After identifying the maximal cliques in a power grid, the next step is to identify chordless cycles.

3.2. Minimal Cycles in a Cycle Basis

Cycle basis identification algorithm in [18] is used to identify the cycle basis. A related MATLAB toolbox is also available [19]. The procedure of cycle basis identification is to first build a spanning tree. The edges that are not in the spanning tree are identified as the back edges. The number of back edges is the number of the cycles in a cycle basis. The back edges are added back to the spanning tree one by one. If a back edge is added, one cycle is identified. The resulting cycles are not necessarily minimal cycles. In this project, we aim to find minimal cycles. The justification of minimal cycles is given by the following example.

Fig. 1 presents an example graph to illustrate the chordal graph construction and why minimal chordless cycles are desired. Fig. 1(a) presents the original topology of a graph. The definition of a chordal graph is that all cycles of four or more vertices have a chord. This original graph is not a chordal graph since there is no chord for cycle $\{2, 3, 4, 5\}$.

As cycle basis identification algorithm does not guarantee minimal chordless cycles, we may end up with two cycles identified for Fig. 1(a): $\{1, 2, 5\}$ and $\{1, 2, 3, 4, 5\}$. Suppose that for the second cycle identified, two lines: $1 - 4$ and $1 - 5$ are added. The resulting graph shown in Fig. 1(b) is not a chordal graph since there is no chord in cycle $\{2, 3, 4, 5\}$.

On the other hand, if we are able to identify the two minimal chordless cycles as $\{1, 2, 5\}$ and $\{2, 3, 4, 5\}$, we may add a chord in the 4-node cycle (line $2 - 4$ or line $3 - 5$). The resulting graphs shown in Fig. 1(c)(d) are two chordal graphs.

To find minimal chordless cycle in a cycle basis, we use shortest path algorithm. For every back edge, first, the entire graph will have this back edge removed. The two nodes of the back edge are defined as the start node and the destination node. Shortest path in the modified graph from the start node to the destination node can be found using MATLAB 2017's graph toolbox.

3.3. Chordal Extension

A further graph decomposition strategy is employed to have virtual lines added and have any minimal chordless cycles with size greater than 3 to be decomposed into cycles with 3 nodes. The number of virtual lines added is $(n-3)$ where n is the size.

The graph after chordal extension is not necessarily a chordal graph. Though these cycles can be extended into chordal graphs, the entire grid may still have other cycles with size greater than 3. If a graph is chordal, then there exists a permutation to make the Cholesky factorization with zero filling. On the other hand, if there exists Cholesky factorization with non-zero filling, then the graph is not a chordal graph. In this project, we adopt MATPOWER's SDP toolbox [20] function `maxcardsearch` written by Dan Molzahn to conduct the check.

IEEE 14-bus system and IEEE 30-bus system are used as two examples in Fig. 2 to demonstrate minimal cycles and 3-node decomposition by adding virtual lines (dotted lines). If the resulting graph after 3-node decomposition is not a chordal graph, Cholesky factorization is then conducted. The additional virtual lines will be added as solid magenta lines. We can see that the 14-bus system after 3-node decomposition is a chordal graph while the 30-bus system after 3-node decomposition is not a chordal graph. Additional lines should be added to achieve a chordal graph.

Two cycles with size greater than 4 are identified for the 14-bus system. They are $\{4,5,6,13,14,9\}$ and $\{4,5,6,11,10,9\}$. Node 6 is used as the starting node to add virtual lines for both cycles. Total there are 4 virtual lines added to decompose the two cycles into 3-node cycles. The resulting graph is a chordal graph.

Four cycles with size greater than 4 are identified for the 30-bus system. They are $\{16,12,4,6,10,17\}$, $\{25,27,28,6,10,22,24\}$, $\{18,15,12,4,6,10,20,19\}$, and $\{23,15,12,4,6,10,22,24\}$. For the first cycle, 3 virtual lines are added starting from node 16: $16-4$, $16-6$, and $16-10$. Similarly, virtual lines are also added. The resulting graph, however, is not a choral graph. Cholesky factorization is then conducted and 5 virtual lines are found: $25-23$, $6-12$, $16-23$, $16-18$, $23-18$.

We have also conducted chordal extension to make every minimal cycle a clique. The resulting graphs for systems with size more than 57 are found as not chordal graphs.

Remarks: This study answers a question naturally arise from the research on cycle-based SDP feasibility enforcement presented in [8] and [9]: Will a PSD solution be found if its SOCP solution's submatrices related to cycles in a cycle basis are PSD? We demonstrate that chordal relaxation for each cycle of a cycle basis cannot result in a chordal graph. Hence, there is no guarantee that the strengthened SOCP in [8] and [9] can lead to SDP solution eventually.

3.4. Proposed sparse convex relaxation Formulation

With no guarantee of a chordal graph, the 3-node decomposition leads to a sparse convex relaxation. The decision variables of the proposed convex relaxation include c_{ii} $(i \in \mathcal{B})$ and c_{ij}, s_{ij} $((i,j) \in \mathcal{L} \cup \mathcal{V})$. $\mathcal{V}$ notates that the set of virtual lines for 3-node cycles decomposition. Note that compared to SOCP formulation whose decision variables include c_{ij} and s_{ij} for every line, the proposed convex relaxation has additional decision variables related to virtual lines.

Sparse matrix technique is employed in the proposed convex relaxation formulation. A sparse matrix X is defined to have its diagonal elements and elements related to lines and virtual lines non zero. The rest elements are all zeros.

$$X_{ii} = c_{ii}, \qquad i \in \mathcal{B} \tag{6a}$$

$$X_{ij} = c_{ij} - js_{ij}, \qquad (i,j) \in \mathcal{L} \cup \mathcal{V} \tag{6b}$$

$$X_{ji} = c_{ij} + js_{ij}, \qquad (i,j) \in \mathcal{L} \cup \mathcal{V} \tag{6c}$$

The sparse convex relaxation enforces all submatrices related to maximal cliques PSD. The maximal cliques include the maximal cliques with size greater than 2 from the original graph, 3-node cycles resulting from decomposition, and rest lines. The formulation is presented in (7).

$$\min \quad f(P_g) \tag{7a}$$

$$s.t. \quad P_i^g - P_i^d = \sum_{j \in \delta_i} (G_{ij} c_{ij} - B_{ij} s_{ij}), \quad i \in \mathcal{B} \tag{7b}$$

$$Q_i^g - Q_i^d = \sum_{j \in \delta_i} (-G_{ij} s_{ij} - B_{ij} c_{ij}), \quad i \in \mathcal{B} \tag{7c}$$

$$P_{ij} = g_{ij}(c_{ii} - c_{ij}) - b_{ij} s_{ij}, \quad (i,j) \in \mathcal{L} \tag{7d}$$

$$Q_{ij} = -b_{ij}(c_{ii} - c_{ij}) - g_{ij} s_{ij}, \quad (i,j) \in \mathcal{L} \tag{7e}$$

$$\sqrt{P_{ij}^2 + Q_{ij}^2} - S^{\max} \leq 0, \quad (i,j) \in \mathcal{L} \tag{7f}$$

$$(V_i^{\min})^2 \leq c_{ii} \leq (V_i^{\max})^2, \quad i \in \mathcal{B} \tag{7g}$$

$$P_g^{\min} \leq P_g \leq P_g^{\max}, \quad Q_g^{\min} \leq Q_g \leq Q_g^{\max} \tag{7h}$$

Constraints (6)

For all cliques MC,

$$\widetilde{X}^{(i)} \succeq 0, \quad i \in \mathcal{S}_{MC} \tag{7i}$$

where $\mathcal{S}_{MC}$ notates the set of maximal cliques and $\widetilde{X}^i$ notates the submatrix of X related to i^{th} maximal clique. $g_{ij} = \text{real}(y_{ij})$ and $b_{ij} = \text{imag}(y_{ij})$ and y_{ij} is a branch (between Bus i and Bus j)'s admittance. (7b) and (7c) represent the net power injection constraints at each bus. (7f) is the line flow limit constraint. (7g) is the voltage limit constraint. (7h) are the generator power limit constraints. (7i) enforces PSD for submatrices related to maximal cliques.

Remarks: (7) is a general SOCP/SDP AC OPF solver employing sparse matrix technique. If chordal extension of a power grid can result in a chordal graph, the above solver is a SDP solver. On the other hand, if the graph is not a chordal graph, the above solver is a stronger convex relaxation solver than SOCP. For comparison, we have employed Cholesky factorization method to find a chordal graph (Solver B in Section V).

4. Convex Iteration

4.1. Exactness based on 3-node cycles

The 3-node cycle decomposition makes computing more efficient. In this paper, we claim that if the submatrices related to the 3-node cycles inside each cycle in a cycle basis are PSD and rank-1, then the full matrix is an exact solution.

As the lower limit of bus voltage is larger than zero in general, the constraint (7g) can ensure X_{ii} is positive for any $i \in \mathcal{B}$. Thus, the sufficient and necessary condition for a solution from SOCP relaxation being feasible or exact is as follows [2].

$$\begin{vmatrix} X_{ii} & X_{ij} \\ X_{ji} & X_{jj} \end{vmatrix} = 0 \tag{8a}$$

$$\sum_{(i,j)\in c} \angle X_{ij} = 0, \quad c \in \mathcal{C} \tag{8b}$$

where $\mathcal{C}$ is the set of cycles in the power network.

(8a) guarantees that the submatrix related to two nodes i and j related to a line is rank-1. Besides (8a), (8b) guarantees the submatrix related to a cycle is PSD and rank-1.

For any chordless cycle of size n, we may add $(n-3)$ virtual lines to decompose the cycle into $(n-2)$ 3-node cycles. The cycle constraint (8b) can then be replaced by the cycle constraints related to every 3-node cycle.

Lemma 1: With every cycle in a cycle basis of a graph decomposed into 3-node cycles, if all submatrices corresponding to 3-node cycles are PSD and rank-1, the full matrix related to the entire graph is PSD and rank-1. □.

Theorem 1 For a 3×3 PSD matrix related to a 3-node cycle, given that all 2×2 submatrices related to lines are PSD and rank-1, then the 3×3 matrix is also rank-1.

Proof: Consider a Hermitian and PSD matrix X related to a 3-node cycle:

$$X = \begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}.$$

Since the three 2×2 principal submatrices of X related to three lines are PSD and rank-1, their determinants are 0.

$$\begin{vmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{vmatrix} = \begin{vmatrix} X_{22} & X_{23} \\ X_{32} & X_{33} \end{vmatrix} = \begin{vmatrix} X_{11} & X_{13} \\ X_{31} & X_{11} \end{vmatrix} = 0, \text{ or :} \tag{9}$$

$$X_{11}X_{22} = |X_{12}|^2,\ X_{22}X_{33} = |X_{23}|^2,\ X_{11}X_{33} = |X_{13}|^2$$

We will examine the determinant of X.

$$\begin{aligned} |X| &= X_{11}X_{22}X_{33} + X_{12}X_{23}X_{31} + X_{13}X_{21}X_{32} \\ &- |X_{13}|^2X_{22} - |X_{23}|^2X_{11} - |X_{12}|^2X_{13} \end{aligned}$$

Replacing $|X_{ij}|^2$ by $X_{ii}X_{jj}$ leads to:

$$\begin{aligned} |X| &= -2X_{11}X_{22}X_{33} + X_{12}X_{23}X_{31} + X_{13}X_{21}X_{32} \\ &= -2X_{11}X_{22}X_{33} + 2|X_{12}||X_{23}||X_{31}|\cos(\theta_{12} + \theta_{23} + \theta_{31}) \end{aligned}$$

where θ_{12}, θ_{23}, θ_{31} represent the angles of X_{12}, X_{23}, and X_{31}. Note that $X_{11}X_{22}X_{33} = |X_{12}||X_{23}||X_{31}|$ according to (9). Thus,

$$|X| = -2X_{11}X_{22}X_{33}(1 - \cos(\theta_{12} + \theta_{23} + \theta_{31})). \tag{10}$$

Since $\cos(\theta_{12}+\theta_{23}+\theta_{31}) \leq 1$, $|X| \leq 0$. On the other hand, X is PSD, hence $|X| \geq 0$. Therefore, $|X| = 0$. This means that the rank of X is less than 3.

The sum of angles is found to be 0 since $|X| = 0$ enforces the following constraint.

$$\cos(\theta_{12}+\theta_{23}+\theta_{31}) = 1 \Rightarrow \quad \theta_{12}+\theta_{23}+\theta_{31} = 0$$

This means that cycle $\{(1,2),(2,3),(3,1)\}$ satisfies the SOCP exactness condition (8b). Therefore, based on the sparse convex relaxation formulation in (7), to have an exact solution, we only need to enforce 2×2 submatrices corresponding to all lines rank-1. $\square$

We will implement this requirement in convex iteration.

4.2. Principle of convex iteration

Convex iteration has been applied to SDP OPF to achieve exact solutions in [11, 12]. We will briefly review convex iteration principle in this section.

For a $n \times n$ Hermitian PSD matrix, its trace equals the sum of all its eigenvalues.

$$\mathrm{Tr}(X) = \sum_{i=1}^{n} \lambda_i, \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0 \tag{11}$$

where λ_i are the eigenvalues of X; $\mathrm{Tr}(\cdot)$ is the "Trace" calculation. If X is rank-1, then all eigenvalues except λ_1 is zero. Thus:

$$\mathrm{Tr}(X) - \lambda_1 = 0. \tag{12}$$

The maximum eigenvalue λ_1 can be obtained through the following equation [21]:

$$\lambda_1 = \quad \mathrm{Tr}(X u_1 u_1^H) \tag{13}$$

where u_1 is the normalized eigenvector correspond to λ_1. Thus, combining (12) and (13) leads to:

$$\mathrm{Tr}(X(I - u_1 u_1^H)) = 0 \tag{14}$$

Define $W \triangleq I - u_1 u_1^H$. If $\mathrm{Tr}(XW) = 0$, then X is rank-1. Thus by adding $\mathrm{Tr}(XW)$ as a penalty term on the objective function of the SOCP formulation, we may enforce X to be rank-1.

Note that W is also a variable. This makes the problem a bilinear problem. To solve this bilinear problem, iterative approach can be implemented. Denote the problem including rank-1 penalty term as $F(X, W)$ whose objective function includes an additional term $\omega\,\mathrm{Tr}(XW)$ (ω is the penalty factor). We may fix W to solve a convex problem and find X. Then for the given X, we may find W. Below is the iterative procedure:

$$\text{problem 1:}\quad F(X, W^*) \rightleftharpoons \text{problem 2:}\quad F(X^*, W)$$

where W^* is the solution of the problem 2; X^* is the solution of the problem 1. Consider our derivation from (11) to (14), it implies that for a fixed X^*, W can be found through eigenvalue-based decomposition of X^*.

$$X^* = U\Lambda U^H \tag{15}$$

where Λ is a diagonal matrix with its elements eigenvalues, U is a unitary matrix, and its columns are the eigenvectors of X^*, i.e., $U = [u_1, u_2, \ldots, u_n]$. Thus, problem 2 for iteration procedure is equivalent to:

$$W = UU^H - u_1 u_1^H, \quad \Rightarrow W = U(:, 2:n)U(:, 2:n)^H \tag{16}$$

4.3. Sparse implementation

Further, we seek sparse matrix-based implementation. The sparse convex relaxation solver does not give the full matrix X. According to Theorem 1, for the 3-node cycle-based convex relaxation, we only need to enforce all 2×2 submatrices related to lines and virtual lines rank-1 to achieve the exactness. Therefore, the rank penalty term $\mathrm{Tr}(XW)$ can be replaced by:

$$\sum_{i \in \mathcal{AL}} \mathrm{Tr}(\widehat{X}^{(i)}\widehat{W}^{(i)}) \tag{17}$$

where $\mathcal{AL}$ is the set of all lines, including original lines and virtue lines, $\widehat{X}^i$ is the i^{th} 2×2 submatrix and $\widehat{W}^i$ can be found based on $\widehat{X}^i$ using (16).

During the iterative procedure, some $\mathrm{Tr}(\widehat{X}^{(k)}\widehat{W}^{(k)})$ term might not keep monotonic decreasing. This may lead to the increase of the ranks of those submatrices. To keep the submatrices rank-1 once they have reached rank-1 during iteration, we implement the following constraints to problem 1:

$$\mathrm{Tr}(\widehat{X}^{(i)}\widehat{W}^{(i)}) \le \epsilon \quad i \in \mathcal{DL} \tag{18}$$

where ϵ is tolerance, $\mathcal{DL}$ is the set for all submatrices that have achieved rank-1. The equivalent form of this constraint has been adopted in [12].

Our experiments show that this constraint is capable to decrease the iterations for large size cases. E.g. for nesta_case1354_pegase and $\omega = 1000$, with constraint (18), convex iteration can converge at 3 steps; without constraint (18), convex iteration can not converge after 10 steps. The formulation of 3-node based convex iteration is as follows.

problem 1:

$$\min \quad f(P_g) + \omega \sum_{i \in \mathcal{AL}} \mathrm{Tr}(\widehat{X}^{(i)} \widehat{W}^{*(i)}) \tag{19}$$

$$s.t. \quad \text{Constraints (6)} \quad \text{(7b)} \sim \text{(7h)}$$

$$\mathrm{Tr}(\widehat{X}^{(i)} \widehat{W}^{*(i)}) \leq \epsilon \quad i \in \mathcal{DL}$$

$$\text{For all cliques } MC$$

$$\widetilde{X}^{(i)} \succeq 0 \quad i \in \mathcal{S}_{MC}$$

problem 2:

$$\widehat{W}^{*(i)} = U^{(i)}(:,2)U^{(i)}(:,2)^H \quad i \in \mathcal{AL} \tag{20}$$

where $U^{(i)}$ is obtained through the eigenvalue decomposition of $\widehat{X}^{*(i)}$: $\widehat{X}^{*(i)} = U^{(i)}\Lambda^{(i)}U^{(i)H}$.

The initial values of the iteration can be provided by the solution of Formulation (7).

5. Numerical Examples

Instances from the NICTA test archive [22] are tested using the proposed formulation. Additional instances with large gaps (case9mod, case39mod1, case300mod) are from [23]. We also implemented the method in [3] and [4] and developed a sparse SDP solver based on a chordal graph using Cholesky factorization. Cholesky factorization of a Hermitian semidefinite matrix A is defined as follows. $P_\sigma A P_\sigma^T = LL^T$, where P_σ is a permutation of the elements in A; L is a lower triangular matrix which is called Cholesky factor of A. The sparsity pattern for the chordal extension of A is the same with $L + L^T$ [3]. Moreover, to obtain minimal number of virtual lines, P_σ will permute the indexes of A based on the minimal degree ordering. Using Cholesy factorization, virtual lines of a power grid graph are found and added to achieve a chordal graph. Maximal cliques of the chordal graph are then found and the submatrices related to the maximal cliques are enforced to be PSD.

The cases were first solved by MATPOWER [20] to obtain feasible solutions as upper bounds. In addition, the cases were solved by the SDP solver developed by Lavaei's group [7] (**Solver A**), the sparse SDP solver based on Cholesky factorization (**Solver B**), and the proposed solver (**Solver C**).

We compare the computing time and solutions of the three solvers to demonstrate that (i) the gaps from the proposed solver is as tight as those from other sparse SDP solvers; (ii) the computing efficiency is higher compared with the two sparse SDP solvers. The number of virtual lines required for the three solvers, sizes of maximal cliques, and ranks of submatrices generated are all compared. To show the proposed relaxation solver is stronger than the SOCP solver, we compared the optimality gap of the proposed relaxation with one strengthen SOCP solver [9]. Finally, we select a few instances with nonzero gaps to demonstrate that convex iteration based on 3-node cycles can give rank-1 PSD solutions in those instances.

In all tests, the gap is defined as: $Gap = \frac{UB-LB}{UB} \times 100\%$, where UB is the upper bound which is calculated through MATPOWER; LB is the lower bound of the objective value. In Table 3 and 4, LB is calculated by the relaxation solvers; in Table 5, LB is calculated by the convex iteration solver.

5.1. Proposed Convex Relaxation

Our numerical experiments are conducted on an Intel(R) Xeon(R) CPU E5-2698 v3 @ 2.30 GHZ (2 processors) computer. All solvers are implemented in MATLAB 2017a-based CVX platform [24]. Mosek 7.1.0.12 solver is invoked. To achieve the balance between the stability and accuracy, we adopt the Mosek setting of Solver A(tuned by Lavaei's group). Although this setting may decrease the accuracy of the solution for large size cases, it provides the best stability for the Mosek solver (Mosek with default setting may fail to solve the cases which are larger than 2736 buses). In Table 4, we compared the proposed relaxation solver with the strengthen SOCP solver [9]. Because the test cases of reference [9] also comes from the NICTA test archive, we cited the results of [9] in Table 4. The numerical results from two SDP solvers and the proposed solver are listed in Table 3. In Table 3, columns A, B, C represent the three solvers; Max_cliqueSize is the size of the largest clique; Max_Rank means the maximum rank of submatrices; Solver Time is the optimizer terminate time of Mosek; N_vline means the numbers of the virtual lines; Decomp_Time is time cost on the cliques decomposition.

According to Table 3, for small and median size cases from nesta_case3_lmbd to nesta_case300_ieee, Solver C obtains the same results as both or one of the two SDP solvers. For large size cases, Solver C has a gap slightly larger but a much higher computing efficiency. This is due to the following two facts. (i) The proposed method deals only 3-node cycles while the two SDP solvers deal with cliques with larger sizes. For example, for case nesta_case3120sp_mp, the size of the cliques reach 29 and 27 for Solver A and Solver B.

(ii) On the other hand, the proposed method adds less virtual lines compared to SDP solver B. For case nesta_case3120sp_mp, the number of virtual lines is 4527 for Solver B while it is 3153 for Solver C.

The proposed relaxation solver is compared with the strengthen SOCP method [9] in Table 4. The table shows the optimiality gaps for two solvers. We can see performance of the proposed relaxation solver is better than the strengthen SOCP solver.

We note that in Table 3, there are some cases showing different relaxation gaps between two SDP solvers, and Solver C showing tighter gaps than one of the SDP solvers. According [2], since both sparse SDP solver A and B are based on chordal graphs, their solutions are SDP OPF solutions and should be the same. Moreover, Solver C should have gaps greater than or equal to those from SDP. In our experiments, the reason of the numerical inconsistency is due to the configuration of Mosek. We tested some cases by CVX with SDPT3 in default setting, and listed the results in the Table 2. The results show SDP solver A and B, and solver C achieve the same gaps. However, as SDPT3 is much slower than Mosek, we use Mosek for all case studies.

Remarks: The proposed convex relaxation solver achieves almost the same tightness of SDP solvers with a much higher computing efficiency. The computing time decreases at least 27% with an average of 49%. Our method solves the dilemma mentioned in [2] that decreasing the size of submatrices results in increased virtual lines for sparse SDP. The proposed sparse convex relaxation solver can achieve almost the same tightness of SDP solvers with much higher computing efficiency.

5.2. Rank-1 solution through convex iteration

We tested 3-node cycle-based convex iteration method on several non-zero gap cases. In the experiments, we keep $\epsilon = 10^{-5}$ used in (18) for all cases. The results are listed in Table 5. Column Nlines is the sum of the numbers of lines and virtual lines in a graph; Niter is the number of iterations. In this Table, we apply maximum active and reactive power mismatch to show the feasibility of the solution, where superscript before means before convex iteration, and after means after convex iteration. The mismatches are calculated through the power balancing equations using the voltage phasor vector recovered from the solution of a sparse matrix X, where superscript sol means the solutions directly from Solver C with or without convex iteration; superscript rec means the solutions obtained through the recovered voltage phasors after the solutions from the proposed solver. V^{rec} and θ^{rec} are the magnitude and angle of the recovered voltage phasor vector.

The voltage vector recovering method in [2] is adopted in this project. First, we define the phase angle of the voltage phasor at the reference node as $0°$; next, we identify the spanning tree of the power network, and define the unique path from the reference node to the node i as $\mathcal{P}_i$; then the voltage phasor at any node

i can be recovered through the following equations.

$$V_i^{\text{rec}} = \sqrt{c_{ii}}, \quad \theta_i^{\text{rec}} = -\sum_{(j,k)\in\mathcal{P}_i} \angle(c_{jk} - js_{jk})$$

The results in Table 5 show that the proposed convex iteration is capable of decreasing maximum rank of submatrices. In Fig.3 we show that the rank error represented by $\sum \mathrm{Tr}(\widehat{X}^{(k)}\widehat{W}^{(k)})$ decreasing for two instances. According to Table 5, all 9 instances successfully achieve rank-1 , the power mismatches are close to 0, and gaps are non-negative. It means the recovered voltage vectors from the convex iteration solutions are feasible for the original AC OPF, and the objective value is not worse than the MATPOWER. We noted in Table 5, the gap for case9mod case is 22.89% while MaxRank is one. The reason of this situation is that the case9mod case has multi-local optimal solutions [23]. The MATPOWER which is based on the interior point method obtained one of the local optimal solution, while the convex iteration solver obtained another optimal solution which is closer to the global optimal.

6. Conclusion

In this paper, we proposed a 3-node cycle decomposition based sparse convex relaxation for AC OPF. We have shown that the 3-node cycle decomposition can not guarantee that the resulting graph is a chordal graph. However, the proposed relaxation can achieve the close tightness as SDP OPF solvers. On the other hand, our method has a clearly higher computing efficiency. In addition, we also investigated a practical application of the relaxed solution: finding a feasible AC OPF solution that is better than the one obtained from MATPOWER flat start. An efficient convex iteration implementation is investigated for the proposed sparse convex solver to achieve exactness or rank-1 solutions. Our experiment results show the feasibility of the implementation. A special case (case9mod) is used to demonstrate that the feasible solution found from convex iteration is better than the one obtained from MATPOWER flat start. Finding feasible solutions better than MATPOWER flat start is indeed of practical interest. Many research works have been carried out in this category. One other example is our work [25], which finds a local solution using interior point algorithm based on a starting point obtained from the proposed convex relaxation solver.

Reference

[1] J. Lavaei, S. H. Low, Zero duality gap in optimal power flow problem, IEEE Transactions on Power Systems 27 (1) (2012) 92–107.

[2] S. H. Low, Convex relaxation of optimal power flow part i: Formulations and equivalence, IEEE Transactions on Control of Network Systems 1 (1) (2014) 15–27.

[3] R. A. Jabr, Exploiting sparsity in SDP relaxations of the OPF problem, IEEE Transactions on Power Systems 27 (2) (2012) 1138–1139.

[4] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, C. L. DeMarco, Implementation of a large-scale optimal power flow solver based on semidefinite programming, IEEE Transactions on Power Systems 28 (4) (2013) 3987–3998.

[5] R. Madani, M. Ashraphijuo, J. Lavaei, Promises of conic relaxation for contingency-constrained optimal power flow problem, IEEE Transactions on Power Systems 31 (2) (2016) 1297–1307.

[6] M. Fukuda, M. Kojima, K. Murota, K. Nakata, Exploiting sparsity in semidefinite programming via matrix completion i: General framework, SIAM Journal on Optimization 11 (3) (2001) 647–674.

[7] R. Madani, M. Ashraphijuo, J. Lavaei, SDP Solver of Optimal Power Flow User's Manual (2014).
URL `http://ieor.berkeley.edu/~lavaei/Software.html`

[8] B. Kocuk, S. S. Dey, X. A. Sun, Strong socp relaxations for the optimal power flow problem, Operations Research 64 (6) (2016) 1177–1196.

[9] Z. Miao, L. Fan, H. G. Aghamolki, B. Zeng, Least Squares Estimation Based SDP Cuts for SOCP Relaxation of AC OPF, IEEE Transactions on Automatic Control 63 (1) (2018) 241–248. `doi:10.1109/TAC.2017.2719607`.

[10] C. Bingane, M. F. Anjos, S. Le Digabel, Tight-and-cheap conic relaxation for the ac optimal power flow problem, IEEE Transactions on Power Systems 33 (6) (2018) 7181–7188.

[11] W. Wang, N. Yu, Chordal conversion based convex iteration algorithm for three-phase optimal power flow problems, IEEE Transactions on Power Systems 33 (2) (2018) 1603–1613.

[12] Y. Shi, H. Tuan, P. Apkarian, A. Savkin, Global optimal power flow over large-scale power transmission networks, Systems & Control Letters 118 (2018) 16–21.

[13] S. Bose, S. H. Low, T. Teeraratkul, B. Hassibi, Equivalent relaxations of optimal power flow, IEEE Transactions on Automatic Control 60 (3) (2015) 729–742.

[14] R. A. Jabr, Radial distribution load flow using conic programming, IEEE Transactions on Power Systems 21 (3) (2006) 1458–1459.

[15] X. Bai, H. Wei, K. Fujisawa, Y. Wang, Semidefinite programming for optimal power flow problems, International Journal of Electrical Power & Energy Systems 30 (6) (2008) 383–392.

[16] H. Johnston, Cliques of a graph-variations on the Bron-Kerbosch algorithm, International Journal of Parallel Programming 5 (3) (1976) 209–238.

[17] J. Wildman, Bron-Kerbosch maximal clique finding algorithm, `https://www.mathworks.com/matlabcentral/fileexchange/30413-bron-kerbosch-maximal-clique-finding-algorithm` (2011).

[18] K. Mehlhorn, D. Michail, Implementing minimum cycle basis algorithms, Journal of Experimental Algorithmics (JEA) 11 (2007) 2–5.

[19] S. Iglin, grTheory - Graph Theory Toolbox, `https://www.mathworks.com/matlabcentral/fileexchange/4266-grtheory-graph-theory-toolbox?focused=5177310&tab=function` (2011).

[20] R. D. Zimmerman, C. E. Murillo-Sánchez, R. J. Thomas, Matpower: Steady-state operations, planning, and analysis tools for power systems research and education, IEEE Transactions on Power Systems 26 (1) (2011) 12–19.

[21] J. Dattorro, Convex optimization & Euclidean distance geometry, Lulu. com, 2010.

[22] C. Coffrin, D. Gordon, P. Scott, NESTA, the NICTA energy system test case archive (2014).
URL `http://arxiv.org/abs/1411.0359`

[23] W. A. Bukhsh, A. Grothey, K. I. McKinnon, P. A. Trodden, Local solutions of the optimal power flow problem, IEEE Transactions on Power Systems 28 (4) (2013) 4780–4788.

[24] M. Grant, S. Boyd, Y. Ye, CVX: Matlab software for disciplined convex programming (2008).

[25] M. Ma, L. Fan, Rank-1 positive semidefinite matrix-based nonlinear programming formulation for ac opf, International Transactions on Electrical Energy Systems 29 (10).

List of Figures

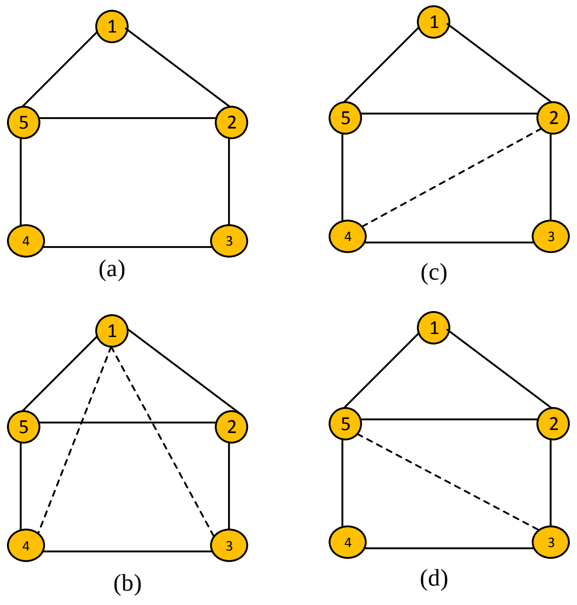

Figure 1: Chordal graph construction explanation.

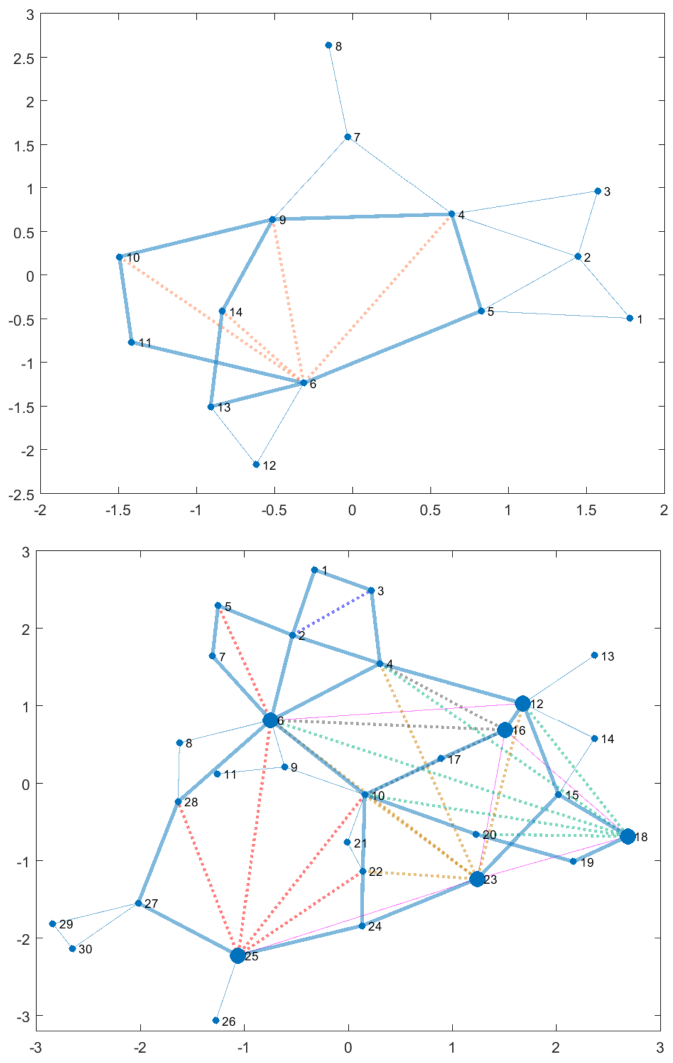

Figure 2: Topologies of IEEE 14-bus case and IEEE 30-bus. Blue solid lines represent the lines of the power grid. Highlighted blue lines notate the edges related to cycles of more than 3 nodes. Dotted lines are the virtual lines added to decompose a cycle into 3-node cycles. The solid magenta lines in the 30-bus case notates additional lines added to make the graph chordal.

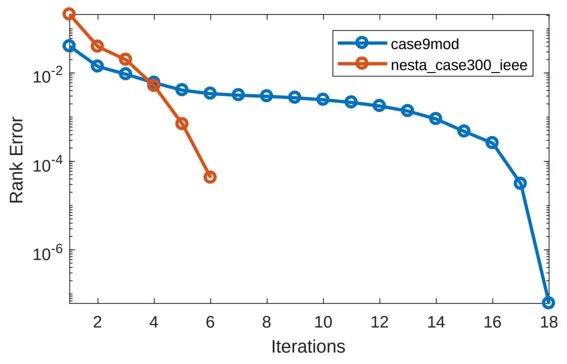

Figure 3: Rank error converging for two instances.

List of Tables

Table 1: Size of the largest maximal cliques

Test case	size	Test case	size
nesta_case3_lmbd	3	nesta_case4_gs	2
nesta_case5_pjm	3	nesta_case14_ieee	3
nesta_case30_ieee	3	nesta_case57_ieee	3
nesta_case118_ieee	4	nesta_case300_ieee	3
nesta_case1354_pegase	3	nesta_case2383wp	3
nesta_case2736sp_mp	3	nesta_case2737sop_mp	3
nesta_case2746wop_mp	3	nesta_case2746wp_mp	3
nesta_case3012wp_mp	3	nesta_case3120sp_mp	3

Table 2: SDPT3 results

Case	UB	Gap		
		A	B	C
nesta_case300_ieee	16891.28	0.08%	0.08%	0.08 %
nesta_case1354_pegase	74064.11	0.01%	0.01%	0.01 %
nesta_case2736sp_mp	1307961.70	0.00%	0.00%	0.00%
nesta_case2737sop_mp	777668.88	0.00%	0.00%	0.00%

Table 3: Results comparison

Case	UB	Gap(%)			Solver Time			Max_cliqueSize			Max_Rank			N_vline		Decomp_Time	
		A	B	C	A	B	C	A	B	C	A	B	C	B	C	B	C
nesta_case3_lmbd	5812.64	0.41	0.39	0.39	0.58	**0.30**	0.48	3	3	3	2	2	2	0	0	0.02	0.02
nesta_case4_gs	156.43	0.00	0.00	0.00	0.56	0.39	**0.39**	3	3	3	1	1	1	1	1	0.01	0.03
nesta_case5_pjm	17551.89	5.22	5.23	5.22	0.90	**0.37**	0.45	3	3	3	2	2	2	1	1	0.02	0.02
nesta_case14_ieee	244.05	0.00	0.00	0.00	0.59	**0.42**	0.51	3	3	3	1	1	2	4	4	0.01	0.02
nesta_case30_ieee	204.97	0.00	0.00	0.00	0.58	0.42	**0.41**	4	4	3	1	1	1	14	14	0.03	0.03
nesta_case57_ieee	1143.28	0.00	0.00	0.00	0.81	0.95	**0.59**	6	6	3	2	1	1	59	55	0.05	0.06
nesta_case118_ieee	3689.92	0.07	0.07	0.09	1.34	1.78	**1.34**	5	5	4	2	2	3	87	73	0.12	0.33
nesta_case300_ieee	16891.28	0.08	0.08	0.09	6.92	4.93	**3.51**	7	7	3	2	2	3	250	193	0.55	0.55
nesta_case1354_pegase	74064.11	0.56	0.50	1.20	26.42	20.10	**11.53**	13	13	3	6	6	3	1020	698	1.13	3.79
nesta_case2383wp_mp	1870807.81	0.96	1.38	1.02	100.04	86.92	**35.05**	24	25	3	6	6	3	3269	2225	4.23	7.55
nesta_case2736sp_mp	1307961.70	28.01	27.77	27.94	36.53	37.00	**11.54**	24	25	3	6	6	3	3878	2810	5.79	8.26
nesta_case2737sop_mp	777668.88	11.84	11.37	11.37	23.40	25.07	**18.21**	24	24	3	6	6	3	3853	2814	5.91	8.22
nesta_case2746wop_mp	1208281.08	15.42	15.44	15.68	46.92	31.98	**11.15**	24	26	3	6	6	3	4103	2819	6.46	9.33
nesta_case2746wp_mp	1631868.17	28.89	28.92	29.37	47.82	23.07	**9.91**	25	26	3	6	6	3	3973	2800	6.04	8.57
nesta_case3012wp_mp	2600842.77	0.23	0.27	0.80	124.16	115.29	**73.40**	26	28	3	6	6	3	4407	3065	7.51	9.58
nesta_case3120sp_mp	2145965.33	0.33	0.86	0.46	172.45	118.98	**81.20**	29	27	3	6	6	3	4527	3153	7.91	9.74
nesta_case30_fsr_api	372.14	11.08	11.09	11.62	0.53	0.52	**0.11**	4	4	3	2	2	2	14	14	0.02	0.02
nesta_case118_ieee_api	6383.57	5.28	5.29	5.56	1.54	1.23	**0.06**	5	5	4	2	2	3	87	73	0.04	0.34
case9mod.m	4267.07	35.48	35.48	35.48	0.48	**0.45**	0.48	3	3	3	2	2	2	3	3	0.01	0.01
case39mod1	11221.00	3.72	3.72	3.72	0.45	0.53	**0.04**	4	4	3	2	2	2	21	21	0.02	0.06
case300mod	378540.50	0.14	0.14	0.14	5.27	4.37	**0.06**	7	7	3	3	3	3	250	193	0.12	0.66

Table 4: Comparison with one strengthened SOCP solver

Case	UB	Gap(%)	
		Proposed Solver	SOCP_SDP[9]
nesta_case3_lmbd	5812.64	0.39	1.27
nesta_case4_gs	156.43	0.00	0.00
nesta_case5_pjm	17551.89	5.22	9.08
nesta_case14_ieee	244.05	0.00	0.00
nesta_case30_ieee	204.97	0.00	0.29
nesta_case57_ieee	1143.28	0.00	0.00
nesta_case118_ieee	3689.92	0.09	1.51
nesta_case300_ieee	16891.28	0.09	0.64

Table 5: Convex iteration results

case	UB	gap(%)	Nlines	Niter	MaxRank	$P_{\text{mis}}^{\text{before}}$	$Q_{\text{mis}}^{\text{before}}$	$P_{\text{mis}}^{\text{after}}$	$Q_{\text{mis}}^{\text{after}}$	ω
nesta_case5_pjm	17551.89	**0.00**	7	2	1	0.57	2.17	**6.27e-6**	**1.46e-5**	28000
nesta_case30_fsr_api	372.14	**0.08**	55	5	1	6.89e-2	3.71e-2	**3.81e-5**	**5.54e-5**	1210
nesta_case118_ieee_api	6383.57	**0.00**	252	6	1	9.26	4.12	**4.13e-4**	**5.25e-4**	560
nesta_case118_ieee	3689.92	**0.00**	252	4	1	5.80e-1	6.55e-1	**2.36e-4**	**1.90e-4**	22
nesta_case300_ieee	16891.28	**0.00**	602	6	1	8.12e-1	4.64e-1	**1.45e-4**	**1.87e-4**	1000
nesta_case1354_pegase	74064.11	**0.00**	2408	3	1	9.73e-1	6.61e-1	**5.27e-4**	**2.05e-3**	821.5
case9mod	4267.07	**22.89**	12	18	1	7.49e-3	2.59e-1	**2.66e-9**	**1.03e-7**	1e6
case39mod1	11221.00	**0.00**	67	19	1	9.92e-2	1.73	**1.38e-6**	**1.53e-5**	1.98e5
case300mod	378540.50	**0.00**	602	21	1	7.13	1.79	**5.95e-5**	**5.71e-4**	3.91e4