A Clique Merging Algorithm to Solve Semidefinite Relaxations of Optimal Power Flow Problems *

Julie Sliwak[1,4], Erling D. Andersen[2], Miguel F. Anjos[3] , Lucas Létocart[4], Emiliano Traversi[4]

[1] RTE & Polytechnique Montréal, Canada
julie.sliwak@polymtl.ca
[2] MOSEK
e.d.andersen@mosek.com
[3] University of Edinburgh, United Kingdom
anjos@stanfordalumni.org
[4] LIPN, UMR CNRS 7030, Université Sorbonne Paris Nord, France
{lucas.letocart,emiliano.traversi}@lipn.univ-paris13.fr

Mots-clés : *Clique Decomposition, Optimal Power Flow, Semidefinite Programming.*

1 Introduction

The Alternating Current Optimal Power Flow (ACOPF) problem is a famous problem in power systems that is highly nonconvex. Although there is still no efficient method to solve this problem to guaranteed global optimality for realistic systems, semidefinite programming (SDP) is one of the most promising approaches to achieve this. SDP relaxations of large-scale problems are computationally demanding and clique decomposition is a well-known technique to speed up their solution by exploiting their structure. Because there are many different clique decompositions for the same SDP problem and they are not equivalent in terms of computation time, it remains an open question to know how to choose a clique decomposition for a given application of SDP such as ACOPF. Our main contribution is a new strategy to compute efficient clique decompositions with a clique merging heuristic. This heuristic is based on two different estimates of the computational burden of a SDP problem : the size of the problem and an estimation of a per-iteration cost for a state-of-the-art interior-point algorithm.

2 Rank relaxation and standard resolution

ACOPF can be formulated as a nonconvex quadratically constrained quadratic problem in complex numbers. The standard SDP relaxation for such a problem is obtained by dropping the rank constraint after introducing the complex matrix $W = vv^H$, equivalent to $W \succeq 0$ and $rank(W) = 1$:

$$\left\{ \begin{array}{ll} \min & v^H Q_0 v \\ \textbf{s.t.} & v^H Q_p v \leq a_p \ \forall p \\ & v \in \mathbb{C}^n \end{array} \right. \Rightarrow \left\{ \begin{array}{ll} \min & Q_0 \cdot W \\ \textbf{s.t.} & Q_p \cdot W \leq a_p \ \forall p \\ & W \succeq 0 \end{array} \right. \quad (1)$$

The SDP problem can be reformulated using clique decomposition as introduced by Fukuda et al. [1] resulting in a problem with several positive semidefinite constraints on small submatrices plus the required linking constraints between these submatrices. The only requirement to apply clique decomposition is the chordality of the aggregate sparsity pattern of the graph G^A representing the nonzeros entries in the SDP data. A graph is chordal if every cycle of length 4 or more has a chord, i.e., an edge joining nonconsecutive vertices. If G^A is not chordal, a

*This work has been published in [3].

chordal extension H has to be computed, usually with a minimal number of additional edges. As finding such a minimal chordal extension is NP-complete, several efficient heuristics are available in the literature. The maximal cliques in H, denoted by $L = \{C_1, ..., C_r\}$, define the submatrices that must be positive semidefinite in the reformulated SDP problem. A clique is a subset of vertices such that every two distinct vertices in the clique are connected. A clique is maximal if it is not a subset of another clique. Linking constraints between the cliques are specified by a clique tree, i.e., a maximum-weight spanning tree for the clique graph W. The nodes of W are the cliques in L and there is a weighted edge between each pair of vertices that share nodes, its weight being the number of shared nodes. It turns out that the computation time is highly sensitive to the choice of the chordal extension heuristic and minimizing the number of additional edges turns out to be inefficient for ACOPF problems.

3 A new clique merging algorithm

For a given decomposition, let L be the set of maximal cliques, m the number of linear constraints in the original SDP problem, and ℓ the number of linking constraints coming from the clique tree T. The cardinality of the clique C_i is denoted by $|C_i|$. The algorithm proposed in [2] seeks to greedily minimize an estimate of the computational burden of a SDP problem $f_M(L, \ell) : f_M(L, \ell) = m + \ell + \sum_{C_i \in L} |C_i|(|C_i| + 1)/2$ where ℓ depends on the clique tree $T : \ell = \sum_{(C_i, C_j) \in E} d_{ij}(2d_{ij} + 1)$ with d_{ij} the number of complex variables shared by C_i and C_j. Minimizing the function $f_M(L, \ell)$ leads to the merging of small cliques. Another reasonable choice is to minimize the cost of an interior-point iteration : $f(L, \ell) = \alpha \sum_{C_i \in L} |C_i|^3 + \beta(m + \ell)^3 + c$, where α, β and c are parameters determined using a multilinear regression. This function is a better estimate of the computational burden but it leads to the merging of large cliques, which is interesting only if we merge a few cliques. We propose to repeatedly merge the two cliques that minimize either our criterion f or the criterion f_M as long as the number of cliques is greater than 10% of the number of buses. This value of 10% was recommended in [2]. Note that we compute both clique decompositions for the complex SDP relaxation. Before the clique merging heuristic, we use a Cholesky factorization with an approximate minimum degree ordering to compute a chordal extension and Prim's algorithm for the clique tree.

4 Conclusion and Future Research

Using this strategy with a suitable choice of the switching parameter k_{max}, we obtained a reduction in total computation time of 12% on standard MATPOWER instances for ACOPF. This strategy could be extended for generic sparse SDP problems for which the decomposition contains lots of small cliques. Future work will consider a chordal extension heuristic that provides clique decompositions similar to those obtained by clique merging.

Références

[1] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion i : General framework," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 647–674, 2001.

[2] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, and C. L. DeMarco, "Implementation of a large-scale optimal power flow solver based on semidefinite programming," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 3987–3998, 2013.

[3] J. Sliwak, E. Andersen, M.F. Anjos, L. Létocart, and E. Traversi, "A Clique Merging Algorithm to Solve Semidenite Relaxations of Optimal Power Flow Problems," *IEEE Transactions on Power Systems*, vol. 36, no. 2, pp. 1641–1644, 2021.