Efficient AC Optimal Power Flow & Global Optimizer Solutions

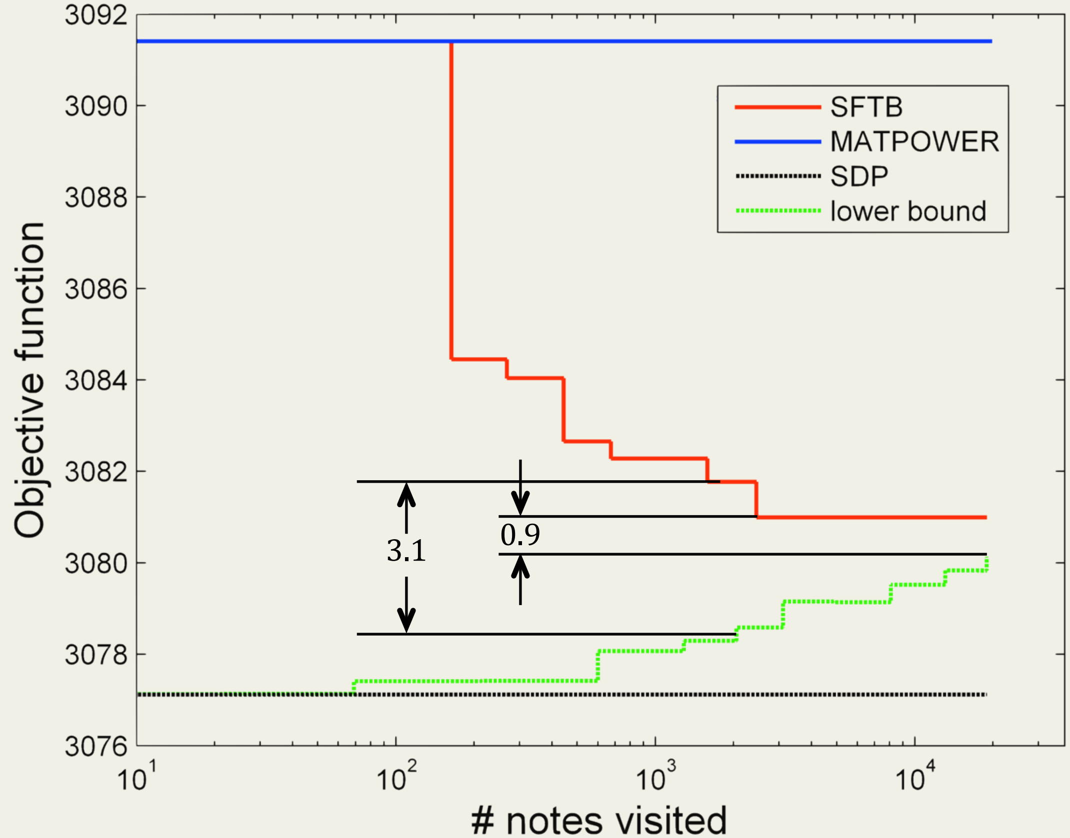
Zhe Hu, Göktürk Poyrazoğlu, and HyungSeon Oh
University at Buffalo
CERTS Review meeting, 8/5/2015

Content

- Presentation at 2014
- Questions Raised 2014
- Major Modifications I – Clique Decomposition & Merging
- Major Modifications II – Angle Cut
- Major Modifications III – Parallelization
- Results
- Certificate System to Guarantee the Global Optimizer
- Conclusions

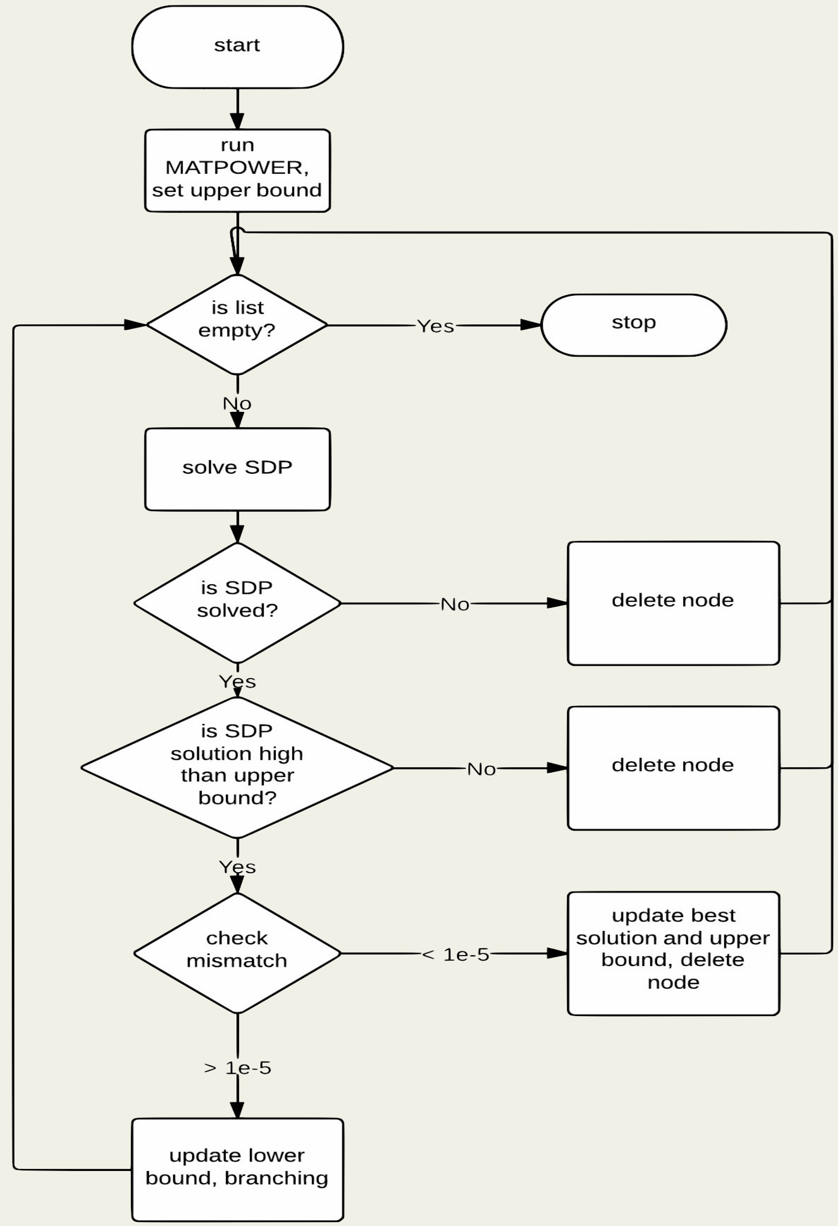
Presentation at 2014

- Optimal power flow
  - Nonlinear and nonconvex → difficult to solve
    - No guarantee to find a solution
    - Heuristic search aiming for a local solution
- Global solution
  - MATPOWER does not find the global optimizer
  - Divide-and-conquer
  - Visit 20,000 nodes
  - Global optimizer within an epsilon gap →
- 2015 research goal
  - Efficient algorithm
  - Parallel computation

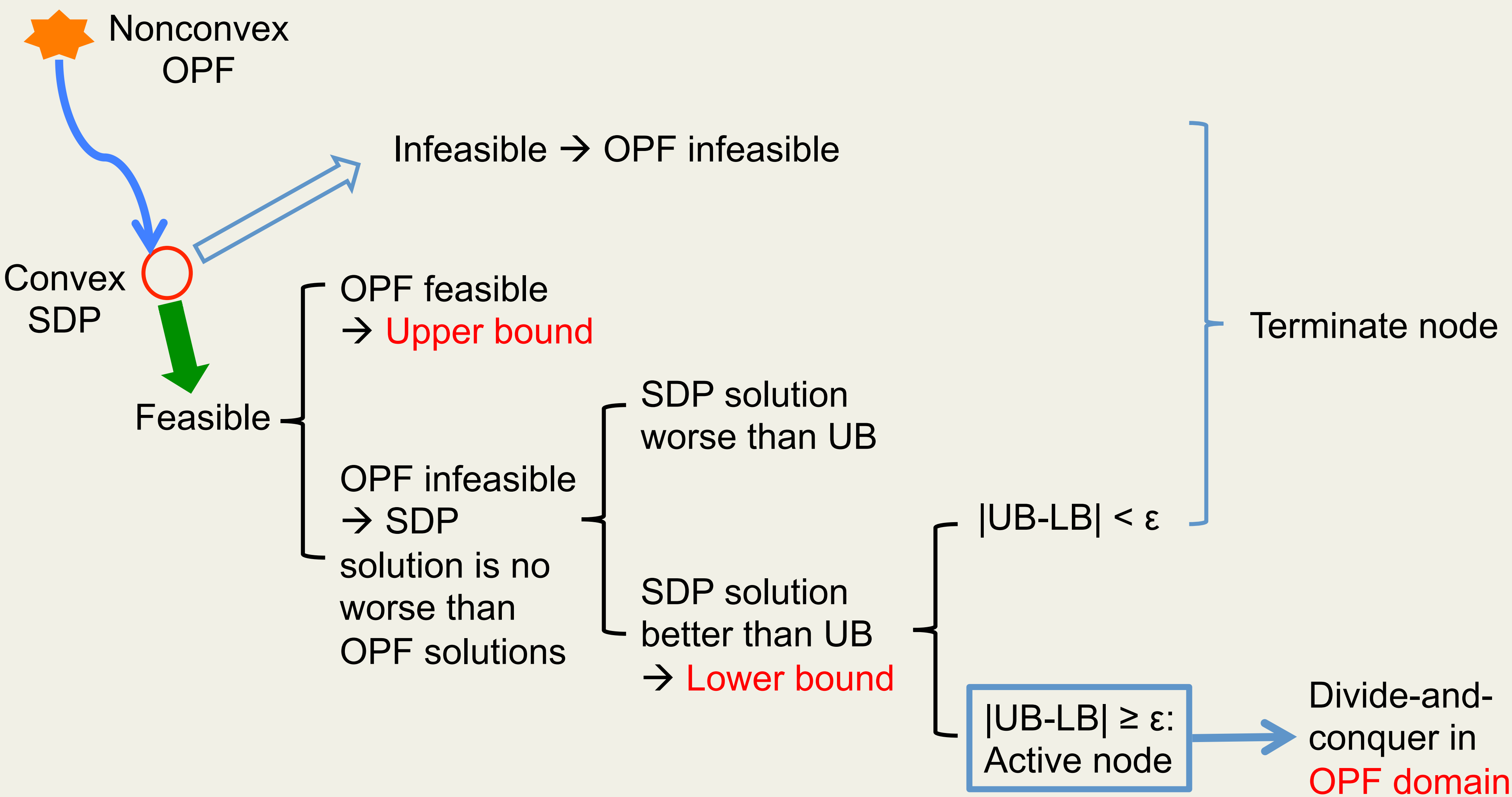


Algorithm 2014

- Start with MATPOWER
  → Initial lower bound
- Upper bound set by SDP
- Divide-and-conquer
  - Voltage cut
  - Angle cut
- Termination criterion
  - |UB-LB| < ε
  - $\varepsilon = 3\times10^{-4}$



Divide-and-Conquer Approach



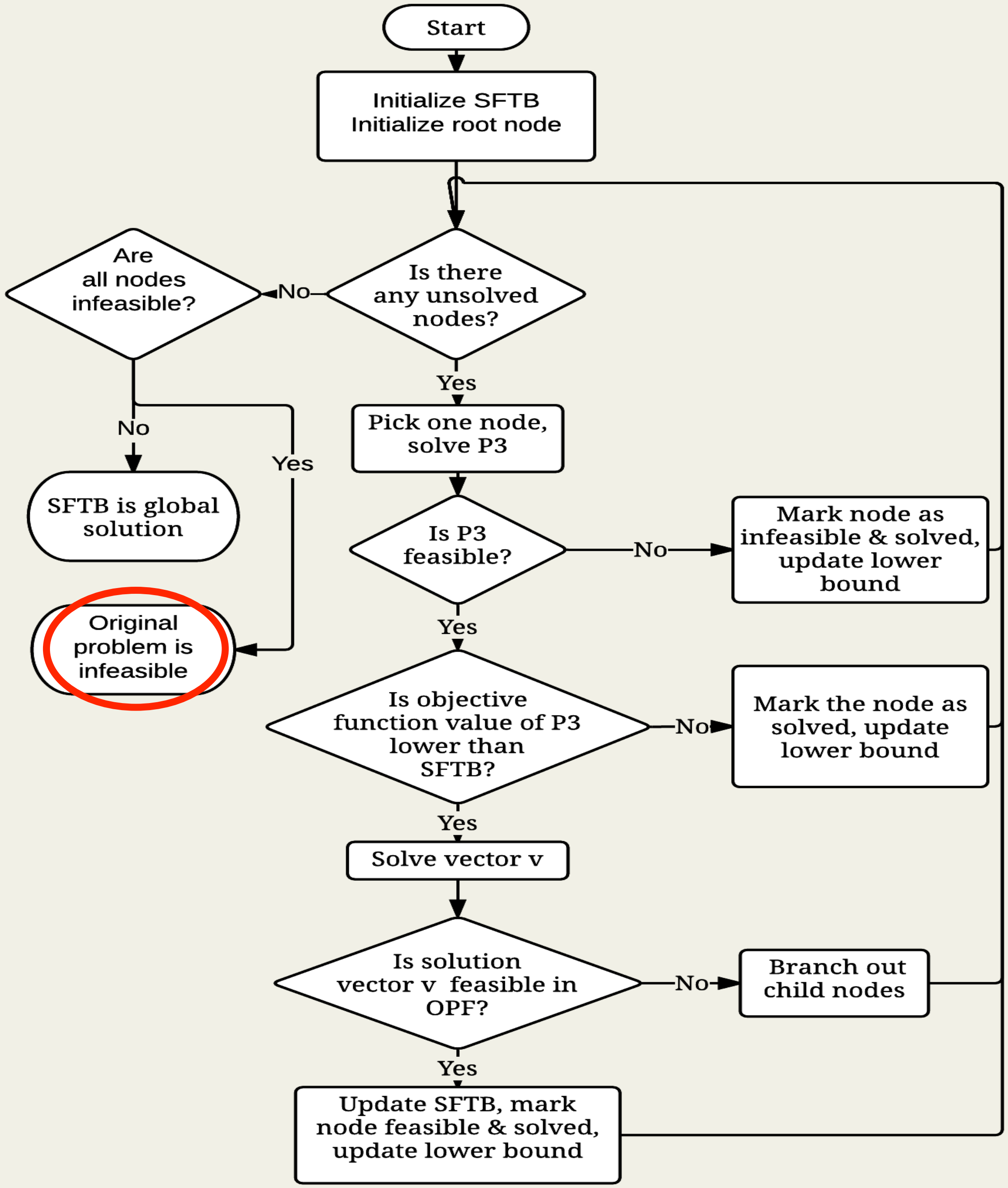
Two Questions

- Global optimizer?
  - A better solution may exist within the epsilon gap
  - Answer: global solution with an epsilon-gap is not uncommon
    - Commercial software such as BARON
    - J. Global Optimization
  - Zero epsilon-gap or an epsilon-gap less than numerical error is computationally expensive
- What if an NLP solver does not find a solution?
  - NLP fails to find solution ≠ infeasible OPF
  - SDP relaxes/expands the feasible region of OPF
  - Infeasible SDP guarantees the infeasibility of OPF

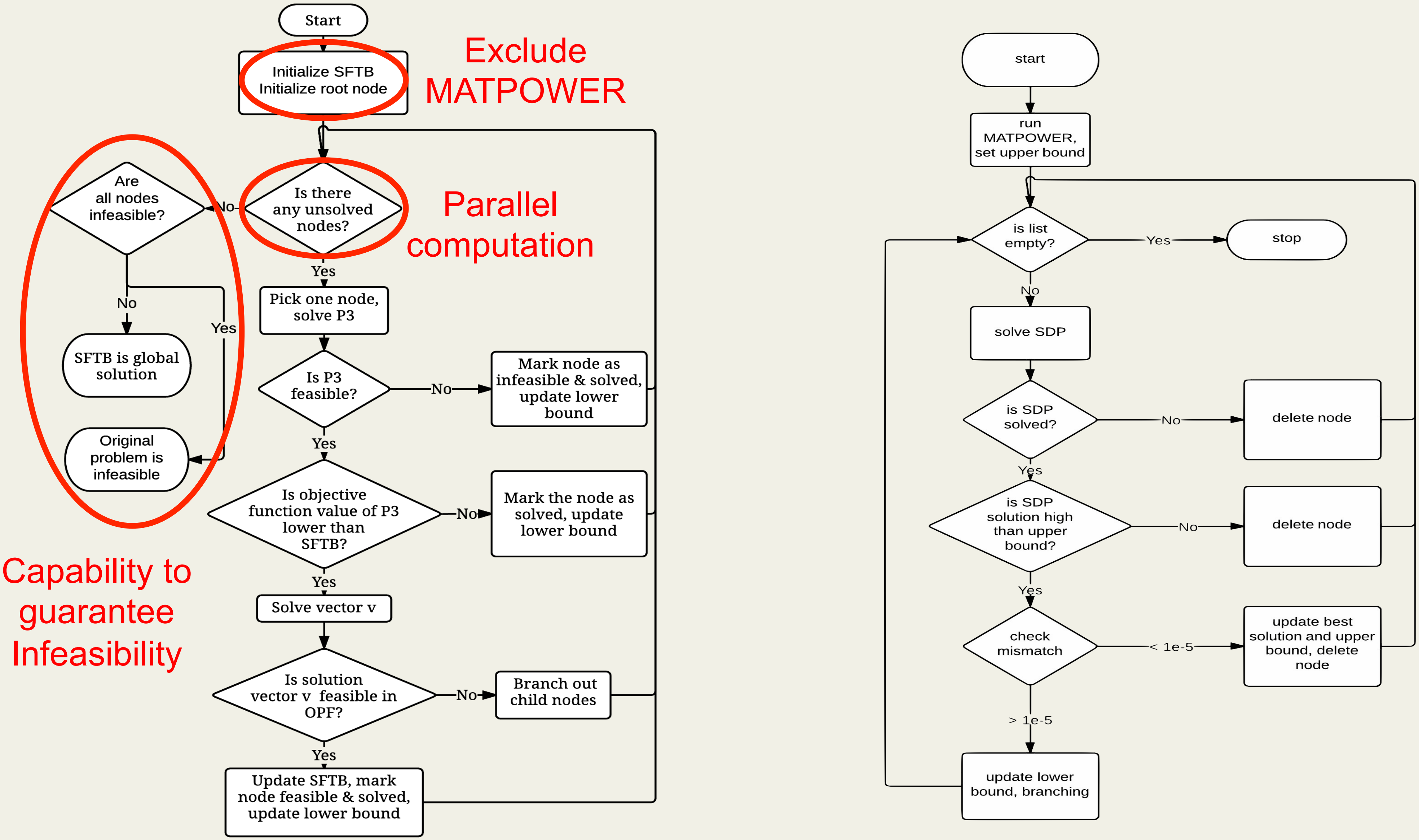
→ Major change in the algorithm

Algorithm

- Infeasible NLP
  - Infeasible SDP
    → Infeasible OPF
  - Feasible SDP
    - Best infeasible solution
    → Lower bound
    → D&C
- SDP finds a feasible OPF solution
  - Best feasible solution → Upper bound



What's New?

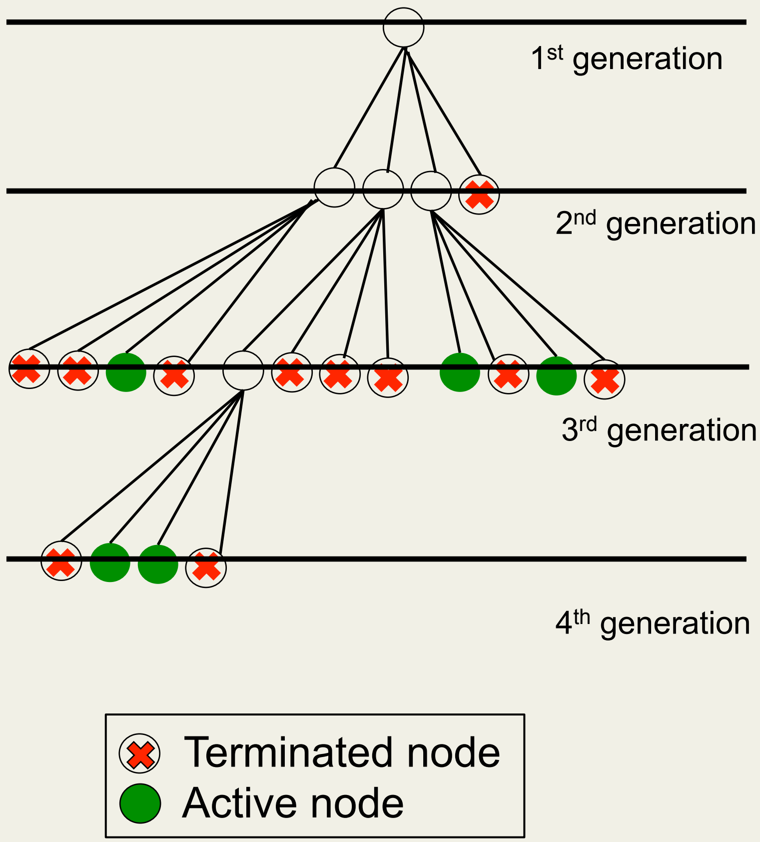


Start
Initialize SFTB
Initialize root node
Exclude MATPOWER
Is there any unsolved nodes?
Parallel computation
Are all nodes infeasible?
No
Yes
SFTB is global solution
Original problem is infeasible
Pick one node, solve P3
Is P3 feasible?
Mark node as infeasible & solved, update lower bound
Is objective function value of P3 lower than SFTB?
Mark the node as solved, update lower bound
Solve vector v
Is solution vector v feasible in OPF?
Branch out child nodes
Update SFTB, mark node feasible & solved, update lower bound
Capability to guarantee Infeasibility
start
run MATPOWER, set upper bound
is list empty?
stop
solve SDP
is SDP solved?
delete node
is SDP solution high than upper bound?
delete node
check mismatch
< 1e-5
update best solution and upper bound, delete node
> 1e-5
update lower bound, branching

Selection Criterion to Prune

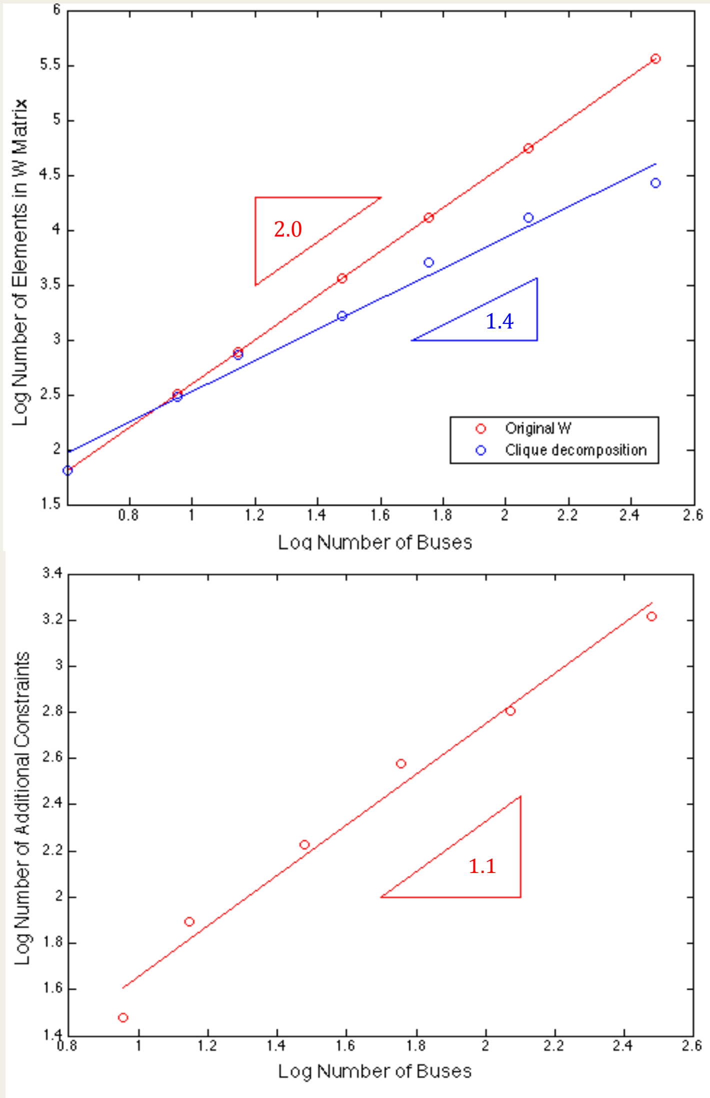
- Best infeasible SDP solution among active nodes set the lower bound
- Upper bound set by the best feasible solution
- How to choose a node to prune
  - SDP finds infeasible but better solution than the upper bound
  - Choose the "best" nodes among active nodes
  - Measure of "good" nodes
    - Close to feasible solution from the eigenvalues of *W*

$$\arg\max\left[1-\left(\frac{1}{\lambda_{\max}}\right)\sum_{n=2}^{N}\lambda_n\right]$$



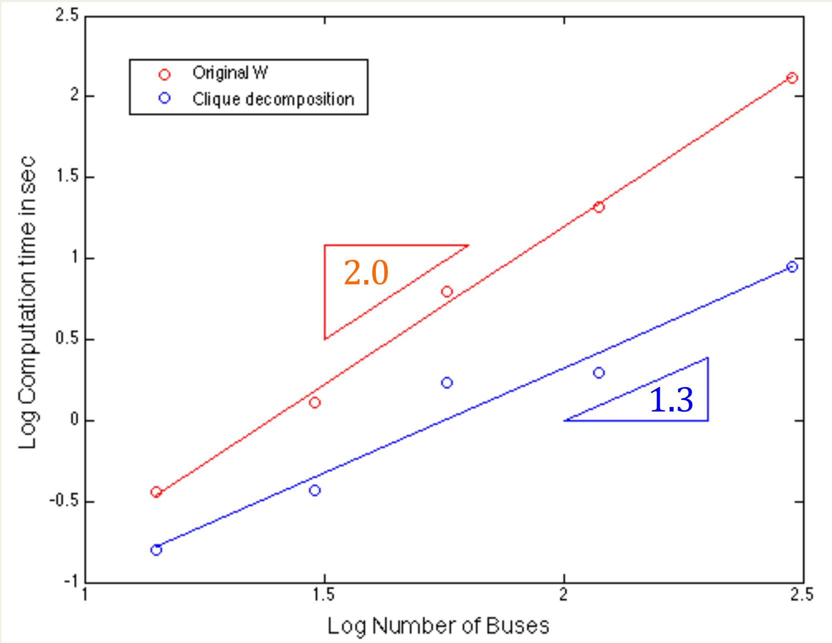
Clique Decomposition I

- Number of elements in W is in $\vartheta(N^2)$
  - Computationally inefficient
  - Infeasible to solve a large-scale OPF (≥ 30-bus case)
- Connectivity in transmission grid is low
  - Sparsity of incidence matrix
  - Decompose into small cliques
  - Large single W → small multiple W's
- Number of independent variables in W decreases (top)
- Number of equality constraints increases to make cliques consistent with W's (bottom)



Clique Decomposition II

- Independent variables in *W*
  - Original *W*: $\vartheta(N_{bus}^{2})$
  - Theoretical max with clique decomposition: $\vartheta(n \times N_{bus})$
  - IEEE model systems: $\vartheta(N_{bus}^{1.3})$
- Computation time decreases →
- SDP solvable for large-scaled systems (≥ 30 bus)



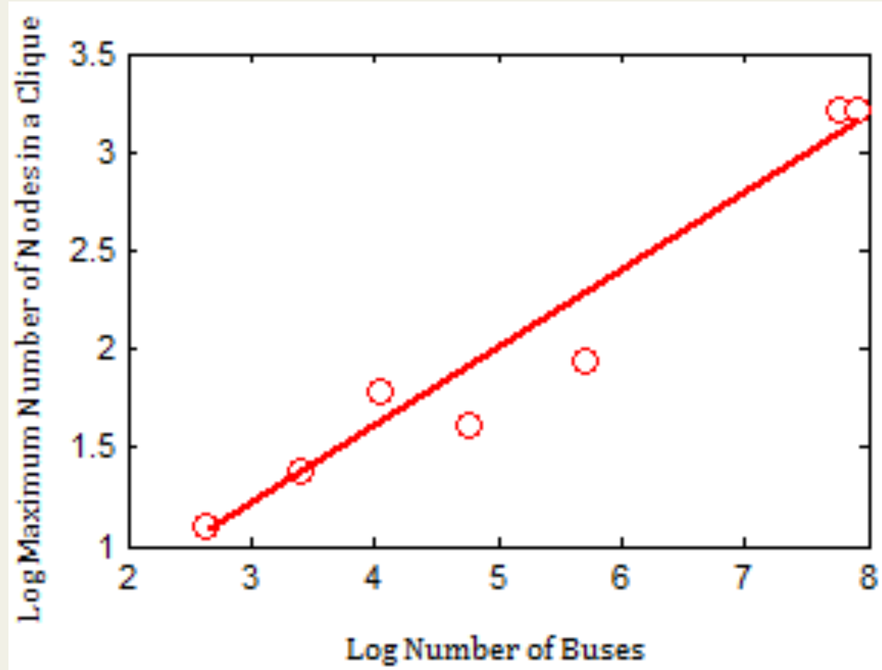
Clique Decomposition III

- Clique decomposition tends to create many small cliques
  - Many equality constraints
  - Computational inefficiency may occur
- SDP with clique decomposition reveals partial information
  - Rank(W) = max {Rank(w_j)} where w_j is the jth clique
  - A low-rank approximation with the rank yields v
  - Many elements in W are evaluated for a better approximation
- Merging small cliques can increase
  - Computation efficiency
  - Number of elements in W evaluated

M. Fukuda et. al., "Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework", *SIAM J. Optim.*, pp. 647-674, 2000

Merging Cliques I

- Molzahn et. al. suggested merging cliques
  - Sparsity for achieving high computational efficiency
  - No specific control of the individual clique size
- Observation
  - Computation time critically depends on the largest clique size
  - Relation between the largest clique size and the number of busses in IEEE cases
- Merge and create a relatively large matrix for high speed

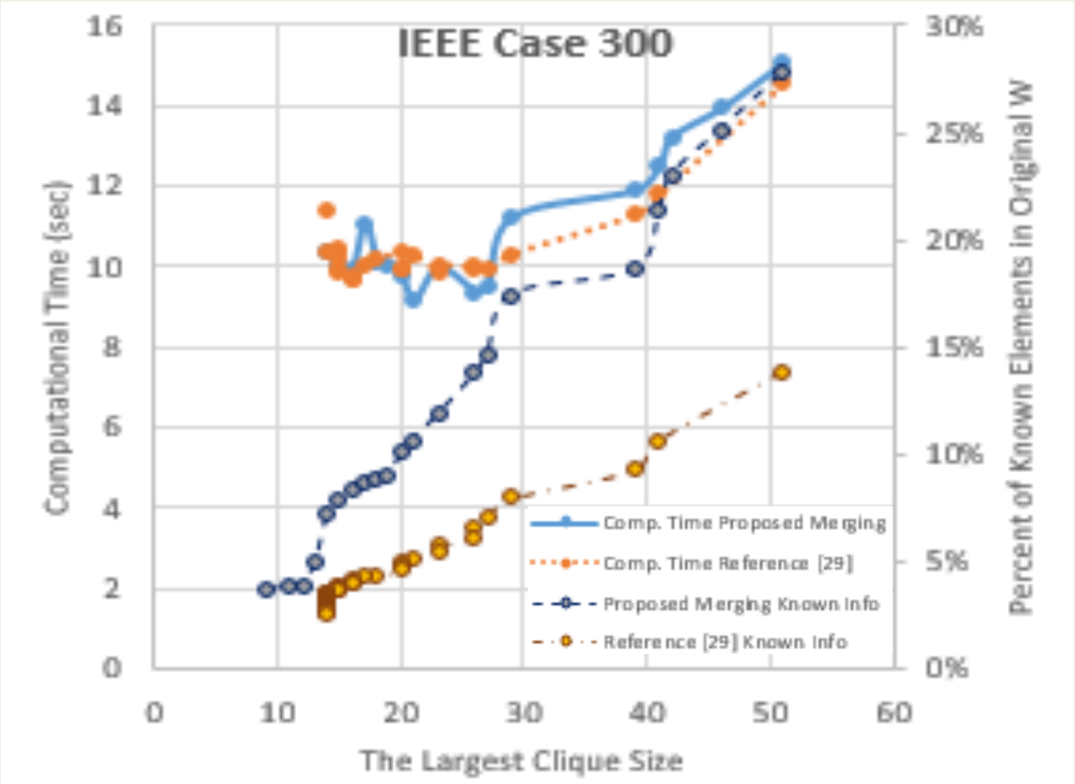


	Case Name - # of Bus						
	14	30	57	118	300	2383	2736
# of Cliques	12	26	52	108	278	2,312	2,652
The largest Clique Size	3	4	6	5	7	25	25
# of Equality Constraints	103	233	758	1,394	3,631	44,729	51,922
# of Independent Variables	241	543	1,477	2,812	7,167	76,498	88,634

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, Apr. 2013

Merging Cliques II – Computed Elements in *W*

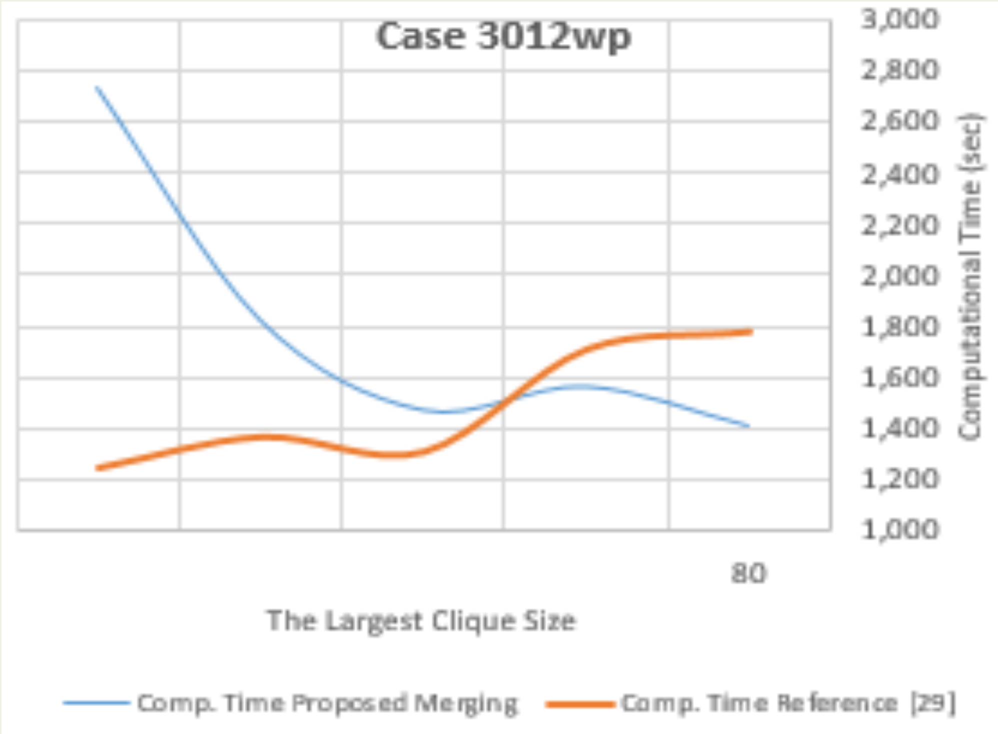
- Molzahn's method (orange)
  - Merging reveals elements in *W*
  - No specific control of the largest clique
- Our approach (blue)
  - Control the largest clique
  - Same computation time, more elements in *W* are evaluated



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, Apr. 2013

Merging Cliques III – Computational Efficiency

- Merging cliques after the clique decomposition
  - Reduce the number of equality constraints
  - Less elements in *W* are evaluated
- Proposed method
  - Control the largest clique
  - Different behavior →
  - Direct relation between the largest clique size and the computational time
  - Easier & faster PSD matrix completion
    → Suitable for D&C



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, Apr. 2013

New Angle Cut

- f: real voltage, e: imaginary voltage
- Voltage angle in the Cartesian CS: $\theta_i = \tan^{-1}\left(f_i / e_i\right)$
- Angle cut imposes limits of voltage angle: $e_i \tan\theta_i^{\min} \leq f_i \leq e_i \tan\theta_i^{\max}$
- In the OPF problem, two constraints are combined

$$f_i^2 - \left(\tan\theta_i^{\min} + \tan\theta_i^{\max}\right) f_i e_i + \left(\tan\theta_i^{\min} \tan\theta_i^{\max}\right) e_i^2 \leq 0$$

- In SDP, $W = vv^T$
- Angle cut is a linear constraint in SDP

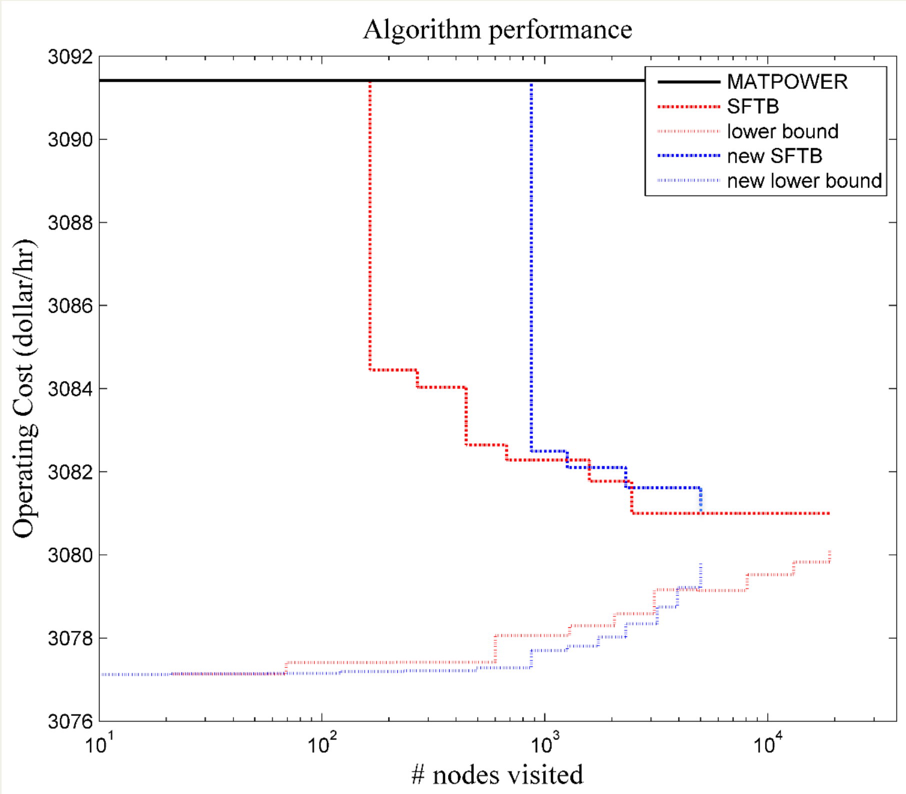
$$W_{i,i} - \left(\tan\theta_i^{\min} + \tan\theta_i^{\max}\right) W_{i,N+i} + \left(\tan\theta_i^{\min} \tan\theta_i^{\max}\right) W_{N+i,N+i} \leq 0$$

Algorithm with the New Angle Cut

- New angle cut is a single linear constraint

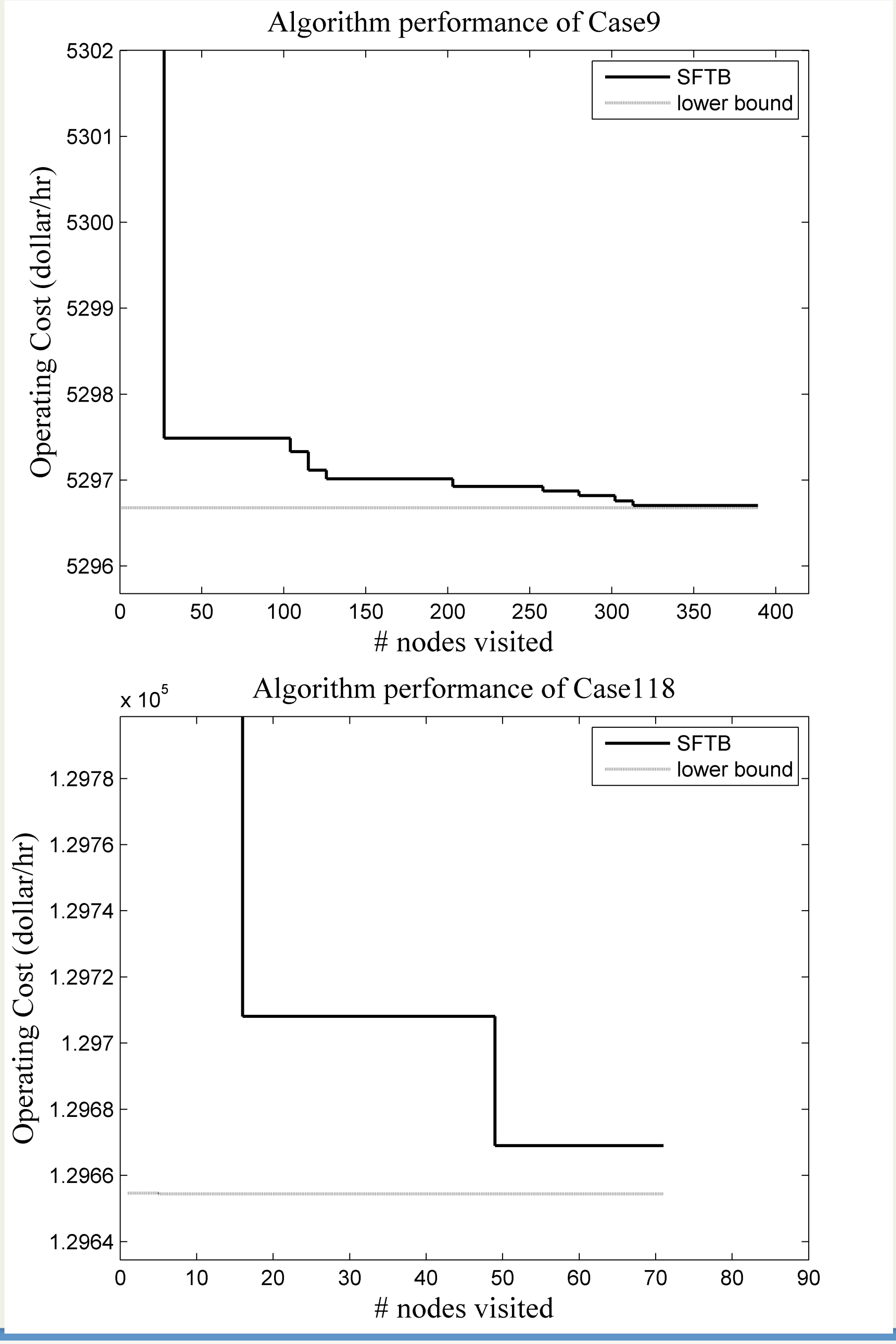
$$W_{i,i} - \left(\tan\theta_i^{\min} + \tan\theta_i^{\max}\right) W_{i,N+i} + \left(\tan\theta_i^{\min} \tan\theta_i^{\max}\right) W_{N+i,N+i} \le 0$$

- In comparison, old angle cut is from upper and lower constraints
$\tan\left(\theta_i^{\min}\right) W_{i,j} \le W_{(i+N),j}$ and
$W_{(i+N),j} \le \tan\left(\theta_i^{\max}\right) W_{i,j}$
- Terminate the process earlier, but find the same solution
→ more efficient



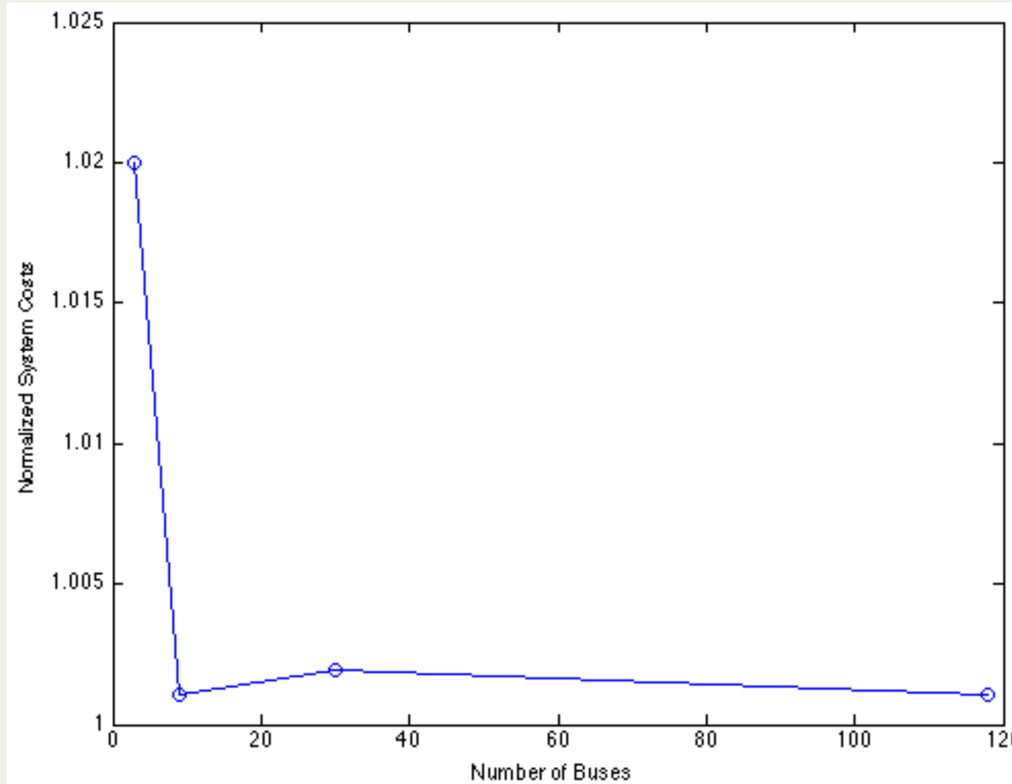
Results: Global Solutions

- Visit multiple nodes simultaneously
- Server for parallel computation
  - 9 machines
  - 2 processors/machine
  - 6 cores/processor
  - 2.50GHz Intel Xeon processor
- For a same system, depending on the loading conditions, there are changes in
  - Number of nodes visited
  - Computation time
- Epsilon gap ≤ 10^{-5}



Results I – Normalized System Costs

- In comparison to the first feasible solution found →
  - 0.1-2% cost reduction
- 0.1% savings is greater than $\$10^9$/year in US [1]
- In comparison to the MATPOWER solutions
  - MATPOWER finds the global solutions
  - Except 14-bus case [2]: 0.3% cheaper solution

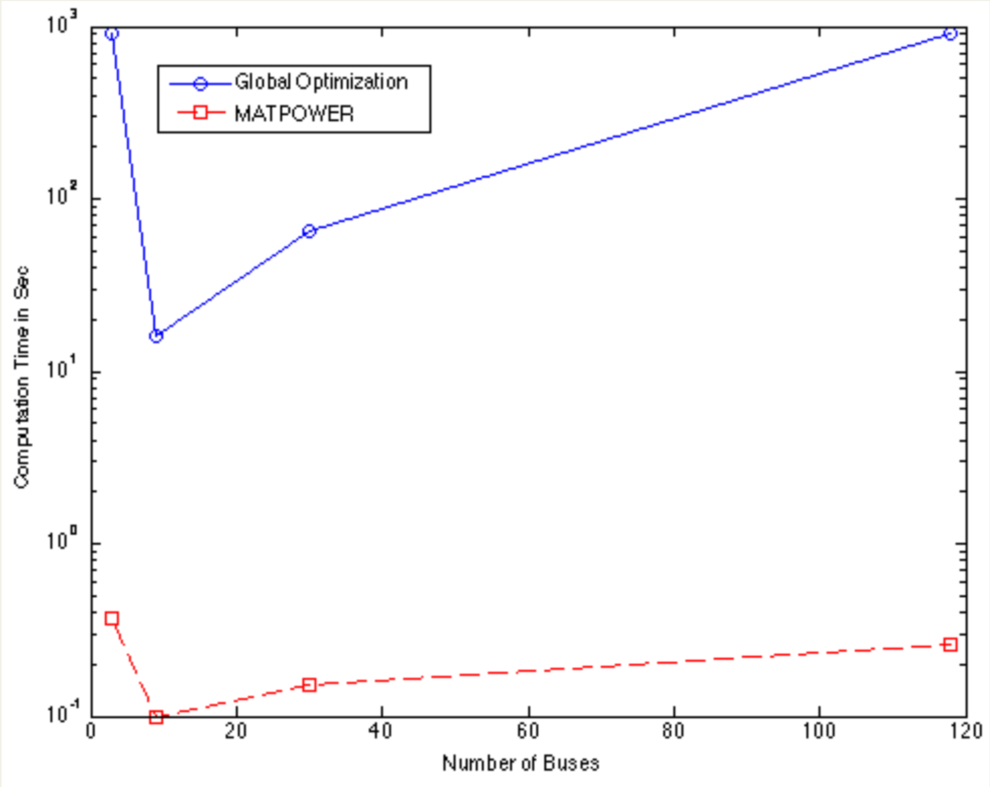


[1] R. O'Neill, "It's Getting Better All the Time (with Mixed Integer Programming)", *Harvard Electricity Policy Group*, Los Angeles, CA, Dec. 2007.

[2] R. Louca, P. Seiler, and E. Bitar. "A Rank Minimization Algorithm to Enhance Semidefinite Relaxations of Optimal Power Flow." *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*, pp. 1010-1020, 2013.

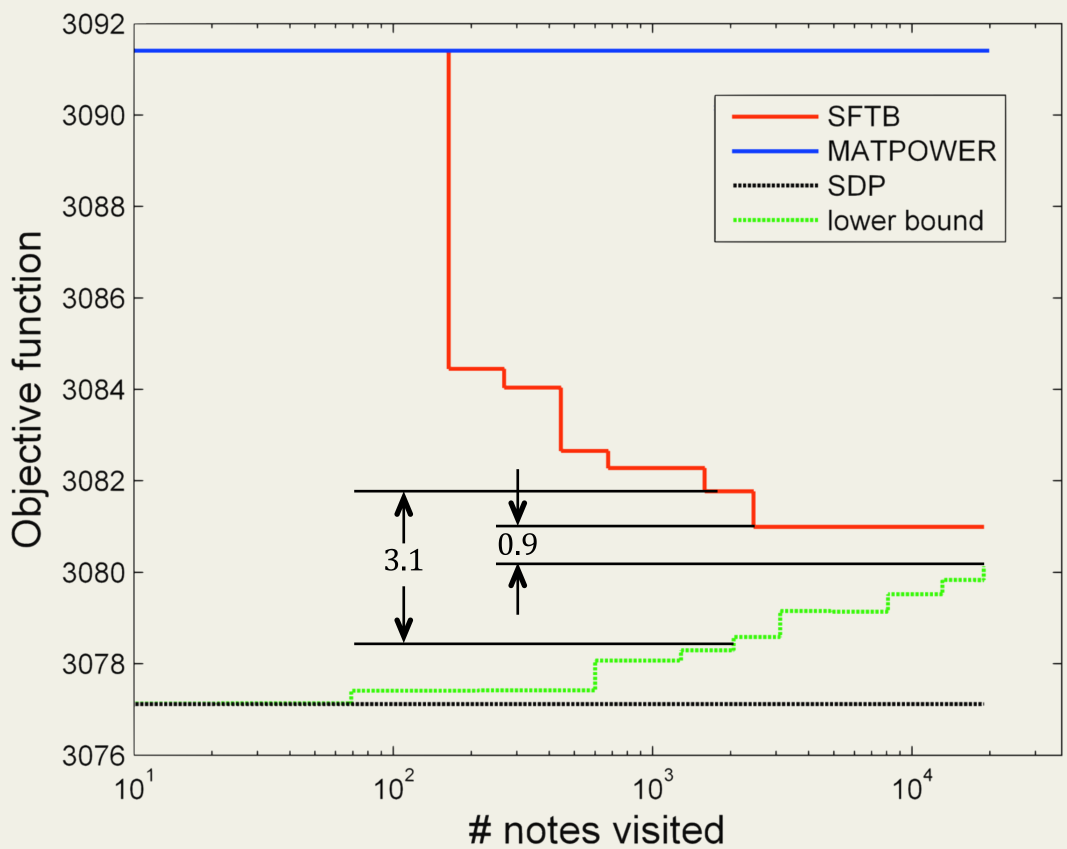
Results II – Computation Time

- Computation time
  - Typically 50-times faster with parallelization
  - 150~3,000 times slower than an NLP solver
  - BARON: 1.5 days (10^5 sec) for 9-bus case
- Epsilon-gap = 10^{-5}
  - BARON's default = 10^{-3}
  - Numerical error ≤ 10^{-8}



Global Optimizer?

- Non-zero epsilon-gap
  - Zero gap is computationally very expensive to achieve with D&C
- Terminate the process after visiting many nodes
- If there is any way to guarantee the global optimizer, the process can be terminated earlier
- Check if MATPOWER solution is the global optimizer



Trust Region Method I

- Quadratic objective function
  - Trust region constraint
  - No other constraints

$$\min \quad g^T x + 0.5 x^T B x$$
$$s.t. \quad \|x\| \le \Delta$$

- If λ and p satisfies the following conditions, p is the global solution
  - Equality constraint: $(B+\lambda I)p = -g$
  - Positive definiteness: $B+\lambda I \succeq 0$
  - Either if
    - $\lambda = 0$ and $\Delta \rightarrow \infty$ OR
    - $\lambda > 0$ and $||p|| = \Delta$

D. C. Sorensen, "Trust region methods for unconstrained optimization", *SIAM J. Numerical Analysis*, 19 (1982), pp. 409-426.

Trust Region Method II

- General quadratic optimization
  - Trust region constraint
  - Equality constraints
  - Inequality constraints

$$\begin{aligned} \min \quad & g^T x + 0.5 x^T B x \\ s.t. \quad & \|x - x_0\| \le \Delta \\ & h^T x + 0.5 x^T C x = 0; \;\; \mu \\ & k^T x + 0.5 x^T D x \le 0; \;\; \sigma \end{aligned}$$

- If μ^*, σ^*, λ, and p satisfies the following conditions, p is the global solution inside the trust-region → epsilon gap is *NOT* necessary

$$\left(B + \mu^* C + \sigma^* D + \lambda I\right) p = -g - \mu^* h - \sigma^* k + \lambda x_0$$

$$B + \mu^* C + \sigma^* D + \lambda I \succeq 0$$

$$\left\{\sigma^* = 0 \;\&\; k^T x + 0.5 x^T D x \le 0\right\} OR \; \left\{\sigma^* > 0 \;\&\; k^T x + 0.5 x^T D x = 0\right\}$$

$$\left\{\lambda = 0 \;\&\; \Delta \to \infty\right\} OR \; \left\{\lambda > 0 \;\&\; \left\|p - x_0\right\| = \Delta\right\}$$

Global optimizer

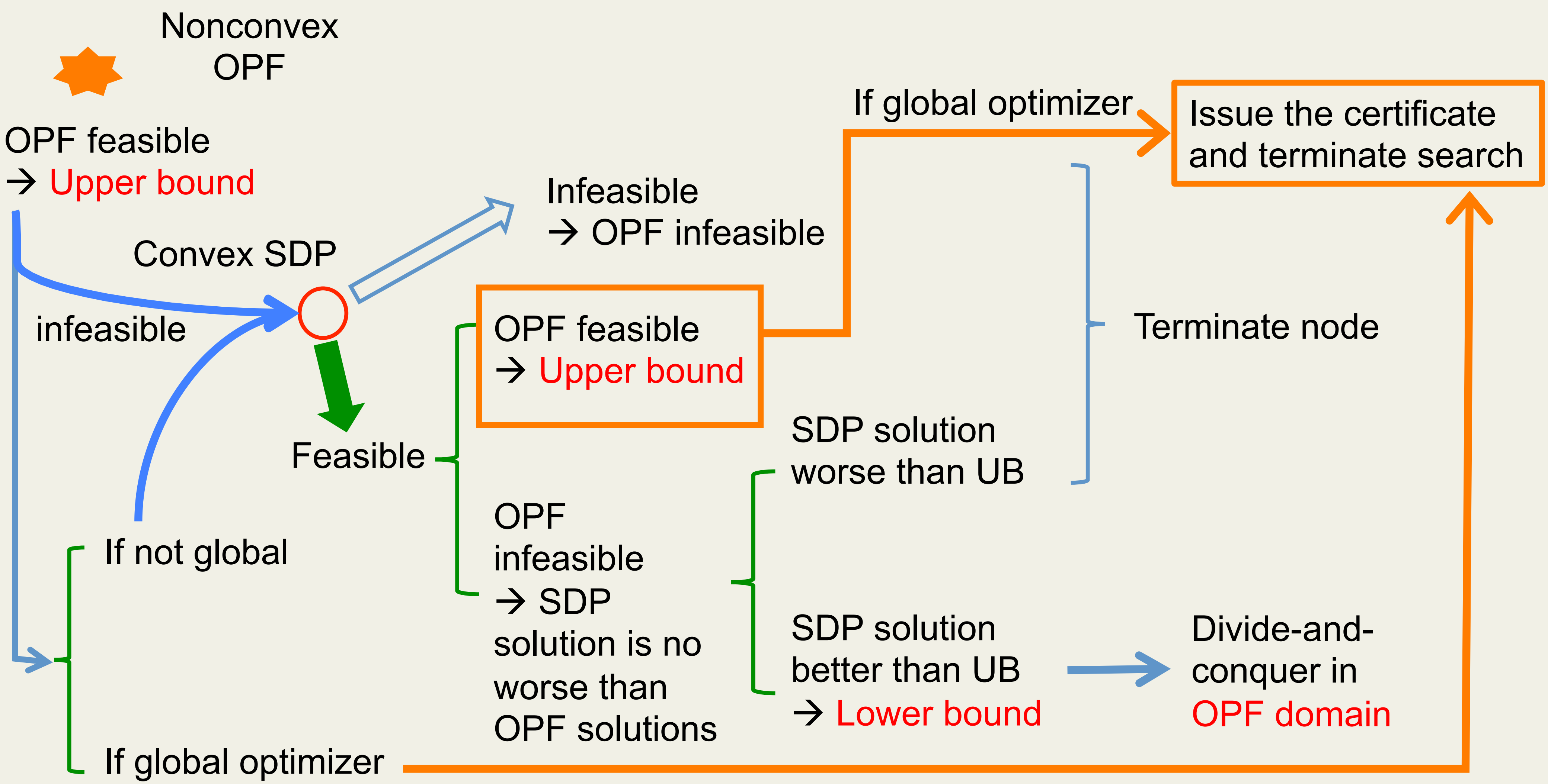
Trust Region Method for Optimal Power Flow

- Rewrite OPF into QCQP
  - Flow constraints are in quartet
  - Introduce real and reactive power injection variable over a line

$$\min \quad 0.5 p^T B p + g^T p$$

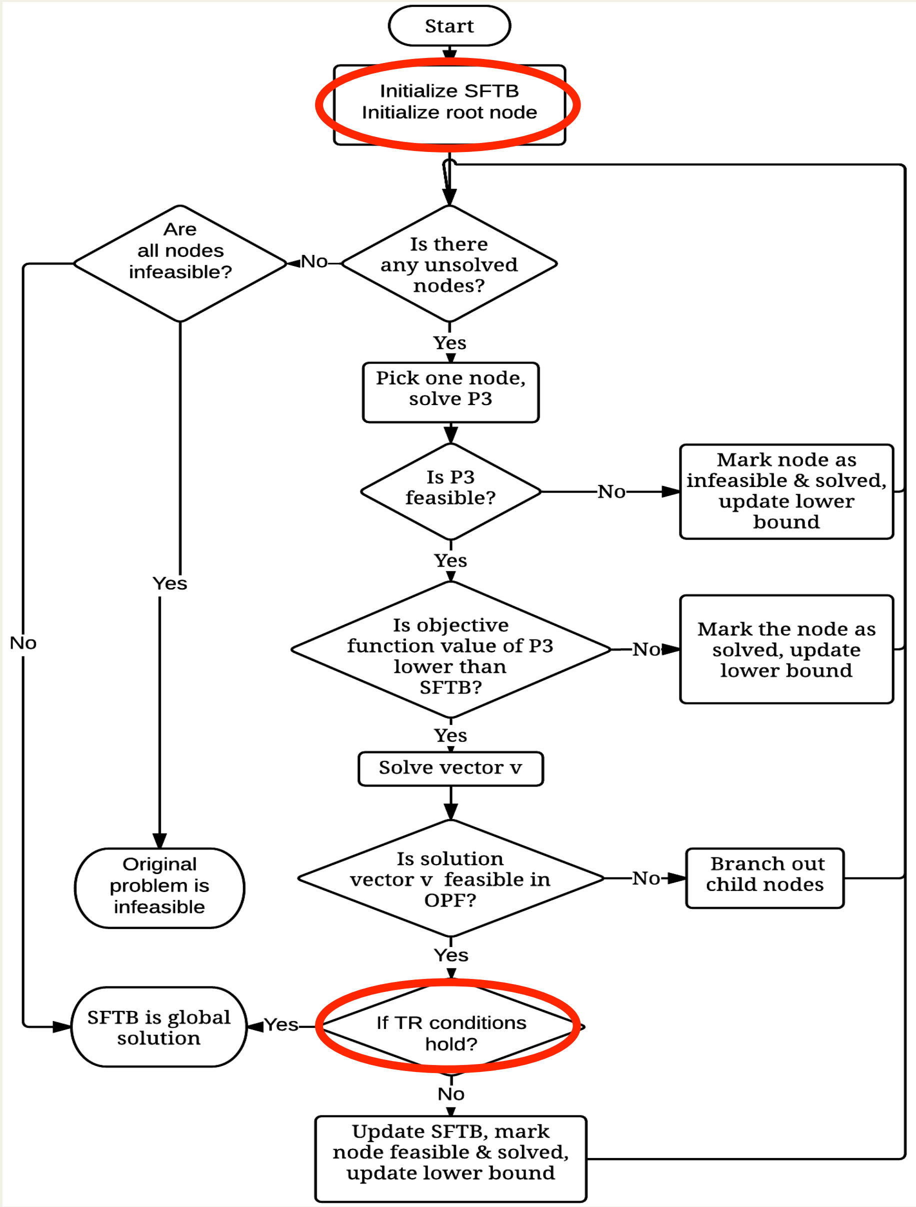
$$s.t. \begin{cases} v^T \Phi_j^P v = p_j - p_{d,j}, \; v^T \Phi_j^Q v = q_j - q_{d,j} \\ v^T \Psi_k^P v = p_k, \; v^T \Psi_k^Q v = q_k, \; p_k^2 + q_k^2 \le c_k^2 \\ v_{\min}^2 \le v^T M v \le v_{\max}^2 \\ p_{\min} \le p \le p_{\max}, \; q_{\min} \le q \le q_{\max} \end{cases}$$

Revisit to the Divide-and-Conquer Approach



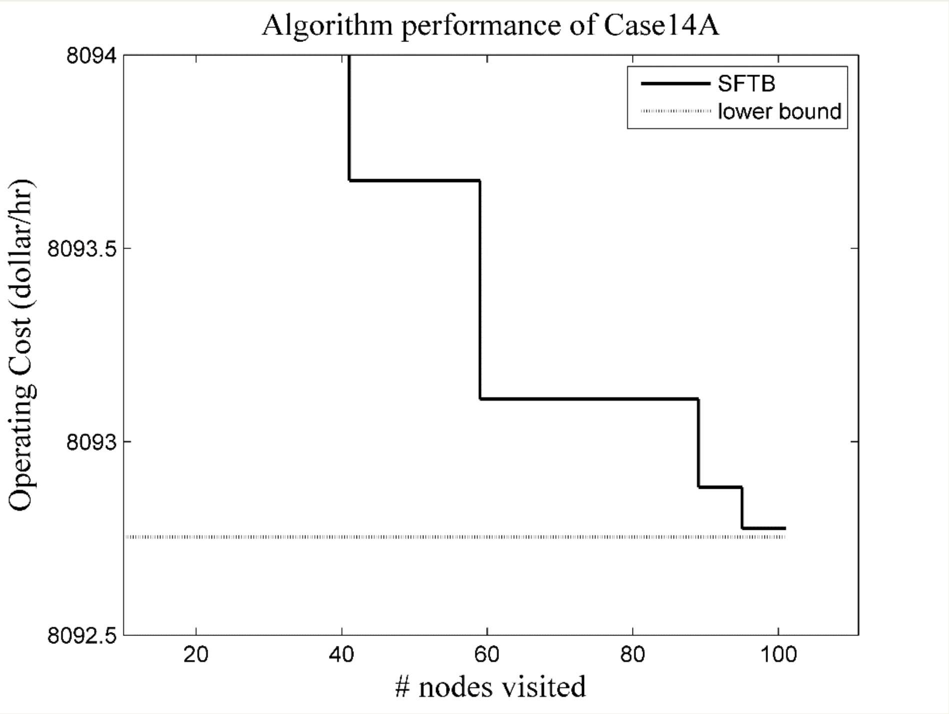
New Algorithm Incorporating the Certificate

- SDP feasible solution
- OPF local solution
- Issue the certificate to guarantee the global optimizer
- Epsilon-gap is removed from the algorithm
- Process to check |UB – LB| ≤ ε is not necessary



New Termination Criterion with the TR Certificate

- During the searching process, find local optimizers
- Trust-region global optimality condition
  - Local optimizers do not satisfy
  - Global optimizer does → Terminate the search process
- Performance of D&C with TRC
  - Terminated when gap > 10^{-5}
  - Only 100 nodes visited
  - Tested with Case14A only



This Approach vs. MATPOWER

- MATPOWER
  - High efficiency to search for an optimizer
  - Based on our experience up to now, MATPOWER finds the global solution except one case (Case14 with modified offers, by 0.2%)
- This approach
  - MATPOWER finds an optimizer
    - Most cases: Global optimizer → Issue the certificate & terminate the search process
    - Some cases: 10 times slower than MATPOWER, but much faster than an algorithm with ε-gap (150-3,000 times slower)
  - Guarantees
    - Infeasibility of an OPF problem
    - Global solution
  - Finds multiple local optimizers

Conclusions

- Our D&C finds the global solution in an efficient way by
  - Dividing regions with voltage cut and angle cut
  - Finding the ideal place to prune using the sub-optimization problem
  - Terminating a node efficiently
- Our D&C is modified
  - Added capability to guarantee the infeasibility of OPF
  - New angle cut in a single linear form
  - Clique decomposition & merging
  - Parallel computation to enhance efficiency
  - Early termination with the Trust-Region Certificate