Noname manuscript No.
(will be inserted by the editor)

Graphical models for optimal power flow

Krishnamurthy Dvijotham · Michael Chertkov · Pascal Van Hentenryck · Marc Vuffray · Sidhant Misra

Received: date / Accepted: date

arXiv:1606.06512v1 [cs.SY] 21 Jun 2016

Abstract Optimal power flow (OPF) is the central optimization problem in electric power grids. Although solved routinely in the course of power grid operations, it is known to be strongly NP-hard in general, and weakly NP-hard over tree networks. In this paper, we formulate the optimal power flow problem over tree networks as an inference problem over a tree-structured graphical model where the nodal variables are low-dimensional vectors. We adapt the standard dynamic programming algorithm for inference over a tree-structured graphical model to the OPF problem. Combining this with an interval discretization of the nodal variables, we develop an approximation algorithm for the OPF problem. Further, we use techniques from constraint programming (CP) to perform interval computations and adaptive bound propagation to obtain practically efficient algorithms. Compared to previous algorithms that solve OPF with optimality guarantees using convex relaxations, our approach is able to work for arbitrary distribution networks and handle mixed-integer optimization problems. Further, it can be implemented in a distributed message-passing fashion that is scalable and is suitable for "smart grid" applications like control of distributed energy resources. We evaluate our technique numerically on several benchmark networks and show that practical OPF problems can be solved effectively using this approach.

Keywords Constraint Programming · Graphical Models · Power Systems

This work was supported by Skoltech through collaboration agreement 1075-MRA. The work at LANL was carried out under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy under Contract No. DE-AC52-06NA25396.

Krishnamurthy Dvijotham
Computing and Mathematical Sciences, California Institute of Technology
E-mail: dvij@cs.washington.edu

Michael Chertkov, Marc Vuffray Sidhant Misra
T-Divison & Center for Nonlinear Studies, Los Alamos National Laboratory

Pascal Van Hentenryck
Industrial and Operations Engineering, University of Michigan, Ann Arbor

1 Introduction

In this paper, we study a novel application of well-known AI techniques (constraint programming and inference in graphical models) to a difficult engineering problem - the optimization of resources in a power distribution network. As larger amounts of renewable generation sources (solar/wind) are incorporated into the power grid, the availability of generation capacity becomes uncertain (due to dependence on unpredictable weather phenomena). However, the physics of the power grid imply the need to maintain real-time balance between demand and generation. Exploiting the flexibility of electricity demand becomes central for solving this problem, which is the core of the "smart grid" vision (`https://www.smartgrid.gov/`). Thus, future grids will require efficient algorithms that can process data from millions of consumers and efficiently compute optimal ways of exploiting demand-side flexibility while respecting engineering constraints. In this paper, we develop a novel algorithm guaranteed to compute an approximately optimal solution for this problem in polynomial time.

More concretely, we study the Optimal Power Flow (OPF) problem [8]. At an abstract level, one can view this as a network flow optimization:

$$\begin{aligned} \text{minimize} \quad & \text{cost of generating electicity} \\ \text{subject to} \quad & \text{conservation of flows, flows consistent with voltages,} \\ & \text{demands are met, engineering limits are respected} \end{aligned}$$

where the engineering limits typically refer to capacities of the transmission lines in the power grid and limits on voltages.

However, as opposed to a standard network-flow problem for which there are well-known efficient algorithms, the physics of electric power flow make the above problem challenging. Electrical flows cannot be arbitrary, but are driven by differences in voltages, so that the flow on a transmission line is a nonlinear function of the voltage difference between the two ends of the line. Due to this nonlinear constraint, the OPF problem becomes non-convex. In fact, it is strongly NP-hard over arbitrary networks [5] and weakly NP-hard over tree-structured networks [19]. The special case of tree-structured networks is particularly important in the context of the smart grid as distribution networks (which connect high-voltage long-distance power transmission network to individual consumers) are typically tree-structured. In order to exploit demand side flexibility, we will need efficient OPF algorithms on tree-networks.

In recent years, several researchers have studied applications of convex relaxation techniques to the OPF problem [20]. In particular, for the tree OPF problem, elegant results have been developed characterizing the conditions under which convex relaxations of OPF are guaranteed to be exact [7, 18, 23, 13]. The most general results are presented in [13] and cover several practical instances of OPF over tree networks. However, the conditions for exactness

require assumptions that are incompatible with the above "smart grid" applications (absence of discrete variables, limits on flow reversal, ...).

In this paper, we develop a new approach to solving optimal power flow on tree-structured networks by using techniques from Constraint Programming (CP) and Graphical Models (GM). We first restate the tree OPF problem as an inference problem over a tree-structured factor graph. Based on this representation, we develop an algorithm that computes a super-optimal approximately feasible solution. The running time of the algorithm is linear in the size of the network and polynomial in $\frac{1}{\epsilon}$, where ϵ is the error tolerance allowed. Relative to the existing algorithms based on convex relaxations, the approach we develop has the following advantages:

- It can handle mixed-integer optimization problems (involving both discrete and continuous variables) and hence is capable of addressing load-control and distributed generation applications with discrete components such as on/off constraints, switching transformer taps, and capacitor banks.
- Unlike [13], the approach does not require restrictive assumptions on flow directionality or voltage limits. It also do not require costs to be convex, allowing arbitrary (possibly discontinuous) piecewise-linear costs.
- The resulting algorithm is inherently distributed and can be implemented using a message-passing framework, which is expected to be a significant advantage for future distribution networks.

On the other hand, a disadvantage of our algorithm is that we only produce approximate solutions and there may be cases where achieving acceptable error tolerances will require intractably fine discretizations. However, we show that, for several practical OPF problems, this problem can be alleviated by leveraging CP techniques.

A closely related approach, developed in an abstract form and stated in the language of linear programming, was presented in [4]. The authors present a general framework that computes super-optimal and approximately feasible solutions to graph-structured mixed-integer polynomial optimization problems. While our approach provides similar guarantees, our main contributions relative to that work are as follows:

1. We restate the approach in the intuitive language of graphical models. This allows us to take advantage of the inference techniques developed for graphical models [26, 24], and study problems beyond optimization (probabilistic inference, for example).
2. As opposed to the point discretization approach employed in [4], we develop an interval discretization approach, that allows us to use interval CP techniques (bound-tightening, constraint propagation etc.) to achieve practically efficient implementation of the algorithm.

We note that a related approach has been used for finding approximate Nash-equilibria in tree-structured graphical games [15]. Finally, we note that related ideas have been explored in detail in the graphical models and constraint programming literature [10, 26, 14, 24].

The rest of this paper in organized as follows. Section 2 provides the background on power systems and graphical models necessary for this paper. Section 3 formulates the OPF problem over a tree network as a MAP-inference problem over a tree-structured factor graph. Section 4 describes a finite dynamic-programming algorithm based on an interval discretization of the variables in the factor graph and presents guarantees associated with the algorithm. Section 5 evaluates our algorithm and compares it to off-the-shelf solvers on a number of IEEE distribution network test cases. Finally, Section 6 summarizes our findings and presents directions for future work.

2 Background

In this Section, we introduce all the necessary background on Alternating Current (AC) power flows and graphical models required to follow the development of the algorithm in this paper. We have attempted to make this Section self-contained providing sufficient details for the purposes of this paper. Interested readers may consult textbooks on power engineering [25,22] and graphical models [16,10,26] for further details.

In the following, $\mathbb{C}$ denotes the set of complex numbers and $\mathbb{R}$ the set of real numbers.$\mathbf{j} = \sqrt{-1}$ to avoid confusion with currents as is traditional in power systems. For $x \in \mathbb{C}$, we use $\overline{x}$ to denote the complex conjugate of x, $\mathrm{Re}\,(x)$ the real part, $\mathrm{Im}\,(x)$ the imaginary part, and $\angle x$ the phase of x. For $x, y \in \mathbb{C}$, $a \leq b$ denotes the pair of inequalities $\mathrm{Re}\,(a) \leq \mathrm{Re}\,(b)\,, \mathrm{Im}\,(a) \leq \mathrm{Im}\,(b)$.

2.1 AC Power Flow over a Tree Network

We will work with a power distribution network transporting Alternating-Current power (AC power). Mimicking power engineering terminology, nodes in the network are called buses and edges are called transmission lines (or simply lines or branches). These networks are typically tree-structured, and the root of the tree is known as the *substation bus* - physically, this is the point at which the power distribution network is connected to the high voltage power transmission network. We label the nodes $0, \ldots, n$, where 0 is the substation bus. The network is represented as a directed graph, with all edges pointing towards the substation bus (this directionality is simply a convention and has no physical meaning - the physical power flow over this edge can be in either direction). Each node k (except the substation) has a unique outgoing edge, connecting it to its *parent node*, denoted $\mathcal{P}\,(k)$, and k is said to be a *child* of $\mathcal{P}\,(k)$. $\mathcal{C}\,(i)$ denotes the set of children of bus i. AC power flow in steady-state is described by complex voltages (known as voltage phasors) that represent sinusoidal steady-state voltage profiles. Let the complex voltage at node i be V_i and let $\mathfrak{v}_i = |V_i|^2$ denote the squared voltage magnitude. The power-flow is also a complex number, whose real part is known as *active power* and imaginary part as *reactive power*. In a tree network, every edge connects a bus and its

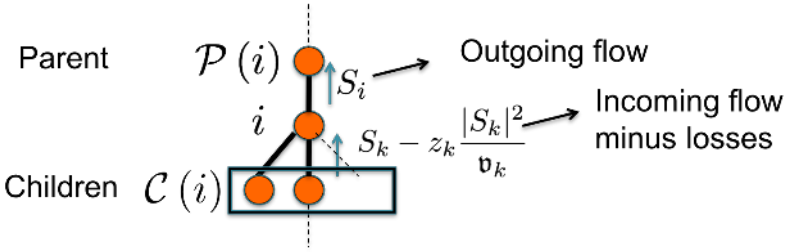


Fig. 1: AC Power flow in a tree network

parent. Thus, we can identify every edge with the child node i incident upon it, and denote its complex impedance by $z_i = r_i + \mathbf{j}x_i$ (r_i is the resistance and x_i the inductance of the line). We denote the sending-end power flow from bus i to bus $\mathcal{P}(i)$ by $S_i = P_i + \mathbf{j}Q_i$. Note that because of losses, the power received at $\mathcal{P}(i)$ is not equal to S_i. The power losses are given by $z_i \frac{|S_i|^2}{\mathfrak{v}_i}$ (see [25] or [21] for further details). Combined with the conservation of flow at each node in the network, this leads to the AC power flow equations, in the so-called branch-flow model (first presented in [3]), illustrated in Figure 1. The power flow equations in the branch flow (or Baran-Wu) form can be written as:

$$P_i = p_i + \sum_{k \in \mathcal{C}(i)} \left(P_k - r_k \left(\frac{P_k^2 + Q_k^2}{\mathfrak{v}_k} \right) \right) \quad \forall i \in \{0, \ldots, n\} \tag{1a}$$

$$Q_i = q_i + \sum_{k \in \mathcal{C}(i)} \left(Q_k - x_k \left(\frac{P_k^2 + Q_k^2}{\mathfrak{v}_k} \right) \right) \quad \forall i \in \{0, \ldots, n\} \tag{1b}$$

$$\mathfrak{v}_i = \mathfrak{v}_k + \left(r_k^2 + x_k^2\right) \frac{\left(P_k^2 + Q_k^2\right)}{\mathfrak{v}_k} - 2\left(r_i P_i + x_i Q_i\right) \quad \forall i \in \{0, \ldots, n\}, k \in \mathcal{C}(i) \tag{1c}$$

where p_i and q_i are the real and reaction power injections/consumptions at bus i, as discussed in the next Section. For the rest of this paper, this is the form of the PF equations we will use.

2.2 Optimal Power Flow (OPF) on a tree network

The optimal power flow (OPF) problem aims at finding the most efficient utilization of generation and flexible demand resources in a grid subject to the power flow constraints and engineering limits on voltages, currents and flows. At each node, there is generation and/or consumption of power. To simplify notation, we assume that there is only one entity (generator or consumer) at each node in the network (but this restriction is not necessary). Both generators and flexible consumers are characterized by their injection domain

$(p_i, q_i) \in \mathbf{S}_i$. The domain may be a finite set (for modeling discrete load control where a load can take on one of a set of possible values, for example) or an infinite set (for modeling continuous constraints like minimum/maximum generation limits). Inflexible generators or consumers are modeled by choosing $\mathbf{S}_i$ to be a singleton set. Additionally, each generator has a cost of production $c_i(p_i, q_i)$ and similarly every flexible consumer may be compensated for adjusting consumption, and this compensation is also denoted $c_i(p_i, q_i)$.

With these assumptions, a generic OPF problem can be stated as

$$\underset{p,q\mathfrak{v},P,Q}{\text{Minimize}} \sum_{i=0}^{n} c_i(p_i, q_i) \ \text{(Minimize Cost)} \tag{2a}$$

$$\text{Subject to } (1a), (1b), (1c) \tag{2b}$$

$$\mathfrak{v}_i^L \leq \mathfrak{v}_i \leq \mathfrak{v}_i^U \quad \forall i \in \{1, \ldots, n\} \tag{2c}$$

$$P_i{}^L \leq P_i \leq P_i{}^U, Q_i{}^L \leq Q_i \leq Q_i{}^U \, \forall i \in \{1, \ldots, n\} \tag{2d}$$

$$(p_i, q_i) \in \mathbf{S}_i \quad \forall i \in \{1, \ldots, n\} \tag{2e}$$

$$P_0 = 0, Q_0 = 0, \mathfrak{v}_0 = \mathfrak{v}^{ref} \tag{2f}$$

The costs and flow balance constraints are described above. The constraints (2c) and (2d) come from engineering limits - devices connected to the grid only work properly for a certain range of voltages, and the flow limits are related to the capacity of transmission lines (both due to dynamic stability and thermal limitations). The constraint (2f) enforces that there is no current or flow going upstream from the substation bus and that the substation bus voltage is set to a fixed reference value, $\mathfrak{v}^{ref}$. Note that more general OPF problems can be handled with our approach (adding tap transformer positions, capacitor banks etc.), but we restrict ourselves to this basic problem to simplify notation and make the exposition clear. We make assumptions on the cost function and problem data as stated below:

Assumption 1 *The injection constraint set can be partitioned as:*

$$\mathbf{S}_i = \cup_{t=1}^{|\mathbf{S}_i|} \mathbb{I}^s(t), \mathbb{I}^s(t) = \left\{(p_i, q_i) : p_i{}^L(t) \leq p_i \leq p_i{}^U(t), q_i{}^L(t) \leq q_i \leq q_i{}^U(t)\right\}$$

and the cost function $c_i(p_i, q_i)$ is a linear function over $\mathbb{I}^s(t)$:

$$c_i(p_i, q_i) = a_i(t) p_i + b_i(t) q_i + c_i(t)$$

Assumption 2 $\exists M > 0$ *such that:*

1. $[P_i{}^L, P_i{}^U], [Q_i{}^L, Q_i{}^U], [\mathfrak{v}_i{}^L, \mathfrak{v}_i{}^U] \subseteq [-M, M] \quad i \in \{1, \ldots, n\}$ *(voltages/flows are bounded uniformly across all nodes).*
3. $\mathfrak{v}_i{}^L \geq \frac{1}{M}$ *for* $i = 1, \ldots, n$ *(voltage lower bounds are bounded below).*
3. $|z_i| \leq M, i = 1, \ldots, n$ *(impedances are bounded uniformly across all nodes).*
4. $|\mathbf{S}_i| \leq M, i = 0, \ldots, n$ *(number of pieces in the cost is bounded).*
5. $\forall i \in \{0, \ldots, n\}, t \in \{1, \ldots, |\mathbf{S}_i|\}$: $\max\left(p_i{}^U(t) - p_i{}^L(t), q_i{}^U(t) - q_i{}^L(t)\right) \leq \frac{1}{M}$ *(the size of each piece in the piecewise linear cost is small).*

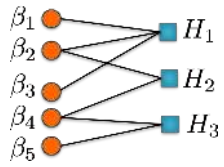


Fig. 2: Factor graph corresponding to (3)

Assumptions 1 and 2 are non-restrictive: Assumption 1 simply requires that the cost function is piecewise-linear (or can be approximated by one) and assumption 2 requires that all parameters are bounded.

2.3 Factor Graphs

A factor graph [17] is a formal tool to express the structure of an optimization problem or a probability distribution. In this paper, we focus on optimization problems and do not discuss the use of factor graphs in probabilistic inference. A factor graph is defined by specifying:

1 A set of n variables $\{\beta_i\}_{i=1}^n$ and their domains $\beta_i \in \mathcal{X}_i$.
2 A set of functions, called *factors*, $\{H_k : \prod_{t\in\alpha(k)} \mathcal{X}_t \mapsto \mathbb{R}\}_{k=1}^m$, where $\alpha(k) \subseteq \{1,\ldots,n\}$ is the set of variables that H_k depends on.

The factor graph is represented as a bipartite graph where the variables live on one side of the graph and the factors on the other side. A variable is connected to a factor if the factor depends on the variable, so that there is an edge between β_i and H_j if and only if $i \in \alpha(j)$. The optimization problem associated with the factor graph is $\text{Minimize}_{\{\beta_i\in\mathcal{X}_i\}_{i=1}^n} \sum_{k=1}^m H_k\left(\beta_{\alpha(k)}\right)$. As a concrete example, the optimization problem represented by the factor graph shown in Figure 2 is:

$$\min_{\{\beta_i\in\{0,1\}\}_{i=1}^5} H_1(\beta_1,\beta_2,\beta_3) + H_2(\beta_2,\beta_4) + H_3(\beta_4,\beta_5) \tag{3}$$

A naive approach to solving (3) would be to simply enumerate all 2^5 assignments to the variables $\beta_1,\ldots,\beta_5$. However, one can solve this problem efficiently by exploiting the factor graph structure as follows: We first note that β_5 only appears in the factor H_3, so we can rewrite the optimization as

$$\min_{\beta_1,\beta_2,\beta_3,\beta_4} H_1(\beta_1,\beta_2,\beta_3) + H_2(\beta_2,\beta_4) + \min_{\beta_5} H_3(\beta_4,\beta_5)$$

Define $\kappa_3(\beta_4) = \min_{\beta_5} H_3(\beta_4,\beta_5)$ (this function can be evaluated using 4 units of time assuming each function evaluation takes 1 unit of time, by evaluating H_3 for all 4 assignments to its arguments). The problem then reduces to

$$\min_{\beta_1,\beta_2,\beta_3} H_1(\beta_1,\beta_2,\beta_3) + \min_{\beta_4} H_2(\beta_2,\beta_4) + \kappa_3(\beta_4)$$

Again, define $\kappa_2(\beta_2) = \min_{\beta_4} H_2(\beta_2, \beta_4) + \kappa_3(\beta_4)$ (this can again be evaluated in 4 units of time). Then, the problem reduces to

$$\min_{\beta_1,\beta_2,\beta_3} H_1(\beta_1, \beta_2, \beta_3) + \kappa_3(\beta_3)$$

No further simplification is possible since H_1 depends on all the remaining variables. The optimal value can be computed now in 8 units of time (since there are 2^3 possible assignments to the variables).

Thus the global optimum can be computed using $4+4+8 = 16$ units of time as opposed to the 32 units of time taken by the naive brute-force approach. This algorithm generalizes to arbitrary tree-structured factor graphs (factor graphs with no cycles). In Section 3, we formulate the ACOPF (2) as a tree-structured factor graph and show how to exploit factor graph techniques to solve the ACOPF with approximation guarantees.

3 OPF as a Graphical Model

This section shows how to rewrite the OPF problem (2) as a graphical model inference problem. We first define the augmented variables $\beta_i = \left(\mathfrak{v}_i \; P_i \; Q_i\right), i = 1, \ldots, n, \beta_0 = \left(\mathfrak{v}^{ref} \; 0 \; 0\right)$. Note that β_0 is fixed and is only introduced for notational convenience. In terms of the augmented variables β_i, the constraints (2c),(2d) are simply bound constraints on components of β_i. The domain of the variable β_i is defined as $\mathcal{X}_i = [\mathfrak{v}_i^L, \mathfrak{v}_i^U] \times [P_i^L, P_i^U] \times [Q_i^L, Q_i^U]$. Further, using (1a), we can define

$$p\left(\beta_i, \beta_{\mathcal{C}(i)}\right) = P_i - \sum_{k \in \mathcal{C}(i)} \left(P_k - r_k \frac{P_k^2 + Q_k^2}{\mathfrak{v}_k}\right) \tag{4a}$$

$$q\left(\beta_i, \beta_{\mathcal{C}(i)}\right) = Q_i - \sum_{k \in \mathcal{C}(i)} \left(Q_k - x_k \frac{P_k^2 + Q_k^2}{\mathfrak{v}_k}\right) \tag{4b}$$

$$\mathfrak{v}_i(\beta_k) = \mathfrak{v}_k + \left(r_k^2 + x_k^2\right) \frac{\left(P_k^2 + Q_k^2\right)}{\mathfrak{v}_k} - 2\left(P_k r_k + Q_k x_k\right) \tag{4c}$$

For each $i = 0, \ldots, n$, we define

$$H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right) = \tag{5}$$

$$\begin{cases} c_i(s_i) & \text{if } \begin{cases} \left(p_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right), q_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right)\right) \in \mathbf{S}_i \\ \mathfrak{v}_i = \mathfrak{v}_i(\beta_k) \quad \forall k \in \mathcal{C}(i) \end{cases} \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

This is a convenient shorthand that allows to write the constrained optimization problem (2) as an unconstrained problem, thereby simplifying notation. The OPF problem (2) is equivalent to

$$\underset{\{\beta_i \in \mathcal{X}_i\}_{i=1}^n}{\text{Minimize}} \sum_{i=0}^{n} H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right) \tag{7}$$

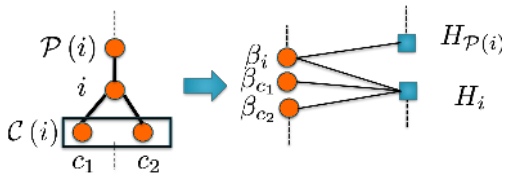


Fig. 3: Transformation from the power network to factor graph: Each variable node i is connected to two factors - $H_{\mathcal{P}(i)}$ and H_i

This corresponds to a graphical model in the factor graph representation [17] where the nodal variables are β_i and the factors correspond to the H_i.

Definition 1 (OPF factor graph) The problem (7) corresponds to a factor graph: The set of variable nodes is

$$\beta_i \in \mathcal{X}_i \quad (i = 1, \ldots, n)$$

and the set of factors is

$$H_i \quad (i = 0, \ldots, n)$$

For $i = 1, \ldots, n$, β_i is connected to $H_i, H_{\mathcal{P}(i)}$ (see Figure 3).

Theorem 1 *The factor graph from definition 1 is a tree. Hence, the problem* (7) *can be solved exactly by a two-pass dynamic programming algorithm.*

Proof Every variable node $\beta_i \quad (i = 1, \ldots, n)$ is connected to two factors $H_{\mathcal{P}(i)}, H_i$. Thus, the total number of edges in the factor graph is $2n$ and the total number of nodes is $2n-1$ (there are factors $H_0, \ldots, H_n$ and variable nodes $\beta_1, \ldots, \beta_n$). Further, the factor graph is connected because variable nodes are connected (Since the variable nodes that are also neighbors in the power network are neighbors 2 hops apart in the factor graph). The factor graph is a connected graph with the number of vertices equal to the number of edges plus one. Therefore, the graph is a tree. The result on efficient inference over a tree factor graph is a standard result [17].

The dynamic-programming (DP) approach is formalized in Algorithm 1. The algorithm works by passing "messages" $\kappa_t(\beta_t)$. Let $\mathrm{Desc}(t)$ denote all the nodes in the subtree rooted at t (the descendants of t, including t itself).

The message denotes the optimal value of the following subproblem of(7):

$$\kappa_t(\beta_t) = \min_{\beta_i \in \mathcal{X}_i : i \in \mathrm{Desc}(t) \setminus \{t\}} \sum_{i \in \mathrm{Desc}(t)} H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right).$$

The messages can be computed recursively starting at the leaves of the tree. For the 5 bus network shown in Figure 4, Algorithm 1 proceeds as follows: At $T = 0$, nodes $3, 4$ send messages κ_3, κ_4 (which are simply equal to H_3, H_4) to

Algorithm 1 DP Algorithm: (optimization implicitly subject to $\beta_i \in \mathcal{X}_i$)

Processed $= \emptyset$
$\kappa_i(\beta_i) = 0, \eta_i(\beta_i) = \emptyset, i = 0, 1 \ldots, n$
while $|\text{Processed}| <= n$ **do**
  Choose $i \in \{k : k \notin \text{Processed}, \mathcal{C}(i) \subseteq \text{Processed}\}$

$$\kappa_i(\beta_i) \leftarrow \min_{\beta_{\mathcal{C}(i)}} H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right) + \sum_{k \in \mathcal{C}(i)} \kappa_k(\beta_k) \tag{8a}$$

$$\eta_i(\beta_i) \leftarrow \operatorname*{argmin}_{\beta_{\mathcal{C}(i)}} H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right) + \sum_{k \in \mathcal{C}(i)} \kappa_k(\beta_k) \tag{8b}$$

  Processed $=$ Processed $\cup \{i\}$
end while
$\left(c^*, \beta_{\mathcal{C}(0)}{}^*\right) \leftarrow \min_{\beta_{\mathcal{C}(0)}} H_0\left(\beta_0, \beta_{\mathcal{C}(0)}\right) + \sum_{k \in \mathcal{C}(0)} \kappa_k(\beta_k)$
Processed $= \{0\}$.
while $|\text{Processed}| \leq n$ **do**
  Choose $i \in \{k : k \notin \text{Processed}, \mathcal{P}(k) \in \text{Processed}\}$
  $\beta^*_{\mathcal{C}(i)} \leftarrow \eta_i\left(\beta^*_i\right)$
  Processed $\leftarrow$ Processed $\cup\, \mathcal{C}(i)$
end while
return (c^*, β^*)

their parent nodes. For brevity, we do not write the constraint $\beta_i \in \mathcal{X}_i$ in the updates below explicitly. At $T = 1$, node 2 computes

$$\kappa_2(\beta_2) = \min_{\beta_3 \in \mathcal{X}_3} H_2(\beta_2, \beta_3) + \kappa_3(\beta_3), \eta_2(\beta_2) = \operatorname*{argmin}_{\beta_3 \in \mathcal{X}_3} H_2(\beta_2, \beta_3) + \kappa_3(\beta_3).$$

At $T = 2$, node 1 computes

$$\begin{aligned}
\kappa_1(\beta_1) &= \min_{\beta_2 \in \mathcal{X}_2, \beta_4 \in \mathcal{X}_4} H_2(\beta_1, \beta_2, \beta_4) + \kappa_2(\beta_2) + \kappa_4(\beta_4) \\
\eta_1(\beta_1) &= \operatorname*{argmin}_{\beta_2 \in \mathcal{X}_2, \beta_4 \in \mathcal{X}_4} H_2(\beta_1, \beta_2, \beta_4) + \kappa_2(\beta_2) + \kappa_4(\beta_4).
\end{aligned}$$

The forward pass ends here, and the backward pass proceeds similarly in the reverse direction:

$$\begin{aligned}
&T = 3 : (\text{OPT}, \beta_1{}^*) = \min_{\beta_1 \in \mathcal{X}_1} \kappa_1(\beta_1) \\
&T = 4 : (\beta_2{}^*, \beta_4{}^*) = \eta_1(\beta_1{}^*) \\
&T = 5 : \beta_3{}^* = \eta_2(\beta_2{}^*).
\end{aligned}$$

4 Finite Algorithm based on Interval Discretization

The algorithm described in the preceding section, as it stands, cannot be implemented on a computer, since it requires representing the functional objects

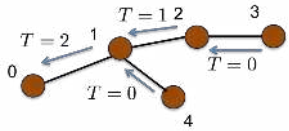


Fig. 4: Transformation of ACOPF to Graphical Model

$\kappa^i(\beta_i)$. A straightforward approach to deal with this would be to discretize the variables β_i to a finite set and allow for some error tolerance on each of the constraints in (2). However, our experiments have indicated that, in order to produce solutions of acceptable accuracy, one needs an intractably fine discretization of β_i resulting in prohibitive computation times, and that an accurate estimation of the error tolerance parameter can be problematic.

Hence, we need an alternative procedure to approximate the infinite dimensional dynamic program. We take the approach of using an interval-based discretization of the power flow variables $\mathfrak{v}_i, P_i, Q_i$ (i.e., each variable can take values in the respective interval). Given the constraints $\beta_i \in \mathcal{X}_i$, we partition the set $\mathcal{X}_i$ into a finite union of interval regions defined by interval constraints on the components of β_i. For any practical OPF problem, $\mathcal{X}_i$ is a compact set (since the bounds on voltages/flows are always finite), so such a decomposition is always possible. Naturally, the computational complexity of the algorithm depends on the number of interval regions. If we fix an interval resolution ϵ (each interval region in our decomposition is made up of interval constraints of width at most ϵ in each variable), the number of interval regions depends on the bounds defining the region $\mathcal{X}_i$. Thus, it is of interest to have as tight bounds as possible on the variables β_i. We describe a procedure to infer tight bounds on the power flow variables using convex relaxations in Section A.3. We use the inferred bounds to redefine $\mathcal{X}_i$ and perform the interval discretization on this refined domain. Then, we perform the dynamic programming update in the space of intervals: For every interval-value of the parent, we look at all possible interval values of the children, and for each combination of intervals, compute the minimum cost solution given the interval constraints. This gives us a lower bound on the message for each interval value of the parent, thus leading to a piecewise-constant lower bound on the message functions $\kappa^i(\beta_i)$. The algorithm is described in Section 4.1.

As the interval discretization gets finer, the relaxation becomes tighter, reducing the errors incurred in the power flow equations due to the relaxation. The errors can be made smaller than ϵ for any $\epsilon > 0$, with the running time polynomial in $\frac{1}{\epsilon}$. These results are formalized in Theorem 2 in Section 4.3.

4.1 Interval Dynamic Programming with Adaptive Refinement

We use Algorithm 1 with the DP update step replaced with a tractable lower bound based on an interval partition of the variable domains. We develop such

a scheme guaranteed to produce a lower bound on the optimal value and an approximately feasible solution. The algorithm is based on a set of operators that will replace the DP update (8):

1 *Interval Partition Operator*: A procedure that take a set and creates a collection of intervals such that every point in the set is in an interval box of size at most ϵ, for some specified tolerance ϵ.
2 *Interval DP Relaxation*: A procedure that takes a set of interval constraints on children of a node i and produces a lower bound on $\kappa^i(\beta_i)$.

We now define these formally.

Definition 2 An interval constraint on a real variable $x \in \mathbb{R}$ is a constraint of the type $a \leq x \leq b$, parameterized by real numbers $a, b, a \leq b$.

Definition 3 An interval region $\mathbb{I}_i$ is a subset of $\mathcal{X}_i$ specified by interval constraints on the variables $\mathfrak{v}_i, P_i, Q_i$:

$$\mathbb{I}_i = \left\{ (\mathfrak{v}_i, P_i, Q_i) : \begin{array}{c} {\mathfrak{v}_i}^L \leq \mathfrak{v}_i \leq {\mathfrak{v}_i}^U \\ {P_i}^L \leq P_i \leq {P_i}^U \\ {Q_i}^L \leq Q_i \leq {Q_i}^U \end{array} \right\}$$

We use $\mathfrak{v}(\mathbb{I}_i) = [{\mathfrak{v}_i}^L, {\mathfrak{v}_i}^U], P(\mathbb{I}_i) = [{P_i}^L, {P_i}^U], Q(\mathbb{I}_i) = [{Q_i}^L, {Q_i}^U]$ to denote the interval constraints in $\mathbb{I}_i$ corresponding to each of the variables,

$$\text{mid}(\mathbb{I}_i) = \left(\frac{{\mathfrak{v}_i}^L + {\mathfrak{v}_i}^U}{2}, \frac{{P_i}^L + {P_i}^U}{2}, \frac{{Q_i}^L + {Q_i}^U}{2} \right)$$

to select the midpoint of an interval region and

$$\text{Rad}(\mathbb{I}_i) = \max\left(|{\mathfrak{v}_i}^L - {\mathfrak{v}_i}^U|, |{P_i}^L - {P_i}^U|, |{Q_i}^L - {Q_i}^U| \right).$$

Definition 4 An interval partition $\mathcal{I}_i$ of a set $X \subseteq \mathcal{X}_i$ is a collection of interval regions satisfying the following conditions:

$$\mathcal{I}_i(t) = \left\{ \beta_i : \begin{array}{c} {\mathfrak{v}_i}^L(t) \leq \mathfrak{v}_i \leq {\mathfrak{v}_i}^U(t) \\ {P_i}^L(t) \leq P_i \leq {P_i}^U(t) \\ {Q_i}^L(t) \leq Q_i \leq {Q_i}^U(t) \end{array} \right\}, t = 1, \ldots, |\mathcal{I}_i| \tag{10a}$$

$$\cup_{t=1}^{|\mathcal{I}_i|} \mathcal{I}_i(t) = X \tag{10b}$$

The number of regions in the partition is denoted by $|\mathcal{I}_i|$. A partition such that $\max_t \text{Rad}(\mathcal{I}_i(t)) \leq \epsilon$ is denoted as $\text{Partition}(\mathcal{X}_i; \epsilon)$.

Definition 5 An interval relaxation of the DP update step (8) is a computational procedure that, given a bus $i \in \{0, \ldots, n\}$ with $\mathcal{C}(i) = \{k_1, \ldots, k_m\}$ and interval regions $\beta_i \in \mathbb{I}_i \subseteq \mathcal{X}_i, \beta_k \in \mathbb{I}_k \subseteq \mathcal{X}_k$ for each $k \in \mathcal{C}(i)$, with

$\max_{k\in\{i\}\cup\mathcal{C}(i)} \mathrm{Rad}\left(\mathbb{I}_k\right) \leq \epsilon$, produces as output an interval region $\mathbb{I}'_i$ and values p_i, q_i such that

$$\exists\{\beta_k \in \mathbb{I}_k\}_{k\in\mathcal{C}(i)}, \beta_i \in \mathbb{I}'_i$$

$$\text{s.t} \begin{cases} \left|p_i - p_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right)\right| & \leq \eta\epsilon \\ \left|q_i - q_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right)\right| & \leq \eta\epsilon \\ \left|\mathfrak{v}_i - \left(\mathfrak{v}_k + \frac{P_k^2+Q_k^2}{\mathfrak{v}_k}\left(r_k^2 + x_k^2\right) - 2\left(P_k r_k + Q_k x_k\right)\right)\right| & \leq \eta\epsilon \quad \forall k \in \mathcal{C}\left(i\right) \end{cases} \tag{11a}$$

$$c_i\left(p_i, q_i\right) \leq \min_{\{\beta_k\in\mathbb{I}_k\}_{k\in\mathcal{C}(i)\cup\{i\}}} H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right) \tag{11b}$$

$$\{\beta_i : \exists\{\beta_k \in \mathbb{I}_k\}_{k\in\mathcal{C}(i)} \text{ s.t } H_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right) < \infty\} \subseteq \mathbb{I}'_i \subseteq \mathbb{I}_i \tag{11c}$$

where η is a constant that depends only on the number M from Assumption 2. We denote this computation as

$$\left(\mathbb{I}'_i, p_i, q_i\right) = \mathrm{PropBound}\left(i, \mathbb{I}_i, \mathbb{I}_{k_1}, \ldots, \mathbb{I}_{k_m}\right).$$

These conditions can be interpreted as follows:

- (11a) states that the relaxation gets tighter as the intervals get smaller. A natural relaxation that satisfies this requirement is to take convex envelopes of the nonlinear terms over the bound constraints defined by the intervals: This is what we use in the concrete implementation described in Section 4.2.2.
- (11b) states that the injections produced by the relaxation step are super-optimal, so that we are guaranteed to get a lower bound on the optimal solution through the DP procedure.
- (11c) states that the bound propagation (which shrinks the interval $\mathbb{I}_i$ to $\mathbb{I}'_i$ using the constraints implicit in H_i) cannot cut off any feasible points.

Given this computational procedure, we can construct a DP-like algorithm where the intractable DP update (8) is replaced with a tractable procedure based on PropBound, thereby producing a lower bound on the message function $\kappa^i\left(\beta_i\right)$. The algorithm is described in Algorithms 2 and 3. The algorithm starts by partitioning the space $\mathcal{X}_i$ using the interval partition operator. For each element of the interval partition, we loop over the pieces in the messages corresponding to each child, and propagate constraints from the children to the parent β_i. If the propagated interval is non-empty (that is, there exists a feasible setting for the parents and children within the interval constraints), the lower bound computed on $c_i\left(p_i, q_i\right)$ is used and a new piece is added to the messages $\kappa^i\left(\beta_i\right), \eta^i\left(\beta_i\right)$. In comparison to the DP Algorithm 1, we also maintain functions p^i, q^i, β^i which store the optimal injections and intervals for every variable computed in the DP procedure.

4.2 Implementation of operators

4.2.1 Interval Discretization Operator

The ϵ-partition operator can be implemented by using a uniform discretization. The bounds $\mathfrak{v}^L, \mathfrak{v}^U, S^L, S^U$ are obtained from the bound tightening procedure (16) described in Section A.3. For the variable $\mathfrak{v}_i$ with bounds ${\mathfrak{v}_i}^L, {\mathfrak{v}_i}^U$, the partition operator will create the intervals

$$\left\{ [{\mathfrak{v}_i}^L, {\mathfrak{v}_i}^L + \epsilon^{\mathfrak{v}}], [{\mathfrak{v}_i}^L + \epsilon^{\mathfrak{v}}, {\mathfrak{v}_i}^L + 2\epsilon^{\mathfrak{v}}], \ldots, \left[{\mathfrak{v}_i}^L + \left\lceil \frac{{\mathfrak{v}_i}^U - {\mathfrak{v}_i}^L}{\epsilon^{\mathfrak{v}}} \right\rceil \epsilon^{\mathfrak{v}} \right] \right\}$$

A similar interval discretization procedure is used for P_i, Q_i.

4.2.2 Interval Relaxation Operator

In order to define the interval relaxation operator, it will be convenient to introduce the square of the current magnitude $\mathfrak{i}_i = \frac{P_i^2 + Q_i^2}{\mathfrak{v}_i}$. This serves to isolate the nonconvexities in the problem and simplify the derivation of convex relaxations. Note also that using the current is natural in view of its edge-invariance: in contrast to power flows, current conserves along any edge (power line).

The interval relaxation operator requires solution of the following problem:

$$\underset{\{\mathfrak{v}_k, P_k, Q_k\}_{k \in \mathcal{C}(i) \cup \{i\}}}{\text{Extremize}} \quad \{\mathfrak{v}_i, P_i, Q_i, c_i(p_i, q_i)\} \tag{12a}$$

$$\text{Subject to } P_i = p_i + \sum_{k \in \mathcal{C}(i)} (P_k - \mathfrak{i}_k r_k) \tag{12b}$$

$$Q_i = q_i + \sum_{k \in \mathcal{C}(i)} (Q_k - \mathfrak{i}_k x_k) \tag{12c}$$

$$\mathfrak{v}_i = \mathfrak{v}_k + \mathfrak{i}_k \left(r_k^2 + x_k^2\right) - 2\left(P_k r_k + Q_k x_k\right), k \in \mathcal{C}(i) \tag{12d}$$

$$\mathfrak{v}_k \mathfrak{i}_k = P_k^2 + Q_k^2, k \in \mathcal{C}(i) \cup \{i\} \tag{12e}$$

$$\mathfrak{v}_k \in \mathfrak{v}(\mathbb{I}_k), P_k \in P(\mathbb{I}_k), Q_k \in Q(\mathbb{I}_k), k \in \{i\} \cup \mathcal{C}(i) \tag{12f}$$

$$(p_i, q_i) \in \mathbf{S}_i \tag{12g}$$

where Extremize means that we both maximize and minimize every term in the set of objectives subject to the constraints specified. Thus, we obtain tighter bounds on the variables $\mathfrak{v}_i, P_i, Q_i$ and a lower bound on the objective given the interval constraints. The nonconvexity in the above problem is due to the constraint (13c) and the possibly nonconvex cost c_i and constraints $\mathbf{S}_i$. To deal with the later, we explicitly enumerate over $\mathbb{I}_i^s(t), t = 1, \ldots, |\mathbf{S}_i|$ so that for each t, the injection constraints and costs are linear. To deal with nonconvexity of (13c), we use convex envelopes of the bilinear and quadratic terms. An abstract version of the problem solved is below (we defer exact details to Section A.2)

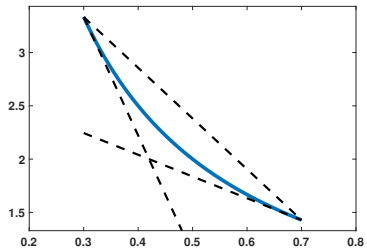


Fig. 5: Convex relaxation of nonlinear constraint $xy = 1$ over the region $x \in [0.3, 0.7], y \in [\frac{1}{0.7}, \frac{1}{0.3}]$: The set of points satisfying the constraint $xy = 1, x \in [.3, .7]$ is plotted in the solid curve (blue). The convex region enclosed by the dashed lines (black) is the feasible region of the convex relaxation.

$$\forall t \in \{1, \ldots, |\mathbf{S}_i|\}$$

$$\underset{p_i, q_i, \{\mathfrak{v}_k, P_k, Q_k\}_{k \in \mathcal{C}(i) \cup \{i\}}}{\text{Extremize}} \quad \{\mathfrak{v}_i, P_i, Q_i, a_i(t)\, p_i + b_i(t)\, q_i + c_i(t)\} \tag{13a}$$

$$\text{Subject to } (12b), (12c), (12d) \tag{13b}$$

$$0 \in \text{Relax}\left(\mathfrak{v}_k \mathfrak{i}_k - \left(P_k^2 + Q_k^2\right), \mathbb{I}_k\right), k \in \mathcal{C}(i) \cup \{i\} \tag{13c}$$

$$\mathfrak{v}_k \in \mathfrak{v}(\mathbb{I}_k), P_k \in P(\mathbb{I}_k), Q_k \in Q(\mathbb{I}_k), k \in \{i\} \cup \mathcal{C}(i) \tag{13d}$$

$$p_i \in [{p_i}^L(t), {p_i}^U(t)], q_i \in [{q_i}^L(t), {q_i}^U(t)] \tag{13e}$$

The relaxations we use depend on the bound constraints $\mathbb{I}_k$ and are denoted as $\text{Relax}\left(\mathfrak{v}_k \mathfrak{i}_k - \left(P_k^2 + Q_k^2\right), \mathbb{I}_k\right)$. A simple example of this kind of relaxation is shown pictorially in Figure 5 for the constraint $xy = 1$. It is easy to see that this relaxation gets tighter as the bound constraints on x get tighter, leading to the property that any feasible solution (x, y) of the relaxation satisfies $|xy - 1| \propto \epsilon$, where ϵ is the size of the interval constraint on x (this is formalized in Lemma 2 in the Appendix Section A.2).

4.3 Analysis of the interval DP algorithm

We now present formal results verifying the correctness, optimality, and feasibility properties of the solutions produced by our DP algorithm.
Before we state our main theorem that provides an approximation guarantee, we note that we can always convert an OPF problem on an arbitrary tree network to a problem on a tree network with maximum degree 3:

Lemma 1 *An OPF problem on an arbitrary tree network with n nodes and maximum degree d can be converted to an OPF problem on a modified tree network with maximum degree 3 and at most nd nodes.*

Algorithm 2 Interval DP update at node i with children (j,k)

$(j,k) \leftarrow \mathcal{C}(i)$
$n_i \leftarrow 0$
$\mathcal{I}_i \leftarrow \text{Partition}(\mathcal{X}_i, \epsilon)$
for $m_i = 1, \ldots, |\mathcal{I}_i|$ **do**
  for $m_j \in 1, \ldots, |\eta^j|$ **do**
    for $m_k \in 1, \ldots, |\eta^k|$ **do**
      $(\mathbb{I}, p, q) \leftarrow \text{PropBound}\left(\mathcal{I}_i(m_i), \beta^j(m_j), \beta^k(m_k)\right)$
      if $\mathbb{I} \neq \emptyset$ **then**
        $n_i \leftarrow n_i + 1, \eta^i(n_i) \leftarrow (m_j, m_k)$
        $\beta^i(n_i) \leftarrow \mathbb{I}, \kappa^i(n_i) \leftarrow c_i(p,q) + \kappa^j(m_j) + \kappa^k(m_k)$
        $p^i(n_i) \leftarrow p, q^i(n_i) \leftarrow q$
      end if
    end for
  end for
end for
return $\eta^i, \kappa^i, p^i, q^i, \beta^i$

Algorithm 3 Interval DP update at leaf node

$n_i \leftarrow 0$
for $m_i \in \left\{1, \ldots, \left\lceil \frac{\mathfrak{v}_i{}^U - \mathfrak{v}_i{}^L}{\epsilon} \right\rceil\right\}$ **do**
  for $t = 1, \ldots, |\mathbf{S}_i|$ **do**
    $n_i = n_i + 1$
    if $[P^L, P^U] \cap [p_i(t)^L, p_i(t)^U] \neq \emptyset, [Q^L, Q^U] \cap [q_i(t)^L, q_i(t)^U] \neq \emptyset$ **then**
      $\mathbb{I}^{\mathfrak{v}} \leftarrow [\mathfrak{v}_i{}^L + (m_i - 1)\epsilon, \min\left(\mathfrak{v}_i{}^L + m_i\epsilon, \mathfrak{v}_i{}^U\right)], \mathbb{I}^P \leftarrow [P^L, P^U] \cap [p_i(t)^L, p_i(t)^U]$
      $\mathbb{I}^Q \leftarrow [Q^L, Q^U] \cap [q_i(t)^L, q_i(t)^U], \eta^i(n_i) \leftarrow \mathbb{I}^{\mathfrak{v}} \times \mathbb{I}^P \times \mathbb{I}^Q$
      $\left(p^i(n_i), q^i(n_i)\right) \leftarrow \underset{p_i \in \mathbb{I}^P, q \in \mathbb{I}^Q}{\mathsf{argmin}} c_i(t) + a_i(t)\, p_i + b_i(t)\, q_i$
      $\kappa^i(n_i) \leftarrow \min_{p_i \in \mathbb{I}^P, q_i \in \mathbb{I}^Q} c_i(t) + a_i(t)\, p_i + b_i(t)\, q_i$
    end if
  end for
end for
return η^i, κ^i

Proof See Appendix Section A.4.

Theorem 2 (Approximate optimality property) *Suppose that assumptions 1 and 2 (see Section 2.2) hold and that the DP Algorithm 1 (with update rule from Algorithm 2) is run on a tree network with maximum degree* 3 *and with* $0 < \epsilon < 1$. *Let* $m_1^*, m_2^*, \ldots, m_n^*$ *be the indices of the variables in the optimal solution. Let* $(\mathfrak{v}_i^*, P_i^*, Q_i^*) = \text{mid}(\eta_i(m_i^*))$ *and* $p_i^* = p^i(m_i^*), q_i^* = q^i(m_i^*)$. *Then, the following guarantees hold:*

1. Approximation guarantee*:* $(\mathfrak{v}^*, P^*, Q^*, p^*, q^*)$ *satisfies each constraint of* (2) *with a bounded error* $\zeta\epsilon$ *where* ζ *is a constant that depends only on* M *(the constant from Assumption 2 in Section 2.2).*
2. Runtime bound*: There is a constant* ζ' *(depending on* M*) such that the algorithm requires at most* $n\zeta'\left(\frac{1}{\epsilon}\right)^5$ *calls to the* PropBound *routine.*

3 Optimality guarantee*: The cost of the solution is bounded as:*

$$\sum_{i=0}^{n} c_i\left(p_i^*, q_i^*\right) \leq \mathrm{OPT}$$

where OPT *is the optimal cost of the original problem* (2)*.*

Thus, we find a super-optimal approximately feasible solution in time linear in the size of the network and polynomial in the error tolerance.

Proof See Appendix Section A.5.

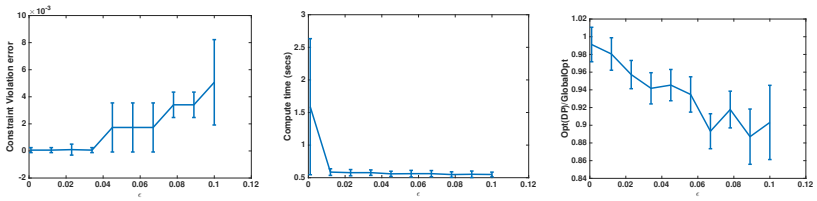
Remark 1 Theorem 2 formalizes the intuition that as we use finer intervals in the Interval DP algorithm, we get closer to the optimal solution in terms of cost, and we get a tighter relaxation as well. The numerical results in Section 5 show that our algorithm often finds the true optimal solutions even with a finite error tolerance.

5 Numerical Illustrations

In this Section, we present numerical tests of our approach on some IEEE benchmark networks - power grids with network topologies and loads that are deemed representative of real power systems. In particular, we use a set of subnetworks of the 56-bus distribution network [2] (based on the IEEE 123 bus distribution feeder network [1]). We create additional subnetworks(14/30/56 bus) by aggregating nodes in the original network. We study discrete load-curtailment problems, which are mixed integer nonconvex nonlinear optimization problems (MINLPs). We analyze a highly overloaded distribution network: a scenario that might arise just before a blackout, or after the loss of a major generator or transmission line. The goal is to curtail (reduce the consumption of) a small number of loads so that the power grid is restored to its normal operating state (bring voltages back to acceptable range). A cost is incurred for curtailing a load (typically proportional to the reduction of load). The total cost is the sum of the load-shedding costs plus a generation cost at the substation (bus 0). The formal statement of the problem is as follows:

$$\begin{aligned}
&\underset{\sigma,\mathfrak{v},S}{\text{Minimize}} \quad c_0\left(p_0, q_0\right) + \sum_{i=1}^{n} c_i\left(\sigma_i\right) \\
&\text{Subject to } (1\text{b}), (1\text{a}), (1\text{c}) \\
&\qquad p_i = p_i^{nom}(1-\sigma_i) + p_i^{red}\sigma_i, q_i = q_i^{nom}(1-\sigma_i) + q_i^{red}\sigma_i \\
&\qquad \sigma_i \in \{0,1\}, \mathfrak{v}_i^L \leq \mathfrak{v}_i \leq \mathfrak{v}_i^U
\end{aligned}$$

The values p_i^{nom}, q_i^{nom} denote the nominal values of the real and reactive demands at the bus i. p_i^{red}, q_i^{red} denote the reduced (curtailed) values of the loads. $\sigma_i \in \{0,1\}$ denote the curtailment decision ($\sigma = 1$ denotes curtailment). Curtailment of loads incurs a cost $c_i\left(\sigma_i\right)$.



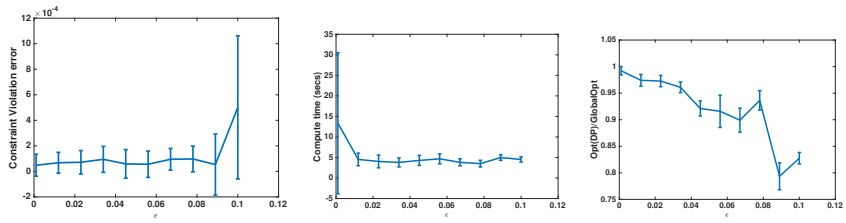
(a) Constraint Violation (b) Computation time (c) Superoptimality ratio

Fig. 6: 14 bus network

We run the DP algorithm (Algorithm 1 with update step from Algorithms 2,3) on our three test cases (14,30, and 56 buses). Ratio of DP optimum to true optimum/upper bound, maximum constraint violation and CPU time are studied as functions of ϵ. To ensure that the results are not artifacts of the particular test cases used, the results averaged over 50 instances of the problem generated by perturbing the loads in the original problem randomly by up to 10% at each bus. We show both the mean and standard deviations of each quantity. We summarized our observations below.

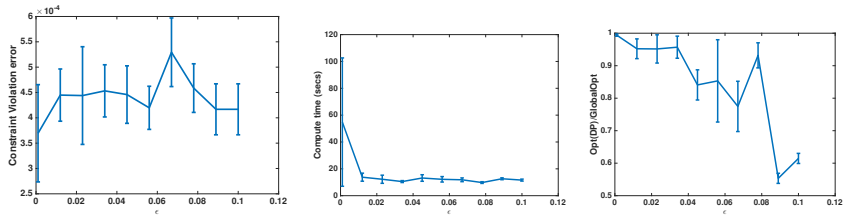
- Since our approach is based on a relaxation, it may produce infeasible solutions. However, as the radius ϵ of the interval discretization reduces, the degree of infeasibility, as measured by the maximum constraint violation, decreases. We quantify this dependence by taking the optimal configuration produced by the DP algorithm and solving the power flow equations (using Newton's method). We then examine if the power flow solution satisfies the bound constraints on voltages. Otherwise, we compute the maximum violation, as shown in Figures 6a,7a, and 8a. The results show that near-feasible super-optimal solutions are found by our DP algorithm consistently.
- The other parameter of interest is the degree of super-optimality, which measures how close the optimal cost of the solution found by the DP is to the true optimal cost of the original problem (2). For the 14 bus network, it is feasible to find the true optimal cost using a brute-force search. However, for larger networks, we rely on the BONMIN solver to get a feasible solution and bound the optimality gap. The results shown in Figures 6b,7b,8b prove that when ϵ is sufficiently small, the DP algorithm optimum is within .99 of the true optimum. Note that the non-monotonic behavior of the optimum is due to the fact that we use an adaptive discretization. Even though our interval discretization gets tighter as ϵ gets smaller, it is not guaranteed to be a strict refinement of the intervals corresponding to a larger ϵ.
- Finally, we study the dependence of the running time of the algorithm on ϵ, in Figures 6c,7c,8c. The running time of the algorithm grows as ϵ decreases, but the plots show that a good optimality ratio and an acceptable error can be achieved with a fairly small running time of several seconds.

We also compared our algorithm to other available MINLP solvers: (1) BONMIN [6] is a solver guaranteed to find global optima of convex MINLPs. It can



(a) Constraint Violation (b) Computation time (c) Superoptimality ratio

Fig. 7: 40 bus network



(a) Constraint Violation (b) Computation time (c) Superoptimality ratio

Fig. 8: 56 bus network

be used as a heuristic solver for nonconvex MINLPs but with no guarantees on global optimality. (2) COUENNE (`http://www.coin-or.org/Couenne/`) is a solver guaranteed to find global optima of nonconvex MINLPs based on a spatial branch and bound algorithm. We access both solvers through the Julia interface available via the JuMP package [11]. For the problems we studied, COUENNE failed to converge within an acceptable time limit (1 hr) so we do not report results from COUENNE. The BONMIN results are summarized in Table 1. While the BONMIN solver was faster in our experiments, it is a heuristic solver, i.e, it is not guaranteed to find a globally optimal solution. Indeed, BONMIN indeed fail to find optimal solutions in the 56 bus network. In contrast, our DP approach always succeeds in finding a global optimum (see table 2), although for the 56 bus network, it requires a very small ϵ which drives up the running time of the algorithm to $1170s$. The reason for this behavior is that there is a super-optimal solution that violates the voltage constraints by only $10^{-4}\%$. This means that the discretization of voltages has to be smaller than this for the solver to be able to recognize infeasibility of this solution, and find the true global optimum.

6 Conclusions

We have presented a novel dynamic programming based algorithm for solving optimal power flow over tree networks. Preliminary experiments have indicated that the approach is promising and that it can solve difficult mixed

Table 1: Performance of BONMIN solver on discrete load-control: Optimality gap computed using lower bound from DP algorithm.

Test Case	Optimality Gap	Computation Time
14 bus	0%	.1s
30 bus	0%	.2s
56 bus	0.1%	.5s

Table 2: Performance of DP solver on discrete load-control.

Solver	Test Case	Optimality Gap	Computation Time
DP	14 bus	0%	1s
DP	30 bus	0%	40s
DP	56 bus	0%	1170s

integer NLP problems arising in power systems. We note that these conclusions are still preliminary and further work needs to be done to carefully test and validate the performance of this approach across a range of test problems. Overall, we envision that graphical models will be a powerful paradigm for analysis and control of power systems and other infrastructure networks. We plan to explore the following concrete directions in future work:
(1) Extension to probabilistic inference problems: As solar penetration increases, the notion of security analysis (making sure that all voltages, flows, currents are within bounds) will need to be phrased in a probabilistic manner. For example, given a joint spatial distribution of solar generation at various points in the network, compute the probability that a given physical quantity (voltage/current/flow) deviates beyond its acceptable bounds. This problem can be phrased as the sum-product analog of the problem solved here.
(2) Extensions to loopy graphs: There are several possibilities for extending the algorithms presented here to loopy graphs. The most straightforward extensions would be based on junction trees [16] (cluster nodes into supernodes to form a tree) or on cutset conditioning [10] (fix values of variables on a cutset of the graph, and given for each fixed value, use inference on the remaining tree-structured graph). Another route is to use loopy belief propagation or the corresponding Linear Programming relaxation of the inference problem [16], and subsequent hierarchies of relaxations, in the spirit of [24][14].
(3) Parameterized messages: We represented messages with piecewise-constant approximations. Another option is to use a parameterized representation (polynomial/piecewise linear/piecewise polynomial for ex). An interesting related development is [12], where the authors show that belief propagation with piecewise-linear messages is guaranteed to find the global optimum of a certain special minimum cost flow problem in polynomial time. Extending this to ACOPF is another promising direction for future work.

References

1. https://ewh.ieee.org/soc/pes/dsacom/testfeeders/
2. https://github.com/saveriob/approx-pf
3. Baran, M.E., Wu, F.F.: Optimal capacitor placement on radial distribution systems. Power Delivery, IEEE Transactions on **4**(1), 725–734 (1989)
4. Bienstock, D., Munoz, G.: Lp approximations to mixed-integer polynomial optimization problems. ArXiv e-prints (2015)
5. Bienstock, D., Verma, A.: Strong np-hardness of ac power flows feasibility. arXiv preprint arXiv:1512.07315 (2015)
6. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., et al.: An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization **5**(2), 186–204 (2008)
7. Bose, S., Gayme, D., Chandy, K., Low, S.: Quadratically constrained quadratic programs on acyclic graphs with application to power flow. Control of Network Systems, IEEE Transactions on **2**(3), 278–287 (2015). DOI 10.1109/TCNS.2015.2401172
8. Carpentier, J.: Contribution to the economic dispatch problem. Bulletin de la Societe Francoise des Electriciens **3**(8), 431–447 (1962)
9. Coffrin, C., Hijazi, H.L., Van Hentenryck, P.: Strengthening convex relaxations with bound tightening for power network optimization. In: Principles and Practice of Constraint Programming, pp. 39–57. Springer (2015)
10. Dechter, R.: Constraint processing. Morgan Kaufmann (2003)
11. Dunning, I., Huchette, J., Lubin, M.: JuMP: A modeling language for mathematical optimization. arXiv:1508.01982 [math.OC] (2015). URL `http://arxiv.org/abs/1508.01982`
12. Gamarnik, D., Shah, D., Wei, Y.: Belief propagation for min-cost network flow: Convergence and correctness. Operations research **60**(2), 410–428 (2012)
13. Gan, L., Li, N., Topcu, U., Low, S.H.: Exact convex relaxation of optimal power flow in radial networks. Automatic Control, IEEE Transactions on **60**(1), 72–87 (2015)
14. Johnson, J.K.: Convex relaxation methods for graphical models: Lagrangian and maximum entropy approaches. Ph.D. thesis, Massachusetts Institute of Technology (2008)
15. Kearns, M., Littman, M.L., Singh, S.: Graphical models for game theory. In: Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, pp. 253–260. Morgan Kaufmann Publishers Inc. (2001)
16. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
17. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. Information Theory, IEEE Transactions on **47**(2), 498–519 (2001)
18. Lavaei, J., Tse, D., Zhang, B.: Geometry of power flows and optimization in distribution networks (2013)
19. Lehmann, K., Grastien, A., Van Hentenryck, P.: Ac-feasibility on tree networks is np-hard. Power Systems, IEEE Transactions on **31**(1), 798–801 (2016). DOI 10.1109/TPWRS.2015.2407363
20. Low, S.H.: Convex relaxation of optimal power flow, part i: Formulations and equivalence. arXiv preprint arXiv:1405.0766 (2014)
21. Low, S.H.: Convex Relaxation of Optimal Power Flow, Part II: Exactness. ArXiv e-prints (2014)
22. Pai, M., Chatterjee, D.: Computer techniques in power system analysis. McGraw-Hill Education (India) (2014)
23. Sojoudi, S., Lavaei, J.: Exactness of semidefinite relaxations for nonlinear optimization problems with underlying graph structure. SIAM Journal on Optimization **24**(4), 1746–1778 (2014)
24. Sontag, D.A.: Approximate inference in graphical models using lp relaxations. Ph.D. thesis, Massachusetts Institute of Technology (2010)
25. Vittal, V., Bergen, A.R.: Power systems analysis. Prentice Hall (1999)
26. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning **1**(1-2), 1–305 (2008)

A Appendix

A.1 Convex Envelopes of quadratic and bilinear terms

The nonlinearities appearing in power flow are of the form x^2 or xy for some variables x, y. We use the following convex envelopes as relaxations of these nonlinear terms:

$$\mathrm{SqRel}\left(y, [y^L, y^U]\right) = \left\{ x : \begin{array}{c} x \geq y^2 \\ x \leq \left(y^L + y^U\right) y - y^L y^U \end{array} \right\} \tag{14a}$$

$$\mathrm{McCormick}\left(y, z, [y^L, y^U], [z^L, z^U]\right) = \left\{ x : \begin{array}{c} x \geq y^L z + z^L y - y^L z^L \\ x \geq y^U z + z^U y - y^U z^U \\ x \leq y^L z - z^U y + y^L z^U \\ x \leq z^L y - y^U z + z^L y^U \end{array} \right\} \tag{14b}$$

A.2 Interval DP Relaxation

In this section, we describe a concrete implementation of the interval relaxation procedure (13):

$$\underset{s_i, \{S_k, \mathfrak{v}_k, \mathrm{Sq}_k^P, , \mathrm{Sq}_k^P, \mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}}\}_{k \in \mathcal{C}(i) \cup \{i\}}}{\text{Extremize}} \quad \{\mathfrak{v}_i, P_i, Q_i, a_i(t) p_i + b_i(t) q_i + c_i(t)\} \tag{15a}$$

Subject to

$$P_i = p_i + \sum_{k \in \mathcal{C}(i)} (P_k - \mathfrak{i}_k r_k) \tag{15b}$$

$$Q_i = q_i + \sum_{k \in \mathcal{C}(i)} (Q_k - \mathfrak{i}_k x_k) \tag{15c}$$

$$\mathfrak{v}_i = \mathfrak{v}_k + \mathfrak{i}_k \left(r_k^2 + x_k^2\right) - 2\left(P_k r_k + Q_k x_k\right), k \in \mathcal{C}(i) \tag{15d}$$

$$\sqrt{P_k^2 + Q_k^2} \leq \sqrt{\mathfrak{v}_k \mathfrak{i}_k} \quad k \in \mathcal{C}(i) \cup \{i\} \tag{15e}$$

$$\mathrm{Sq}_k^P + \mathrm{Sq}_k^Q = \mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}} \quad k \in \mathcal{C}(i) \cup \{i\} \tag{15f}$$

$$\mathrm{Sq}_k^P \in \mathrm{SqRel}\left(P_k, P(\mathbb{I}_k)\right) \quad k \in \mathcal{C}(i) \cup \{i\} \tag{15g}$$

$$\mathrm{Sq}_k^Q \in \mathrm{SqRel}\left(Q_k, Q(\mathbb{I}_k)\right) \quad k \in \mathcal{C}(i) \cup \{i\} \tag{15h}$$

$$\mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}} \in \mathrm{McCormick}\left(\mathfrak{v}_k, \mathfrak{i}_k, \mathfrak{v}(\mathbb{I}_k), \mathfrak{i}(\mathbb{I}_k)\right) \quad k \in \mathcal{C}(i) \cup \{i\} \tag{15i}$$

$$\mathfrak{v}_k \in \mathfrak{v}(\mathbb{I}_k), \mathfrak{i}_k \in \mathfrak{i}(\mathbb{I}_k), P_k \in P(\mathbb{I}_k), Q_k \in Q(\mathbb{I}_k), k \in \{i\} \cup \mathcal{C}(i) \tag{15j}$$

$$p_i \in [{p_i}^L(t), {p_i}^U(t)], q_i \in [{q_i}^L(t), {q_i}^U(t)] \tag{15k}$$

This requires solution of a small number of SOCPs within each DP update (specifically 10 SOCPs in $6d$ variables where d is the maximum degree of a node in the tree - note that we can always choose $d \leq 2$ by modifying the original problem as in lemma 1). Note also that as the intervals $\mathbb{I}_k$ get smaller, the relaxation gets tighter - this is formalized in the lemma below:

Lemma 2 *The relaxation defined by* (15) *is a valid interval relaxation and satisfies the conditions of the definition 5.*

Proof Through this proof, we use $\eta(M)$ to refer to some constant that depends on the number M from assumption 2. Properties (11b),(10b) are obvious since (15) is a valid relaxation of the problem (12). The property (10a) follows from the tightness of the McCormick relaxation. If $\mathbb{I}_i, (\mathbb{I}_k)_{k \in \mathcal{C}(i)}$ are of radius at most ϵ, we know that

$$\mathfrak{v}_k{}^L \mathfrak{i}_k + \mathfrak{i}_k{}^L \mathfrak{v}_k - \mathfrak{v}_k{}^L \mathfrak{i}_k{}^L \leq \mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}} \leq \mathfrak{v}_k{}^U \mathfrak{i}_k + \mathfrak{i}_k{}^L \mathfrak{v}_k - \mathfrak{v}_k{}^U \mathfrak{i}_k{}^L$$

so that the range of $\mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}}$ is of size at most

$$\left(\mathfrak{v}_k{}^U - \mathfrak{v}_k{}^L\right)\left(\mathfrak{i}_k - \mathfrak{i}_k{}^L\right) \leq \left(\mathfrak{v}_k{}^U - \mathfrak{v}_k{}^L\right)\left(\mathfrak{i}_k{}^U - \mathfrak{i}_k{}^L\right) \leq \eta\epsilon$$

since $\mathfrak{i}_k$ has upper and lower bounds depending on the problem data. Thus, we know that $\mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}}$ is at most $\eta(M)\epsilon$ away from $\mathfrak{v}_k\mathfrak{i}_k$. Similarly, Sq^{P_k} is at most $\eta(M)\epsilon$ away from P_k^2, Sq^{Q_k} is at most $\eta(M)\epsilon$ away from Q_k^2. Combining these results, we get the that (10a) holds.

A.3 Bound-tightening procedure

We use a scheme similar to the one described in [9] that infers bounds on the variables $\mathfrak{v}_i, P_i, Q_i\mathfrak{i}_i$ given the constraints in the ACOPF problem (8) . Let $\mathrm{Conv}(\mathbf{S}_i)$ denote the convex hull of the set $\mathbf{S}_i$. We use a convex relaxation of the constraints (2) that depends on the variable bounds. We then iterate this procedure where we use a relaxation to infer tighter bounds, and then tighten the relaxation using the inferred bounds. In practice, we find that the procedure converges in a few iterations typically to a stable set of bounds.

The nonconvex constraint $P_k^2 + Q_k^2 = \mathfrak{v}_k\mathfrak{i}_k$ can be relaxed to a convex constraint: $\sqrt{P_k^2 + Q_k^2} \leq \sqrt{\mathfrak{i}_k\mathfrak{v}_k}$. This can be tightened by replacing the nonlinear terms in the equation $(P_k)^2 + (Q_k)^2 = \mathfrak{v}_k\mathfrak{i}_k$ with their McCormick envelopes. Plugging all this into a single formulation, we obtain:

$$\underset{\mathfrak{v},\mathfrak{i},S,s,\mathrm{Sq}^P,\mathrm{Sq}^Q,\mathrm{Prod}^{\mathfrak{v},\mathfrak{i}}}{\text{Extremize}} \quad \{\mathfrak{v}_i, \mathfrak{i}_i, P_i, Q_i\}_{i=1}^n \tag{16a}$$

$$\text{Subject to } S_i - \sum_{k\in\mathcal{C}(i)} (S_k - \mathfrak{i}_k z_k) \in \mathrm{Conv}(\mathbf{S}_i), i \in \{0,\ldots,n\} \tag{16b}$$

$$\mathfrak{v}_i = \mathfrak{v}_k + \mathfrak{i}_k|z_k|^2 - 2(P_k r_k + Q_k x_k), i \in \{0,\ldots,n\}, k \in \mathcal{C}(i) \tag{16c}$$

$$\sqrt{P_k^2 + Q_k^2} \leq \sqrt{\mathfrak{v}_k\mathfrak{i}_k}, k \in \{1,\ldots,n\} \tag{16d}$$

$$\mathrm{Sq}_k^P + \mathrm{Sq}_k^Q = \mathrm{Prod}_k^{\mathfrak{v},\mathfrak{i}} \quad k \in \{1,\ldots,n\} \tag{16e}$$

$$\mathrm{Sq}_k^P \in \mathrm{SqRel}\left(P_k, [P_k{}^L, P_k{}^U]\right), k \in \{1,\ldots,n\} \tag{16f}$$

$$\mathrm{Sq}_k^Q \in \mathrm{SqRel}\left(Q_k, [Q_k{}^L, Q_k{}^U]\right), k \in \{1,\ldots,n\} \tag{16g}$$

$$\mathrm{Prod}_k^{\mathfrak{v},I} \in \mathrm{McCormick}\left(\mathfrak{v}_k, \mathfrak{i}_k, [\mathfrak{v}_k{}^L, \mathfrak{v}_k{}^U], [\mathfrak{i}_k{}^L, \mathfrak{i}_k{}^U]\right), k \in \{1,\ldots,n\} \tag{16h}$$

Each minimum/maximum value involves solving a Second Order Cone Program (SOCP) and can be done in parallel over the variables involved. This entire procedure can be viewed as a mapping:

$$\left(\mathfrak{v}^L\ \mathfrak{v}^U\ S^L\ S^U\ \mathfrak{i}^L\ \mathfrak{i}^U\right)_t \mapsto \left(\mathfrak{v}^L\ \mathfrak{v}^U\ S^L\ S^U\ \mathfrak{i}^L\ \mathfrak{i}^U\right)_{t+1}$$

We iterate this mapping until there the improvement in bounds is smaller than some threshold. The obtained bounds are used to redefine the domains $\mathcal{X}_i$ for each of the variables.

A.4 Proof of Lemma 1

We describe a transformation that takes a node i with m children and adds at most $r = \lceil \log_2(m) \rceil$ additional buses to create a new network where each node has at most 2 children. We add children in "levels" $p = 1,\ldots,r$: At level 1, we add children c_{10}, c_{11} connected to

bus i by 0-impedance transmission lines. We have the following constraints between i and its children $\mathcal{C}(i)' = \{c_{10}, c_{11}\}$:

$$S_i = s_i + \sum_{k \in \mathcal{C}(i)'} S_k$$
$$\mathfrak{v}_i = \mathfrak{v}_k, k \in \mathcal{C}(i)'$$
$$\mathfrak{i}_k \mathfrak{v}_k = |S_k|^2, k \in \mathcal{C}(i)'$$

At any level $p \leq r$, all nodes are of the form $c_{i_1 \ldots i_p}$. We add its children $\mathcal{C}\left(c_{i_1 \ldots i_p}\right) = \{c_{i_1 \ldots i_p 0}, c_{i_1 \ldots i_p 1}\}$ connected to it by 0-impedance lines with the constraints:

$$S_{c_{i_1 \ldots i_p}} = \sum_{k \in \mathcal{C}\left(c_{i_1 \ldots i_p}\right)} S_k$$
$$\mathfrak{v}_{c_{i_1 \ldots i_p}} = \mathfrak{v}_k, k \in \mathcal{C}\left(c_{i_1 \ldots i_p}\right)$$
$$\mathfrak{i}_k \mathfrak{v}_k = |S_k|^2, k \in \mathcal{C}\left(c_{i_1 \ldots i_p}\right)$$

At the final level $p = r - 1$, every node is of the form $c_{i_1 \ldots i_{r-1}}$ and its children are picked from the set of original children $\mathcal{C}(i)$. One way of doing this is to assign children in order: $\mathcal{C}(c_{10\ldots 0}) = \{c_1, c_2\}, \mathcal{C}(c_{10\ldots 1}) = \{c_3, c_4\}, \ldots$. Then, we add the balance equations:

$$S_{c_{i_1 i_2 \ldots i_{r-1}}} = \sum_{k \in \mathcal{C}\left(c_{i_1 i_2 \ldots i_{r-1}}\right)} (S_k)$$
$$\mathfrak{v}_i = \mathfrak{v}_k, k \in \mathcal{C}(i)'$$
$$\mathfrak{i}_k \mathfrak{v}_k = |S_k|^2, k \in \mathcal{C}(i)'$$

Adding the power balance equations at all the intermediate buses, we recover the original power balance condition

$$s_i = \sum_{k \in \mathcal{C}(i)} (S_k - z_k \mathfrak{i}_k)$$

Further, we have that $\mathfrak{v}_i = \mathfrak{v}_{c_{i_1 i_2 \ldots i_p}}$ for every $1 \leq p \leq r - 1$.

A.5 Proof of Theorem 2

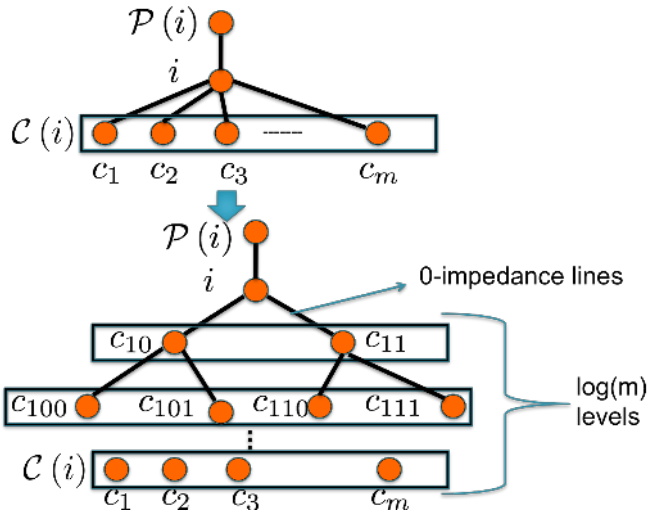
Proof Through this proof, we will use $\zeta(M)$ to denote an arbitrary function of M. The proof of the algorithm breaks down to three key statements:

1 The size of the messages $|\eta^i|$ is bounded by $\frac{\zeta(M)}{\epsilon^3}$.
2 For each message, $\text{Rad}\left(\beta^i(t)\right) \leq \zeta(M)\epsilon, t = 1 \ldots, |\beta^i|$.
3 In the interval DP update Algorithm 2 makes at most $\frac{\zeta(M)}{\epsilon^5}$ calls to the PropBound routine.

Proof of 1,2

For the leaf nodes, the size of the messages is bounded by $|\mathbf{S}_i| \left(\frac{\mathfrak{v}_i{}^U - \mathfrak{v}_i{}^L}{\epsilon} + 1\right) \leq \frac{\zeta(M)}{\epsilon}$ (since $\mathfrak{v}_i^U \leq M, \mathfrak{v}_i{}^L \geq \frac{1}{M}, |\mathbf{S}_i| \leq M$). Since $0 < \epsilon < 1$, this is smaller than $\frac{\zeta(M)}{\epsilon^3}$. Further, since $p_i{}^U(t) - p_i(t), q_i{}^U(t) - q_i(t) \leq \frac{1}{M}$, we know that $\text{Rad}\left(\beta^i(t)\right) \leq \epsilon$ for each $t = 1, \ldots, |\beta^i|$. For non-leaf node, the size of the messages is bounded by the size of $|\text{Partition}(\mathcal{X}_i, \epsilon)| \leq \frac{\zeta(M)}{\epsilon^3}$. Further, since $\beta^i(t) \subset \mathbb{I}_i \in \text{Partition}(\mathcal{X}_i, \epsilon)$, we know that $\text{Rad}\left(\beta^i(t)\right) \leq \epsilon$.

Thus, using (11), the error of the PF equations $|p_i - p_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right)|, |q_i - q_i\left(\beta_i, \beta_{\mathcal{C}(i)}\right)|, |\mathfrak{v}_i - \mathfrak{v}_i(\beta_k)|$ can be bounded by $\zeta(M)\epsilon$. This proves claim 1 of the theorem. Further, since at each step we propagate all interval regions consistent with at least one interval value of the



child, the optimal solution to the original problem is feasible for the interval relaxation. Thus, for each we have that $c_i\left(p_i^*, q_i^*\right) \leq c_i\left(p_i^{OPT}, q_i^{OPT}\right)$. Adding this over all i gives us claim 2 of the theorem.

Finally, we show that the DP update can be implemented with $\frac{\zeta(M)}{\epsilon^5}$ calls to the PropBound routine. In the loops in algorithm 1, we are implicitly looping over possible interval values of $\mathfrak{v}_i, P_i, Q_i, \mathfrak{v}_j, P_j, Q_j, \mathfrak{v}_k, P_k, Q_k$. However, these variables are linked by the constraints:

$$P_i = p_i + \left(P_k - r_k \frac{P_k^2 + Q_k^2}{\mathfrak{v}_k}\right) + \left(P_j - r_j \frac{P_j^2 + Q_j^2}{\mathfrak{v}_j}\right)$$

$$Q_i = q_i + \left(Q_k - x_k \frac{P_k^2 + Q_k^2}{\mathfrak{v}_k}\right) + \left(Q_j - x_j \frac{P_j^2 + Q_j^2}{\mathfrak{v}_j}\right)$$

$$\mathfrak{v}_i = \mathfrak{v}_k + \left(r_k^2 + x_k^2\right) \frac{\left(P_k^2 + Q_k^2\right)}{\mathfrak{v}_k} - 2\left(P_k r_k + Q_k x_k\right)$$

$$\mathfrak{v}_i = \mathfrak{v}_j + \left(r_j^2 + x_j^2\right) \frac{\left(P_j^2 + Q_j^2\right)}{\mathfrak{v}_j} - 2\left(P_j r_j + Q_j x_j\right)$$

Thus, if p_i, q_i are fixed, the 9 variables are constrained to lie on a 5-dimensional manifold (since there are 4 non-redundant . This suggests that we only need to do an exhaustive search over a 5-dimensional space rather than a 9 dimensional space.

Suppose $p_i \in [{p_i}^L(t), {p_i}^U(t)], q_i \in [{q_i}^L(t), {q_i}^U(t)]$ and we fix particular interval values (of radius smaller than ϵ) for variables $\mathfrak{v}_i, P_k, Q_k, P_j, Q_j$. Then, from the first two equations, we know that P_i, Q_i must lie in an interval of size $\zeta(M)\epsilon$ (since p_i, q_i lie in intervals of size $\zeta(M)$). Thus, we need to loop over at most $\zeta(M)$ possible values of P_i, Q_i. Similarly, if $\mathfrak{v}_i, P_k, Q_k$ are fixed to interval values of radius ϵ, the third equation says that $\mathfrak{v}_k$ must lie in an interval of size $\zeta(M)\epsilon$, and similarly if $\mathfrak{v}_i, P_j, Q_j$ are fixed to intervals of radius of ϵ, $\mathfrak{v}_j$ must lie in an interval of size $\zeta(M)\epsilon$. Thus, we need to loop over at most $\zeta(M)$ possible values of $\mathfrak{v}_j, \mathfrak{v}_k, P_i, Q_i$ once the values of $\mathfrak{v}_i, P_k, Q_k, P_j, Q_j$ are fixed to intervals of radius ϵ. Finally, the total number of loops in algorithm 2 is at most $\frac{\zeta(M)}{\epsilon^5}$. Adding this over all nodes of the network, we get the third claim of Theorem 2.