Machine Learning for Solving Optimal Power Flow Problems

Enming Liang[a], Minghua Chen[a,b] and Steven Low[c]

[a]Department of Data Science, City University of Hong Kong

[b]School of Data Science, The Chinese University of Hong Kong (SZ)

[c]Department of Computing & Mathematical Sciences, Caltech

香 港 中 文 大 學 (深 圳)
The Chinese University of Hong Kong, Shenzhen

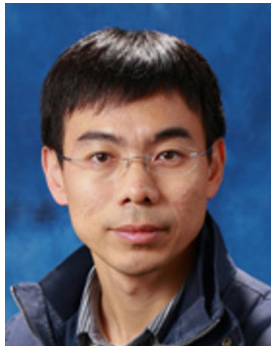
Caltech

Acknowledgements

Xiang Pan
(CUHK; Tencent)

Tianyu Zhao
(CUHK; Lenovo Research)

Min Zhou
(CityU)



Shengyu Zhang
(Tencent)

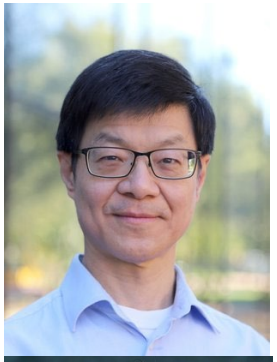
Wanjun Huang
(CityU; Beihang Univ)

Outline

- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems
- Open issues and potential directions



Outline

1. OPF: basic formulation (10 mins)
2. OPF: unbalanced radial network (15 mins)
3. Solution approaches (10 mins)
4. New challenge: uncertainty (15 mins)

$\Longrightarrow$ machine learning for OPF

Outline

1. OPF: basic formulation (10 mins)
   - Single-phase in BIM
   - OPF as QCQP
   - Semidefinite relaxation
2. OPF: unbalanced radial network (15 mins)
3. Solution approaches (10 mins)
4. New challenge: uncertainty (15 mins)

$\Longrightarrow$ machine learning for OPF

Single-phase OPF

Optimal power flow (OPF) is fundamental because it underlies numerous power system applications

- Unit commitment, optimal dispatch, state estimation, contingency analysis, voltage control, …

OPF is a constrained optimization problem

$$\min_{u,x} \quad c(u,x) \qquad \text{s.t.} \qquad f(u,x) = 0,\ g(u,x) \leq 0$$

Single-phase OPF

Optimal power flow (OPF) is fundamental because it underlies numerous power system applications

- Unit commitment, optimal dispatch, state estimation, contingency analysis, voltage control, …

OPF is a constrained optimization problem

$$\min_{u,x} \quad c(u,x) \qquad \text{s.t.} \qquad f(u,x)=0,\ g(u,x)\le 0$$

- Control u : generation commitment, generation set points, transformer taps, EV charging levels, inverter reactive power, …
- Network state x : voltages, line currents, power flows, …
- Cost function $c(u,x)$: generation cost, voltage deviation, power loss, user disutility, …
- Equality constraint $f(u,x)=0$: power flow equations, …
- Inequality constraint $g(u,x)\le 0$: operation constraints, e.g., generation/consumption limits, voltage limits, line limits, security constraints, …

Explain each in turn: var (u,x), cost c, constraints f, g

Single-phase OPF

Setup

Network: $G := (\overline{N}, E)$ with $N+1$ buses in $\overline{N} := \{0,1,\ldots,N\}$ and M lines in E

- Line $(j,k) \in E$: characterized by $\left(y^s_{jk}, y^m_{jk}\right) \in \mathbb{C}^2$ and $\left(y^s_{kj}, y^m_{kj}\right) \in \mathbb{C}^2$

Assume WLOG

- Single-phase devices: voltage sources and power sources only
- Each bus has a single device with $\left(s_j, V_j\right)$

Formulate the simplest OPF to study general computational properties

Single-phase OPF

Variable (u, x), cost function $c(u, x)$

Optimization variable: $(s, V) := \left(s_j, V_j, j \in \overline{N}\right)$

- Represents voltage sources $V_j \in \mathbb{C}$ and power sources $s_j \in \mathbb{C}$

Cost function $C(s, V)$

- Fuel cost : $C(s, V) := \sum_{j:\text{gens}} c_j \operatorname{Re}(s_j)$
- Total real power loss: $C(s, V) := \sum_{j} \operatorname{Re}(s_j)$

Single-phase OPF

Equality constraint $f(u, x) = 0$

Power flow equations in BIM

- Equality constraints on (V, s)

$$s_j = \sum_{k:j\sim k} S_{jk}(V) := \sum_{k:j\sim k} \bar{y}^s_{jk} \left(|V_j|^2 - V_j \bar{V}_k \right) + \bar{y}^m_{jj} |V_j|^2, \qquad j \in \overline{N}$$

where $y^m_{jj} := \sum_{k:j\sim k} y^m_{jk}$

Single-phase OPF

Equality constraint $f(u, x) = 0$

Power flow equations in BIM

- Equality constraints on (V, s)

$$s_j = \sum_{k:j\sim k} S_{jk}(V) := \sum_{k:j\sim k} \bar{y}^s_{jk}\left(|V_j|^2 - V_j\bar{V}_k\right) + \bar{y}^m_{jj}|V_j|^2, \qquad j \in \overline{N}$$

where $y^m_{jj} := \sum_{k:j\sim k} y^m_{jk}$

- Derivation:

$$I_{jk}(V) := y^s_{jk}(V_j - V_k) + y^m_{jk}V_j$$

$$S_{jk}(V) := V_j\bar{I}_{jk}(V) := \bar{y}^s_{jk}\left(|V_j|^2 - V_j\bar{V}_k\right) + \bar{y}^m_{jk}|V_j|^2$$

Single-phase OPF

Equality constraint $f(u, x) = 0$

Power flow equations in BIM

- Equality constraints on (V, s)

$$s_j = \sum_{k:j\sim k} S_{jk}(V) := \sum_{k:j\sim k} \bar{y}^s_{jk}\left(|V_j|^2 - V_j\bar{V}_k\right) + \bar{y}^m_{jj}|V_j|^2, \qquad j \in \overline{N}$$

where $y^m_{jj} := \sum_{k:j\sim k} y^m_{jk}$

- Derivation:

$$I_{jk}(V) := y^s_{jk}(V_j - V_k) + y^m_{jk}V_j$$

$$S_{jk}(V) := V_j\bar{I}_{jk}(V) := \bar{y}^s_{jk}\left(|V_j|^2 - V_j\bar{V}_k\right) + \bar{y}^m_{jk}|V_j|^2$$

- Can also use polar form and Cartesian form
- Nonlinear and global equality constraints, resulting in nonconvexity of OPF

Single-phase OPF

Inequality constraint $g(u, x) \leq 0$

Operational constraints

- Injection limits (e.g. gen. or load capacity limits): $s_j^{\min} \leq s_j \leq s_j^{\max}$

Single-phase OPF

Inequality constraint $g(u, x) \leq 0$

Operational constraints

- Injection limits (e.g. gen. or load capacity limits): $s_j^{\min} \leq s_j \leq s_j^{\max}$
- Voltage limits: $v_j^{\min} \leq |V_j|^2 \leq v_j^{\max}$

Single-phase OPF

Inequality constraint $g(u, x) \leq 0$

Operational constraints

- Injection limits (e.g. gen. or load capacity limits): $s_j^{\min} \leq s_j \leq s_j^{\max}$
- Voltage limits: $v_j^{\min} \leq |V_j|^2 \leq v_j^{\max}$
- Line limits: $|I_{jk}(V)|^2 \leq \ell_{jk}^{\max}$, $|I_{kj}(V)|^2 \leq \ell_{kj}^{\max}$

$$\left| y_{jk}^s (V_j - V_k) + y_{jk}^m V_j \right|^2 \leq \ell_{jk}^{\max}, \qquad (j,k) \in E$$

$$\left| y_{kj}^s (V_k - V_j) + y_{kj}^m V_k \right|^2 \leq \ell_{kj}^{\max}, \qquad (j,k) \in E$$

Single-phase OPF

Inequality constraint $g(u, x) \leq 0$

Operational constraints

- Injection limits (e.g. gen. or load capacity limits): $s_j^{\min} \leq s_j \leq s_j^{\max}$
- Voltage limits: $v_j^{\min} \leq |V_j|^2 \leq v_j^{\max}$
- Line limits: $|I_{jk}(V)|^2 \leq \ell_{jk}^{\max}$, $|I_{kj}(V)|^2 \leq \ell_{kj}^{\max}$

$$\left| y_{jk}^s (V_j - V_k) + y_{jk}^m V_j \right|^2 \leq \ell_{jk}^{\max}, \qquad (j,k) \in E$$

$$\left| y_{kj}^s (V_k - V_j) + y_{kj}^m V_k \right|^2 \leq \ell_{kj}^{\max}, \qquad (j,k) \in E$$

Line limits can also be on line powers $\left(S_{jk}(V), S_{kj}(V) \right)$ or apparent powers $\left(\left| S_{jk}(V) \right|, \left| S_{kj}(V) \right| \right)$

Single-phase OPF

Simplest formulation

OPF in BIM

$$\min_{(s,V)} \quad C(s,V)$$

$$\text{s.t.} \quad f(s,V) = 0 \qquad \text{power flow equations}$$

$$g(s,V) \leq 0 \qquad \text{operational constraints}$$

Single-phase OPF

Simplest formulation

OPF in BIM

$$\begin{aligned} \min_{(s,V)} \quad & C(s,V) \\ \text{s.t.} \quad & f(s,V) = 0 \qquad \text{power flow equations} \\ & g(s,V) \leq 0 \qquad \text{operational constraints} \end{aligned}$$

- Does not need assumption $y^s_{jk} = y^s_{kj}$
- Can accommodate single-phase transformers with *complex* turns ratios
- Can allow voltages or power injections be fixed and given; e.g., $s_j^{\min} = s_j^{\max}$
- … or unconstrained, e.g., $s_0^{\min} := -\infty - i\infty, \quad s_0^{\max} := \infty + i\infty$
- Can include other devices, or more than a single device on a bus

OPF as QCQP

OPF in BIM

$$\min_{s,V} \quad c(s,V) \quad \text{s.t.} \quad f(s,V) = 0, \quad g(s,V) \leq 0$$

Can formulate OPF in terms of V only

- Use power flow equation $f(s,V) = 0$ to express injections $s_j(V)$ as functions of V
- Eliminate s_j and equality constraint $f(s,V) = 0$

$$\Longrightarrow \min_{V} \quad c(s(V),V) \quad \text{s.t.} \quad g(s(V),V) \leq 0$$

nonconvex quadratically constrained quadratic program (QCQP)

OPF as QCQP

In terms of V only

Equality constraints (BIM in complex form)

- Expresses s_j in terms of voltages V

$$s_j(V) \;=\; \sum_{k:j\sim k} S_{jk}(V) \quad := \quad \sum_{k:j\sim k} \bar{y}^s_{jk}\left(\,|\,V_j|^2 - V_j\bar{V}_k\right) \;+\; \bar{y}^m_{jj}\,|\,V_j|^2, \qquad j \in \overline{N}$$

Cost $C(V) := C(s(V), V)$ expressed as function of V

- Fuel cost:

$$C(V) \;:=\; \sum_{j:\text{gens}} c_j\,\mathrm{Re}(s_j(V)) \;=\; \sum_{j:\text{gens}} c_j\,\mathrm{Re}\left(\sum_{k:j\sim k} \bar{y}^s_{jk}\left(\,|\,V_j|^2 - V_j\bar{V}_k\right) \;+\; \bar{y}^m_{jj}\,|\,V_j|^2\right)$$

- Total real power loss:

$$C(V) \;:=\; \sum_j \mathrm{Re}(s_j(V))$$

OPF as QCQP

Operational constraints

Injection limits (e.g. generation or load capacity limits) $s_j^{\min} \leq s_j(V) \leq s_j^{\max}$:

$$s_j^{\min} \leq \sum_{k:j\sim k} \bar{y}_{jk}^s \left(|V_j|^2 - V_j \bar{V}_k \right) + \bar{y}_{jj}^m |V_j|^2 \leq s_j^{\max}, \qquad j \in \overline{N}$$

- Or in polar form:

$$p_j^{\min} \leq \sum_{k:k\sim j} \left(g_{jk}^s + g_{jk}^m \right) |V_j|^2 - \sum_{k:k\sim j} |V_j||V_k| \left(g_{jk}^s \cos\theta_{jk} + b_{jk}^s \sin\theta_{jk} \right) \leq p_j^{\max}$$

$$q_j^{\min} \leq -\sum_{k:k\sim j} \left(b_{jk}^s + b_{jk}^m \right) |V_j|^2 - \sum_{k:k\sim j} |V_j||V_k| \left(g_{jk}^s \sin\theta_{jk} - b_{jk}^s \cos\theta_{jk} \right) \leq q_j^{\max}$$

OPF as QCQP

In terms of V only

Feasible set

$$\mathbb{V} := \left\{ V \in \mathbb{C}^{N+1} \mid V \text{ satisfies operational constraints} \right\}$$

OPF in BIM

$$\min_{V \in \mathbb{V}} \quad C(V)$$

- Does not need assumption $y^s_{jk} = y^s_{kj}$
- Can accommodate single-phase transformers with *complex* turns ratios

OPF as QCQP

QCQP

Quadratically constrained quadratic program:

$$\begin{aligned} \min_{x \in \mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \ \leq \ b_l, \qquad l = 1, \ldots, L \end{aligned}$$

- C_l : $n \times n$ Hermitian matrix $\Rightarrow x^{\mathsf{H}} C_l x \in \mathbb{R}$
- $b_l \in \mathbb{R}$
- Homogeneous QCQP : all monomials are of degree 2

OPF as QCQP

QCQP

Quadratically constrained quadratic program:

$$\begin{aligned} \min_{x \in \mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \;\leq\; b_l, \qquad l = 1, \ldots, L \end{aligned}$$

Inhomogeneous QCQP

$$\begin{aligned} \min_{x \in \mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \;+\; \left(c_0^{\mathsf{H}} x + x^{\mathsf{H}} c_0 \right) \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \;+\; \left(c_l^{\mathsf{H}} x + x^{\mathsf{H}} c_l \right) \;\leq\; b_l, \qquad l = 1, \ldots, L \end{aligned}$$

Can always be homogenized

OPF as QCQP

Equivalent real QCQP

Even though OPF is often formulated in $\mathbb{C}$, it is converted to $\mathbb{R}$ before being solved iteratively

QCQP

$$\min_{x \in \mathbb{C}^n} \quad x^{\mathsf{H}} C_0 x$$

$$\text{s.t.} \quad x^{\mathsf{H}} C_l x \;\leq\; b_l, \qquad l = 1, \ldots, L$$

- C_l : $n \times n$ complex Hermitian matrix
- $b_l \in \mathbb{R}$

Equivalent to:

$$\min_{(x_r, x_i) \in \mathbb{R}^{2n}} \quad \begin{bmatrix} x_r \\ x_i \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} C_{0r} & -C_{0i} \\ C_{0i} & C_{0r} \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix}$$

$$\text{s.t.} \quad \begin{bmatrix} x_r \\ x_i \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} C_{lr} & -C_{li} \\ C_{li} & C_{lr} \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix} \leq\; b_l, \quad l = 1, \ldots, L$$

- $2n \times 2n$ real symmetric matrices

OPF as QCQP

Rewrite operational constraints and cost function in quadratic form

$$
\begin{aligned}
\min_{V \in \mathbb{C}^{N+1}} \quad & V^{\mathsf{H}} C_0 V \\
\text{s.t.} \quad & p_j^{\min} \le V^{\mathsf{H}} \Phi_j V \le p_j^{\max}, && j \in \overline{N} \\
& q_j^{\min} \le V^{\mathsf{H}} \Psi_j V \le q_j^{\max}, && j \in \overline{N} \\
& v_j^{\min} \le V^{\mathsf{H}} E_j V \le v_j^{\max}, && j \in \overline{N} \\
& V^{\mathsf{H}} \hat{Y}_{jk} V \le \ell_{jk}^{\max}, && (j,k) \in E \\
& V^{\mathsf{H}} \hat{Y}_{kj} V \le \ell_{kj}^{\max}, && (j,k) \in E
\end{aligned}
$$

Homogeneous nonconvex QCQP

Dealing with nonconvexity

OPF is nonconvex and NP-hard

- Verma (2009), Lavaei-Low (2012), Lehmann-Grastien-Hentenryck (2016), Khonji-Chau-Elbassioni (2018), Bienstock-Verma (2019)

There are 3 traditional ways to deal with nonconvexity

- One of them is semidefinite relaxation
- We discuss other ways below

Semidefinite relaxation

Equivalent QCQP problem

Quadratically constrained quadratic program:

$$\begin{aligned} \min_{x \in \mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \;\leq\; b_l, \qquad l = 1, \ldots, L \end{aligned}$$

on

Semidefinite relaxation

Equivalent QCQP problem

Quadratically constrained quadratic program:

$$\begin{aligned} \min_{x \in \mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \;\leq\; b_l, \qquad l = 1, \ldots, L \end{aligned}$$

Using $x^{\mathsf{H}} C_l x = \text{tr}\left(C_l x x^{\mathsf{H}}\right)$, this is equivalent to:

$$\begin{aligned} \min_{X \in \mathbb{S}^n, x \in \mathbb{C}^n} \quad & \text{tr}\left(C_0 X\right) \\ \text{s.t.} \quad & \text{tr}\left(C_l X\right) \;\leq\; b_l, \quad l = 1, \ldots, L \\ & X \;=\; x x^{\mathsf{H}} \end{aligned}$$

on

Semidefinite relaxation

Equivalent QCQP problem

Quadratically constrained quadratic program:

$$\begin{aligned} \min_{x \in \mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \leq b_l, \qquad l = 1, \ldots, L \end{aligned}$$

Using $x^{\mathsf{H}} C_l x = \text{tr}\left(C_l x x^{\mathsf{H}}\right)$, this is equivalent to:

$$\begin{aligned} \min_{X \in \mathbb{S}^n, x \in \mathbb{C}^n} \quad & \text{tr}\left(C_0 X\right) \\ \text{s.t.} \quad & \text{tr}\left(C_l X\right) \leq b_l, \quad l = 1, \ldots, L \\ & X = x x^{\mathsf{H}} \end{aligned}$$

- Any psd rank-1 matrix $X \in \mathbb{S}_+^{n \times n}$ has a spectral decomposition $X = x x^{\mathsf{H}}$ for some $x \in \mathbb{C}^n$, unique up to a rotation
- Therefore can eliminate x

Semidefinite relaxation

Equivalent QCQP problem

Quadratically constrained quadratic program:

$$\begin{aligned} \min_{x\in\mathbb{C}^n} \quad & x^{\mathsf{H}} C_0 x \\ \text{s.t.} \quad & x^{\mathsf{H}} C_l x \;\leq\; b_l, \qquad l = 1,\ldots,L \end{aligned}$$

Using $x^{\mathsf{H}} C_l x = \text{tr}\left(C_l x x^{\mathsf{H}}\right)$, this is equivalent to:

$$\begin{aligned} \min_{X\in\mathbb{S}^n, x\in\mathbb{C}^n} \quad & \text{tr}\left(C_0 X\right) \\ \text{s.t.} \quad & \text{tr}\left(C_l X\right) \;\leq\; b_l, \quad l = 1,\ldots,L \\ & X \;\succeq\; 0, \quad \text{rank}(X) \;=\; 1 \end{aligned}$$

- $\text{tr}\left(C_l X\right) \leq b_l$ is linear in X, $X \succeq 0$ is convex in X
- $\text{rank}(X) = 1$ is nonconvex in X removing rank constraint yields SDP relaxation

Semidefinite relaxation

SDP relaxation of QCQP

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \quad & \text{tr}\left(C_0 X\right) \\ \text{s.t.} \quad & \text{tr}\left(C_l X\right) \leq b_l, \quad l = 1, \ldots, L \\ & X \succeq 0 \end{aligned}$$

- This is a standard semidefinite program which is a convex problem
- Solution strategy:
  - Solve SDP for an optimal solution X^{opt}
  - If $\text{rank}\left(X^{\text{opt}}\right) = 1$, then $x^{\text{opt}} \in \mathbb{C}^n$ from spectral decomposition $X^{\text{opt}} = x^{\text{opt}} \left(x^{\text{opt}}\right)^{\text{H}}$ is optimal
- If $\text{rank}\left(X^{\text{opt}}\right) > 1$, then, in general, no feasible solution of QCQP can be directly obtained

Semidefinite relaxation

SDP relaxation of QCQP

$$\begin{aligned} \min_{X \in \mathbb{S}^n} \quad & \text{tr}\left(C_0 X\right) \\ \text{s.t.} \quad & \text{tr}\left(C_l X\right) \;\leq\; b_l, \quad l = 1, \ldots, L \\ & X \;\succeq\; 0 \end{aligned}$$

- Even though SDP is convex, for large networks, it is still computationally impractical
- How to exploit sparsity of large networks to reduce computational burden?

Ans: partial matrices and completions !
See, e.g., Power System Analysis (Ch 10)

Outline

1. OPF: basic formulation (10 mins)
2. OPF: unbalanced radial network (15 mins)
   - Three-phase in BFM
3. Solution approaches (10 mins)
4. New challenge: uncertainty (15 mins)

$\Longrightarrow$ machine learning for OPF

Motivation

Almost all distribution systems are unbalanced three-phase radial networks

- Distribution systems are where most innovations are happening

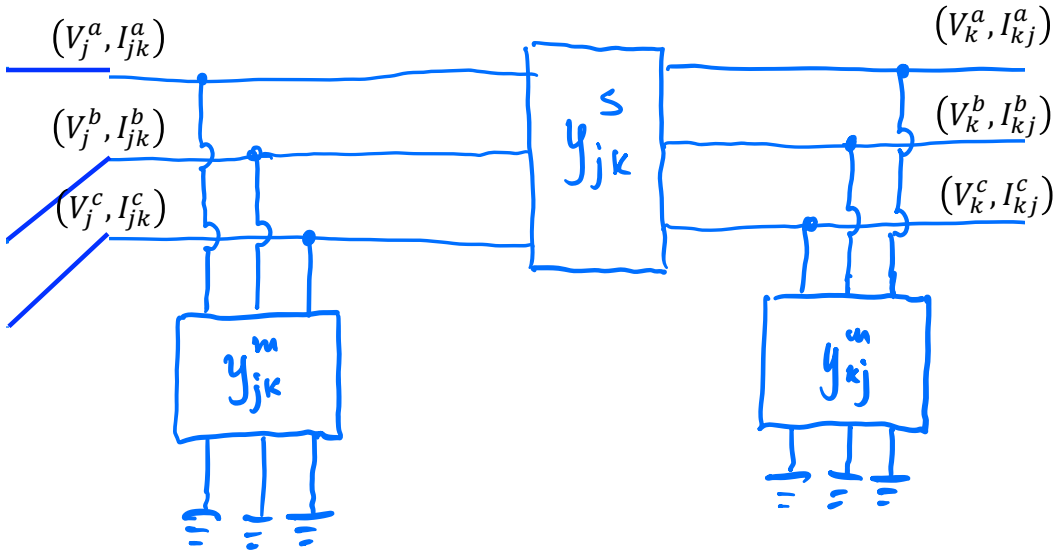
Single-phase BIM is suitable for transmission systems

- Power flows are (mostly) balanced
- Networks are (mostly) meshed

Three-phase BFM is suitable for distribution systems

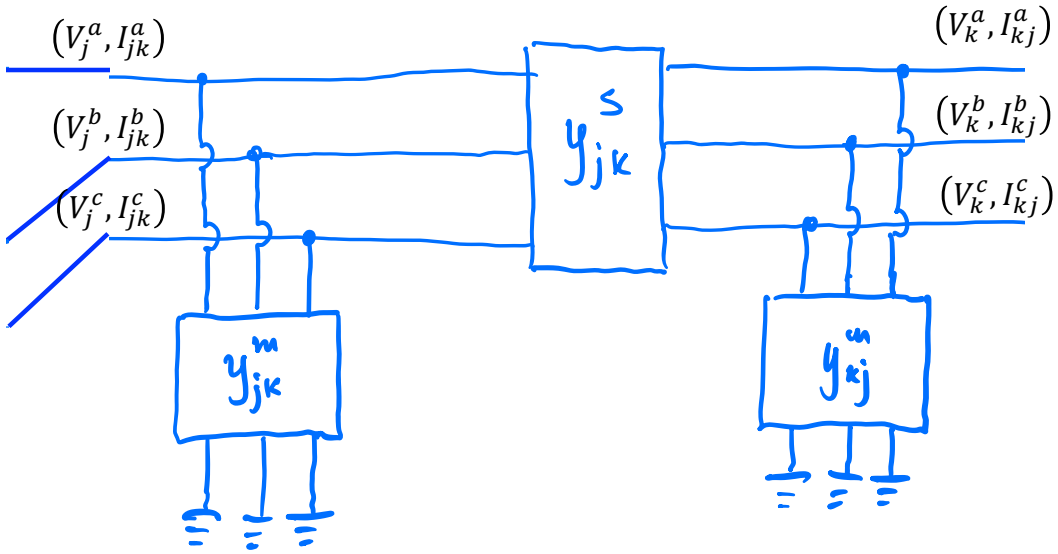
- Takes into account of unbalanced power flows
- Exploits radial (tree) structure
- … when only single-phase devices that make up a three-phase device are directly controllable

Motivation



- Many models assume terminal currents $\left(I_{jk}^a, I_{jk}^b, I_{jk}^c\right)$ are controllable (optimization vars)
- Extension to 3-phase setting is straightforward

Motivation

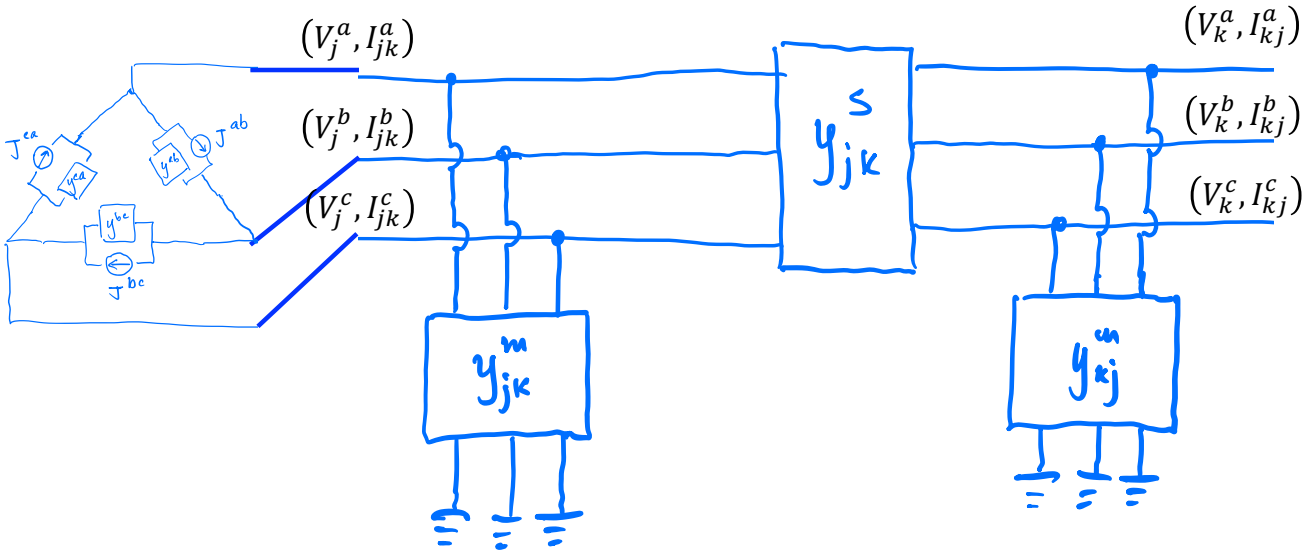


$$I_{jk} = y_{jk}^s(V_j - V_k) + V_{jk}^m V_j$$
$$I_{kj} = y_{kj}^s(V_k - V_j) + V_{kj}^m V_k$$

1-phase: $I_{jk}, V_j \in \mathbb{C},\ y_{jk}^s, y_{jk}^m \in \mathbb{C}$

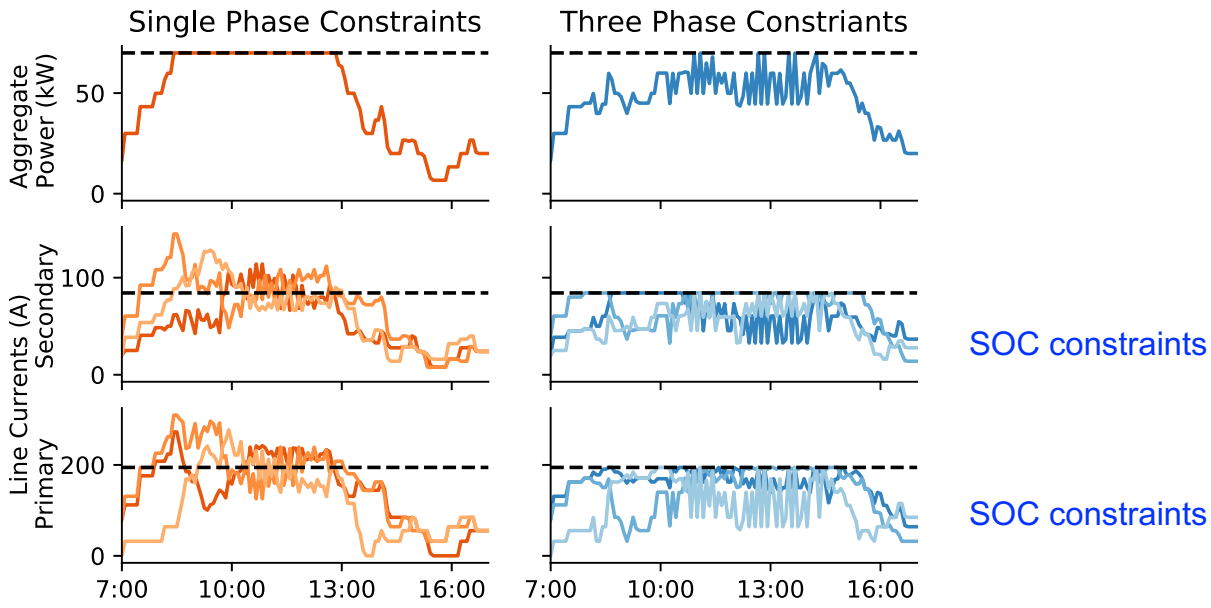
3-phase: $I_{jk}, V_j \in \mathbb{C}^3,\ y_{jk}^s, y_{jk}^m \in \mathbb{C}^{3\times 3}$

Motivation



- Terminal currents $\left(I_{jk}^a, I_{jk}^b, I_{jk}^c\right)$ are externally observable, but often not directly controllable
- When only internal currents $\left(J^{ab}, J^{bc}, J^{ca}\right)$ are controllable, e.g., EV charging rates, then 3-phase device models to convert between internal and terminal vars

Motivation



Left panel: Actual 3-phase currents violate capacity constraints if “single-phase constraints” are used (ACN-Sim based on Caltech ACN on Sept 5, 2018 data)

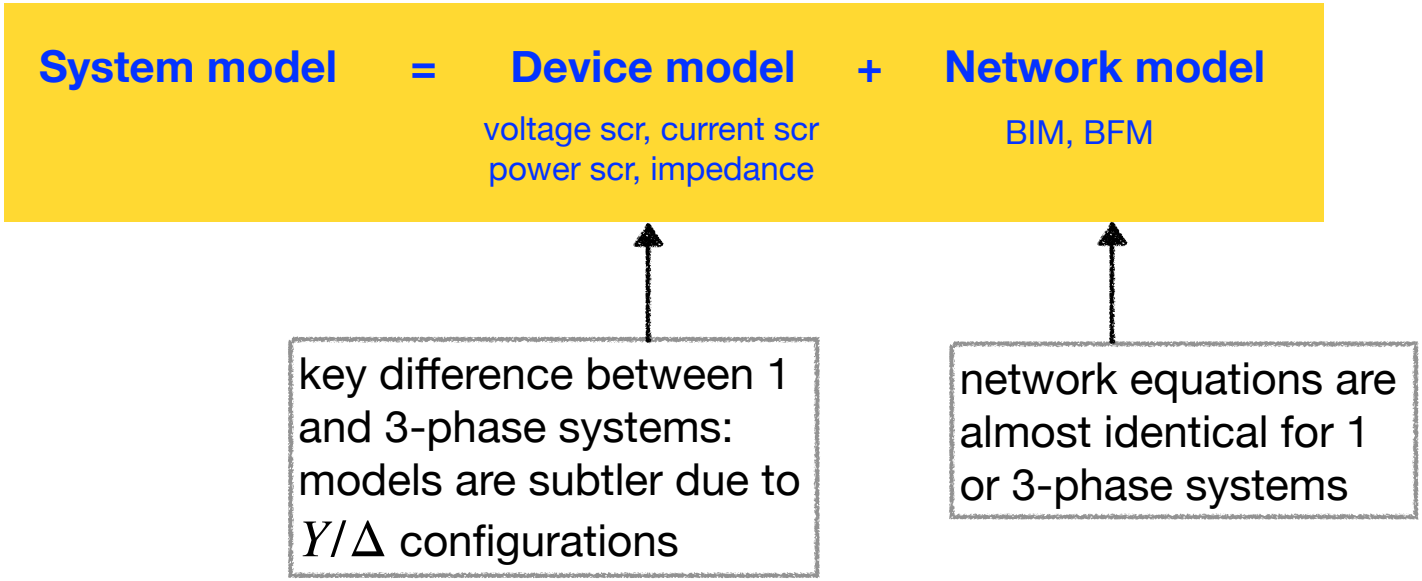
“single-phase constraints” : $\sum_i r_i(t) \leq R$ (no phase line constraints for lack of phase info)

[Lee et al (2021), ACN-Sim, TSG]

System model

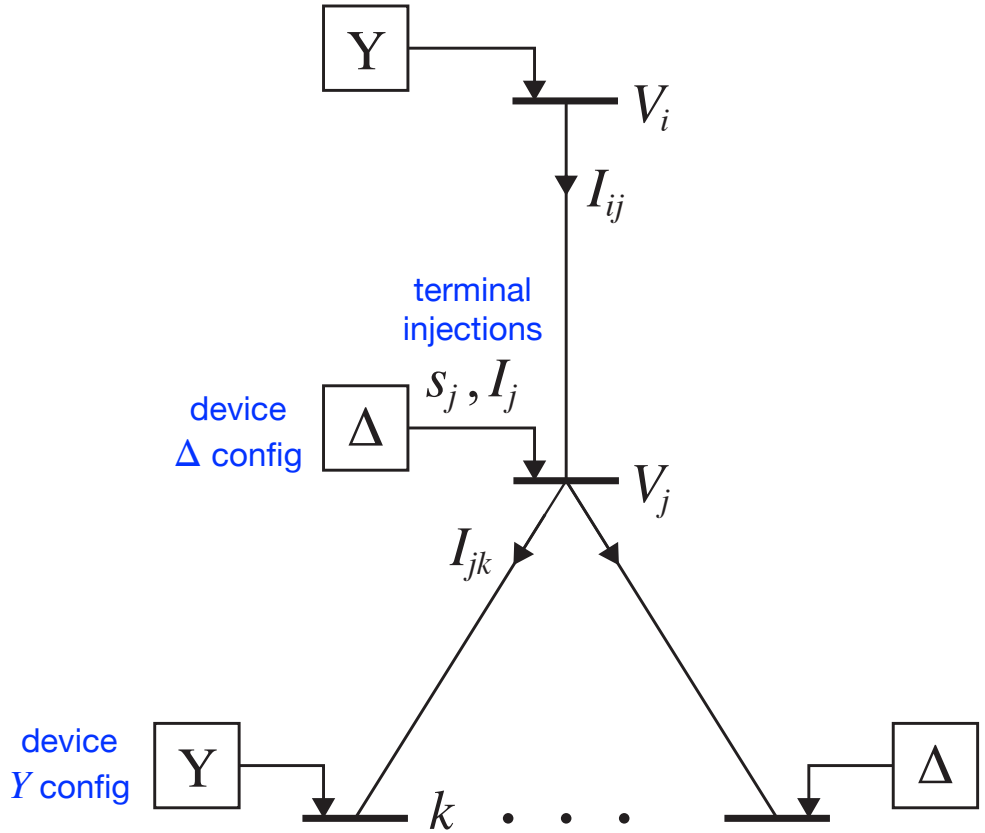
System model	=	Device model	+	Network model
		voltage scr, current scr power scr, impedance		BIM, BFM

System model



3-phase system model

Device + network



- Internal vars $\left(V_j^{Y/\Delta}, I_j^{Y/\Delta}, s_j^{Y/\Delta}\right)$ are controllable, depending on types of device
- Terminal vars $\left(V_j, I_j, s_j\right)$ are not directly controllable
- Devices interact over network only through their terminal vars, but controlled through their internal vars

Three-phase OPF

Overview

1. Derive device model
   - Internal vars u_j, terminal vars $\left(s_j, V_j\right)$
   - Conversion rule, $f_j^{Y/\Delta}\left(u_j, s_j, V_j\right) = 0$
   - Internal (local) device constraints $g_j^{Y/\Delta}(u_j) \leq 0$
2. Derive BFM (network model)
   - Nonlinear power flow equations (global nonconvex equality constraints)
   - Operational constraints (local linear inequality constraints)
3. Derive three-phase OPF in BFM
   - Nonconvex constrained optimization

Three-phase OPF

Overview

1. Derive device model
   - Internal vars u_j, terminal vars $\left(s_j, V_j\right)$
   - Conversion rule, $f_j^{Y/\Delta}\left(u_j, s_j, V_j\right) = 0$
   - Internal (local) device constraints $g_j^{Y/\Delta}(u_j) \leq 0$
2. Derive BFM (network model)
   - Nonlinear power flow equations (global nonconvex equality constraints)
   - Operational constraints (local linear inequality constraints)
3. Derive three-phase OPF in BFM
   - Nonconvex constrained optimization

Explain each in turn: var (u, x), cost c, constraints f, g

Three-phase OPF

Overview

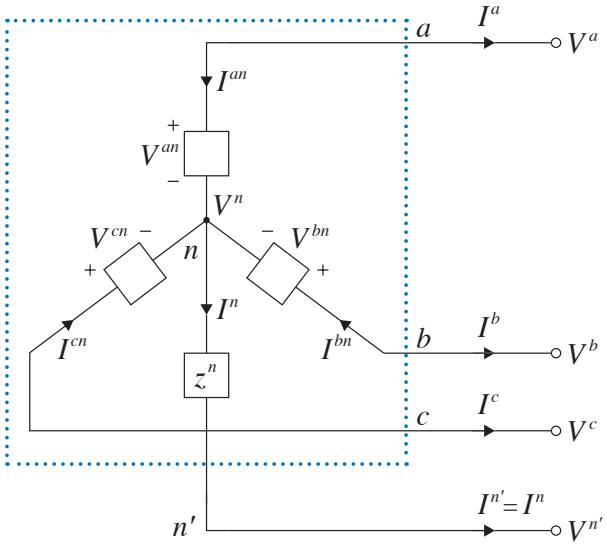
1. Derive device model
   - Internal vars u_j, terminal vars $\left(s_j, V_j\right)$
   - Conversion rule, $f_j^{Y/\Delta}\left(u_j, s_j, V_j\right) = 0$
   - Internal (local) device constraints $g_j^{Y/\Delta}(u_j) \leq 0$ (PSA Ch 14)

Internal variables

Y configuration

Internal voltage, current, power:

$$V^Y := \begin{bmatrix} V^{an} \\ V^{bn} \\ V^{cn} \end{bmatrix}, \; I^Y := \begin{bmatrix} I^{an} \\ I^{bn} \\ I^{cn} \end{bmatrix}, \; s^Y := \begin{bmatrix} s^{an} \\ s^{bn} \\ s^{cn} \end{bmatrix} := \begin{bmatrix} V^{an}\bar{I}^{an} \\ V^{bn}\bar{I}^{bn} \\ V^{cn}\bar{I}^{cn} \end{bmatrix}$$



neutral voltage (wrt common reference pt) $V^n \in \mathbb{C}$

neutral current (away from neutral) $I^n \in \mathbb{C}$

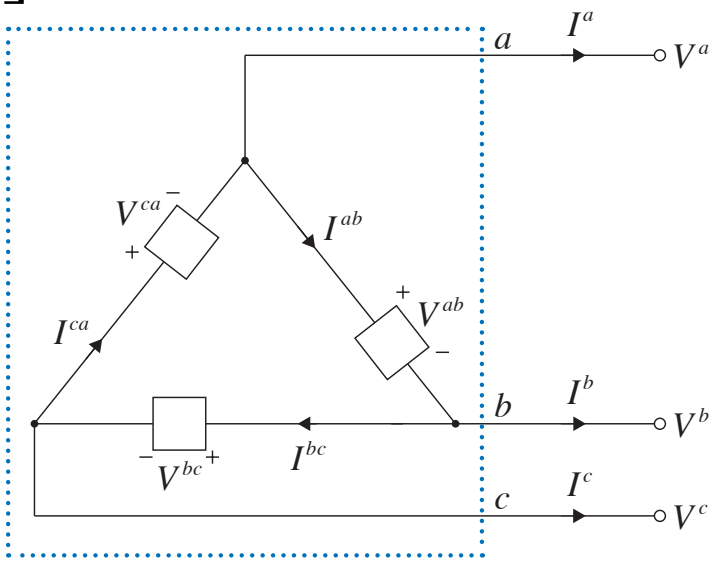
Device may or may not be grounded, and neutral impedance z^n may or may not be zero

Internal variables

Δ configuration

Internal voltage, current, power:

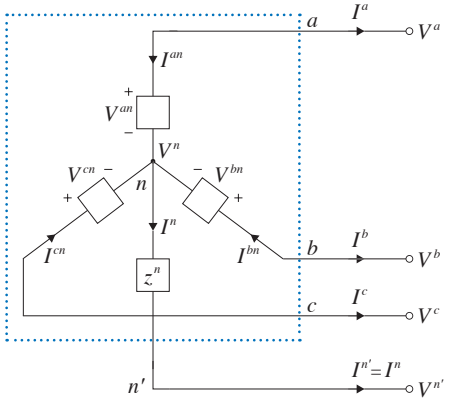
$$V^{\Delta} := \begin{bmatrix} V^{ab} \\ V^{bc} \\ V^{ca} \end{bmatrix}, \; I^{\Delta} := \begin{bmatrix} I^{ab} \\ I^{bc} \\ I^{ca} \end{bmatrix}, \; s^{\Delta} := \begin{bmatrix} s^{ab} \\ s^{bc} \\ s^{ca} \end{bmatrix} := \begin{bmatrix} V^{ab}\bar{I}^{ab} \\ V^{bc}\bar{I}^{bc} \\ V^{ca}\bar{I}^{ca} \end{bmatrix}$$



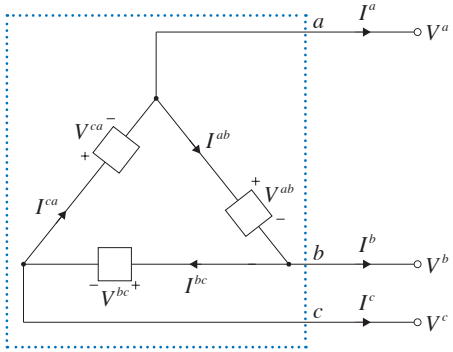
Terminal variables

Terminal voltage, current, power (for both Y and Δ):

$$V := \begin{bmatrix} V^a \\ V^b \\ V^c \end{bmatrix}, \; I := \begin{bmatrix} I^a \\ I^b \\ I^c \end{bmatrix}, \; s := \begin{bmatrix} s^a \\ s^b \\ s^c \end{bmatrix} := \begin{bmatrix} V^a \bar{I}^a \\ V^b \bar{I}^b \\ V^c \bar{I}^c \end{bmatrix}$$

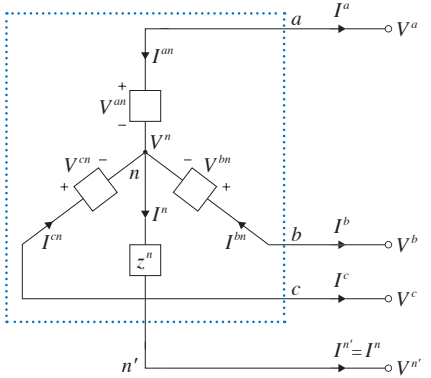


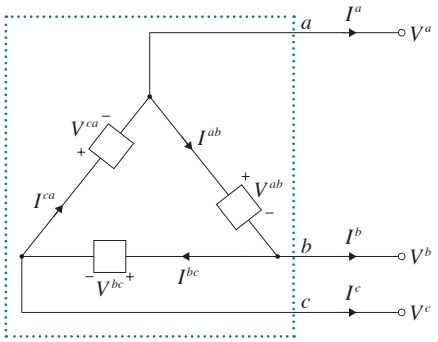
- V is with respect to an arbitrary common reference point, e.g. the ground
- I and s are in the direction out of the device



Internal vs external model

1. Internal model depends only on type of single-phase devices
   - Internal model: relation between $\left(V^{Y/\Delta}, I^{Y/\Delta}, s^{Y/\Delta}\right)$
   - Voltage/current/power source, impedance
   - Independent of Y or Δ configuration
2. Conversion rule depends only on type of configuration
   - Converts between internal and terminal variables
   - Depends only on Y or Δ configuration
   - Independent of type of single-phase devices
3. External model = Internal model + Conversion rule
   - External model: relation between (V, I, s)
   - Devices interact over network only through their terminal variables



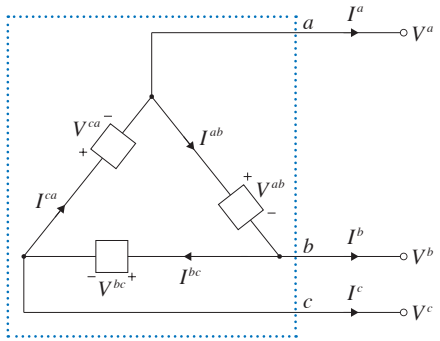


What is conversion rule? How to model a 3-p device using conversion rule?

Conversion rule

Convert between internal vars and external vars

$$\begin{bmatrix} V^{ab} \\ V^{bc} \\ V^{ca} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix}}_{\Gamma} \begin{bmatrix} V^{a} \\ V^{b} \\ V^{c} \end{bmatrix}, \qquad \begin{bmatrix} I^{a} \\ I^{b} \\ I^{c} \end{bmatrix} = - \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}}_{\Gamma^{\mathsf{T}}} \begin{bmatrix} I^{ab} \\ I^{bc} \\ I^{ca} \end{bmatrix}$$



Conversion rule

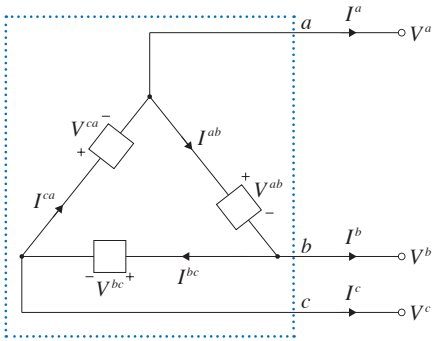
Convert between internal vars and external vars

$$\begin{bmatrix} V^{ab} \\ V^{bc} \\ V^{ca} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{bmatrix}}_{\Gamma} \begin{bmatrix} V^{a} \\ V^{b} \\ V^{c} \end{bmatrix}, \qquad \begin{bmatrix} I^{a} \\ I^{b} \\ I^{c} \end{bmatrix} = - \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}}_{\Gamma^{\mathsf{T}}} \begin{bmatrix} I^{ab} \\ I^{bc} \\ I^{ca} \end{bmatrix}$$

In vector form

$$V^{\Delta} = \Gamma V, \qquad I = -\Gamma^{\mathsf{T}} I^{\Delta}$$

internal voltage — terminal voltage — terminal current — internal current



Conversion matrices $\Gamma, \Gamma^{\mathsf{T}}$

Most properties of unbalanced 3-phase systems boils down to

- Kirchhoff laws, spectral properties of $\Gamma, \Gamma^{\mathsf{T}}$
- e.g., single-phase equivalent of balanced 3-phase systems

Conversion matrices $\Gamma, \Gamma^{\mathsf{T}}$

Most properties of unbalanced 3-phase systems boils down to

- Kirchhoff laws, spectral properties of $\Gamma, \Gamma^{\mathsf{T}}$
- e.g., single-phase equivalent of balanced 3-phase systems

Theorem

1. The null spaces of Γ and Γ^{T} are both span($\mathbf{1}$).
2. Γ is normal. Moreover, $\Gamma\Gamma^{\dagger} = \Gamma^{\dagger}\Gamma = \frac{1}{3}\Gamma\Gamma^{\mathsf{T}} = \frac{1}{3}\Gamma^{\mathsf{T}}\Gamma = \mathbb{I} - \frac{1}{3}\mathbf{1}\mathbf{1}^{\mathsf{T}}$
3. Their Pseudo-inverses are: $\Gamma^{\dagger} = \frac{1}{3}\Gamma^{\mathsf{T}}, \qquad \Gamma^{\mathsf{T}\dagger} = \frac{1}{3}\Gamma$
4. Consider $\Gamma x = b$. Solutions x exist if and only if $\mathbf{1}^{\mathsf{T}} b = 0$, in which case

$$x = \frac{1}{3}\Gamma^{\mathsf{T}} b + \gamma \mathbf{1}, \qquad \gamma \in \mathbb{C}$$

Example 3-phase devices

Voltage source $V_j^{Y/\Delta}$

- *Internal* optimization variable $u_j := V_j^{Y/\Delta}$
- Local constraint that relates internal var u_j to $\left(V_j, s_j\right)$

$$Y: \qquad V_j = V_j^Y \qquad \text{(assumed grounded)}$$

$$\Delta: \qquad \Gamma V_j = V_j^\Delta$$

Power source $\left(s_j^{Y/\Delta}, I_j^{Y/\Delta}\right)$

- *Internal* optimization variable $u_j := \left(s_j^{Y/\Delta}, I_j^{Y/\Delta}\right)$ (assume $\gamma_j^Y := V_j^n = 0$)
- Local constraint that relates internal var u_j to $\left(V_j, s_j\right)$

$$Y: \qquad s_j = -s_j^Y$$

$$\Delta: \qquad s_j = -\,\mathrm{diag}\left(V_j I_j^{\Delta \mathsf{H}} \Gamma\right), \qquad s_j^\Delta = \mathrm{diag}\left(\Gamma V_j I_j^{\Delta \mathsf{H}}\right)$$

Three-phase OPF

Overview

1. Derive device model
    - Internal vars u_j, terminal vars $\left(V_j, s_j\right)$
    - Conversion rule, $f_j^{Y/\Delta}\left(u_j, V_j, s_j\right) = 0$
    - Internal (local) device constraints $g_j^{Y/\Delta}(u_j) \leq 0$ (PSA Ch 18)
2. Derive BFM (network model)
    - Nonlinear power flow equations (global nonconvex equality constraints)
    - Operational constraints (local linear inequality constraints)
3. Derive three-phase OPF in BFM
    - Nonconvex constrained optimization

3-phase BFM

Network variables:

$$s_j \in \mathbb{C}^3, \qquad v_j \in \mathbb{S}^3_+, \qquad j \in \overline{N}$$

$$\ell_{jk} \in \mathbb{S}^3_+, \qquad S_{jk} \in \mathbb{C}^{3\times 3}, \qquad j \to k \in E$$

$\mathbb{S}^n_+$: complex psd matrices

3-phase BFM

Network variables:

$$s_j \in \mathbb{C}^3, \qquad v_j \in \mathbb{S}_+^3, \qquad j \in \overline{N}$$

$$\ell_{jk} \in \mathbb{S}_+^3, \qquad S_{jk} \in \mathbb{C}^{3\times 3}, \qquad j \to k \in E$$

$\mathbb{S}_+^n$: complex psd matrices

Power flow equations (generalized DistFlow):

$$\sum_{k:j\to k} \operatorname{diag}(S_{jk}) = \sum_{i:i\to j} \operatorname{diag}\left(S_{ij} - z_{ij}^s \ell_{ij}\right) + s_j, \qquad j \in \overline{N}$$

$$v_j - v_k = \left(z_{jk}^s S_{jk}^{\mathsf{H}} + S_{jk} z_{jk}^{s\mathsf{H}}\right) - z_{jk}^s \ell_{jk} z_{jk}^{s\mathsf{H}}, \qquad j \to k \in E$$

$$\begin{bmatrix} v_j & S_{jk} \\ S_{jk}^{\mathsf{H}} & \ell_{jk} \end{bmatrix} \succeq 0, \qquad \operatorname{rank} \begin{bmatrix} v_j & S_{jk} \\ S_{jk}^{\mathsf{H}} & \ell_{jk} \end{bmatrix} = 1, \qquad j \to k \in E$$

3-phase BFM

Network variables:

$$s_j \in \mathbb{C}^3, \qquad v_j \in \mathbb{S}^3_+, \qquad j \in \overline{N}$$
$$\ell_{jk} \in \mathbb{S}^3_+, \qquad S_{jk} \in \mathbb{C}^{3\times 3}, \qquad j \to k \in E$$

$\mathbb{S}^n_+$: complex psd matrices

Power flow equations (generalized DistFlow):

$$\sum_{k:j\to k} \operatorname{diag}(S_{jk}) = \sum_{i:i\to j} \operatorname{diag}\left(S_{ij} - z^s_{ij}\ell_{ij}\right) + s_j, \qquad j \in \overline{N}$$
$$v_j - v_k = \left(z^s_{jk} S^{\mathsf{H}}_{jk} + S_{jk} z^{s\mathsf{H}}_{jk}\right) - z^s_{jk}\ell_{jk} z^{s\mathsf{H}}_{jk}, \qquad j \to k \in E$$
$$\begin{bmatrix} v_j & S_{jk} \\ S^{\mathsf{H}}_{jk} & \ell_{jk} \end{bmatrix} \succeq 0, \qquad \operatorname{rank}\begin{bmatrix} v_j & S_{jk} \\ S^{\mathsf{H}}_{jk} & \ell_{jk} \end{bmatrix} = 1, \qquad j \to k \in E$$

injection limits: $s^{\min}_j \le s_j \le s^{\max}_j, \qquad j \in \overline{N}$

voltage limits: $v^{\min}_j \le \operatorname{diag}(v_j) \le v^{\max}_j, \qquad j \in \overline{N}$

line limits: $\operatorname{diag}(\ell_{jk}) \le \ell^{\max}_{jk}, \qquad (j,k) \in E$

3-phase OPF

3-phase OPF in BFM

$$\min_{u,x} \quad C(u,x)$$

over u internal var (depending on device)

$x := (s, v, \ell, S)$ terminal var (independent)

3-phase OPF

3-phase OPF in BFM

$$\min_{u,x} \quad C(u,x)$$

over	u	internal var (depending on device)
	$x := (s, v, \ell, S)$	terminal var (independent)
s.t.	$f_j(u_j) = 0,\ g_j(u_j) \leq 0$	device model & constraint (local)
	$f_j^{Y/\Delta}(u_j, x_j) = 0$	device conversion rule (depending on config, local)

3-phase OPF

3-phase OPF in BFM

$$\min_{u,x} \quad C(u,x)$$

over	u	internal var (depending on device)
	$x := (s, v, \ell, S)$	terminal var (independent)
s.t.	$f_j(u_j) = 0,\ g_j(u_j) \leq 0$	device model & constraint (local)
	$f_j^{Y/\Delta}(u_j, x_j) = 0$	device conversion rule (depending on config, local)
	$f(x) = 0,\ g(x) \leq 0$	network model & constraint (BFM, global)

3-phase OPF

3-phase OPF in BFM

$$\min_{u,x} \quad C(u,x)$$

over	u	internal var (depending on device)
	$x := (s, v, \ell, S)$	terminal var (independent)
s.t.	$f_j(u_j) = 0,\ g_j(u_j) \le 0$	device model & constraint (local)
	$f_j^{Y/\Delta}(u_j, x_j) = 0$	device conversion rule (depending on config, local)
	$f(x) = 0,\ g(x) \le 0$	network model & constraint (BFM, global)

Three-phase OPF in BFM is equivalent to three-phase OPF in BIM:

- Their feasible sets are equivalent (PSA Ch 10)
- Generally nonconvex
- Can apply SDP relaxation

Recap: 3-phase OPF

Device + network

1. Device model for each 3-phase device
   - Internal model on $\left(V_j^{Y/\Delta}, I_j^{Y/\Delta}, s_j^{Y/\Delta}\right)$ + conversion rules
   - External model on $\left(V_j, I_j, s_j\right)$
   - Either can be used
   - Power source models are nonlinear; other devices are linear

Our perspective:

- Internal vars $\left(V_j^{Y/\Delta}, I_j^{Y/\Delta}, s_j^{Y/\Delta}\right)$ are controllable, depending on types of device
- External vars $\left(V_j, I_j, s_j\right)$ are not directly controllable

$\therefore$ use internal model + conversion rules

Recap: 3-phase OPF

Device + network

2. Network model relates terminal vars $x := (s, v, \ell, S)$
   - 3-phase BFM (generalized DistFlow) $f(x) = 0$
   - Network constraints $g(x) \leq 0$
3. 3-phase OPF in BFM
   - Nonconvex optimization
   - Local device model, device constraints, conversion rule
   - Global network model, network constraints

Outline

1. OPF: basic formulation (10 mins)
2. OPF: unbalanced radial network (15 mins)
3. Solution approaches (10 mins)
4. New challenge: uncertainty (15 mins)

$\Longrightarrow$ machine learning for OPF

Dealing with nonconvexity

OPF is nonconvex and NP-hard

- Verma (2009), Lavaei-Low (2012), Lehmann-Grastien-Hentenryck (2016), Khonji-Chau-Elbassioni (2018), Bienstock-Verma (2019)

There are 3 traditional ways to deal with nonconvexity

1. Linear approximation
   - e.g. DC OPF is widely used for electricity market and planning applications
2. Local algorithms, e.g., Newton-Raphson, Fast Decoupled alg, interior-point
   - Optimality conditions for convex problems not applicable
3. Convex relaxation, e.g., semidefinite relaxation
   - Optimality conditions apply to convex relaxations

Unlike approximations, semidefinite relaxation has 3 advantages

- We can easily check if a solution of relaxation is a global optimum
- If not, it provides a lower bound on optimal value
- If relaxation is infeasible, then the nonconvex problem is infeasible

Dealing with nonconvexity

How well do these approaches work?

1. Linear approximation
   - Good enough for electricity market (pricing needs convexity) and planning (coarse decisions)
   - Not enough for operational safety, e.g., state estimation, contingency analysis
2. Local algorithms, e.g., Newton-Raphson, Fast Decoupled alg, interior-point
   - Surprisingly well !
   - If local algorithm computes a local optimum, it is often a global optimum
3. Convex relaxation, e.g., semidefinite relaxation
   - Surprisingly well !
   - Relaxation is often exact, yielding a global optimum

Why ?

Empirical experience

OPF is “easy” in practice

- Convex relaxations often exact
- Local optima often globally optimal

ARPAe NODES quarterly review (Caltech 2013 Sept)

Simulation results

	#vars	#constrs	IPM (sec)	$\|S\|^2/\|v\|$	Eig-ratio
IEEE 13-bus	97	40	0.28	1.0000	2.10e-16
IEEE 34-bus	287	120	0.50	1.0000	3.09e-16
IEEE 37-bus	306	126	0.30	1.0000	2.45e-16
IEEE 123-bus	1,030	436	0.41	1.0000	3.31e-16
SCE 47-bus	387	168	0.56	1.0000	2.68e-14
SCE 56-bus	398	173	0.59	1.0000	7.85e-17
SCE Rossi 2145-bus	**16,593**	**6,683**	**2.20**	**0.9997**	**3.71e-16**

SOCP is fast SOCP is exact

SOCP is exact (radial)

Comparison (mesh)

Test case	Objective values ($/hr)		Running times (sec)		
	SDP/ch	SOCP	SDP	chordal	SOCP
9 bus	5297.4	5297.4	0.2	0.2	0.2
14 bus	8081.7	8075.3	0.2	0.2	0.2
30 bus	574.5	573.6	0.4	0.3	0.3
39 bus	41889.1	41881.5	0.7	0.3	0.3
57 bus	41738.3	41712.0	1.3	0.5	0.3
118 bus	129668.6	129372.4	6.9	0.7	0.6
300 bus	720031.0	719006.5	109.4	2.9	1.8
2383 bus	1840270	1789500.0	-	1005.6	155.3

SOCP inexact SDP not scalable

SDP is exact (mesh)

ARPAe NODES quarterly review (Caltech 2013 August)

Network	% inc from SDP	% inc from chordal SDP	% inc from SOCP
9-bus (line 3 = 34)	0.32	0.32	7
9-bus (line 3 = 35)	0.20	0.20	6.4
30-bus (line 33 = 7.5) flow_move_factor = 2	0.45	0.45	3.24
30-bus (line 33 = 8)	0.11	0.11	2.29
39-bus (line 2 = 220) flow move_factor = 4	0.01	0.01	0.31

< 0.5%

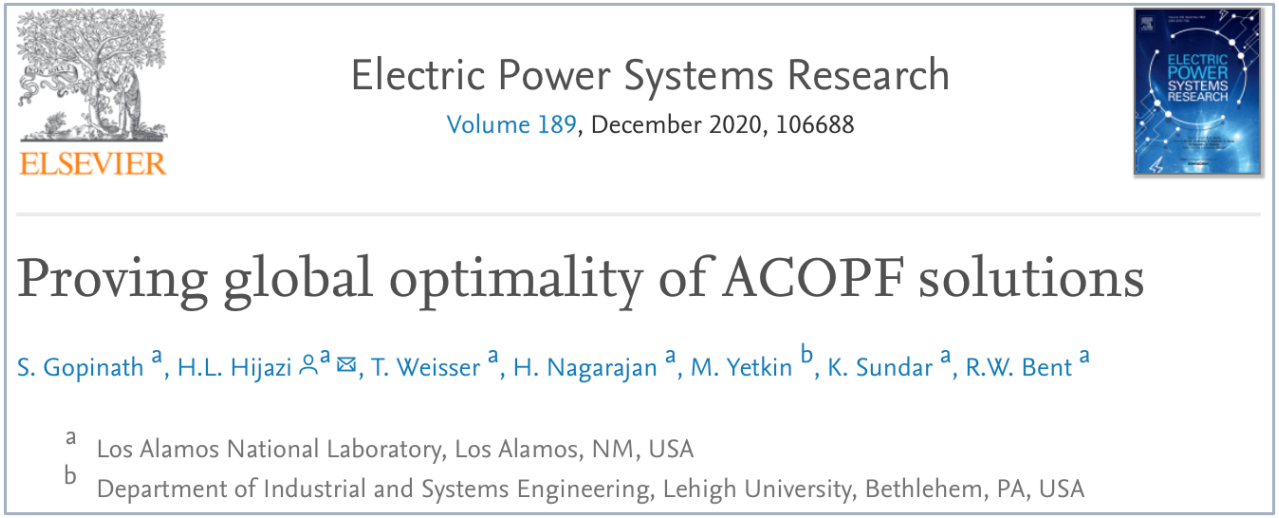
SQP corrects for the inaccuracies of SOCP

Local algorithms attain global optimal or is close (<0.5% optimality gap)

Empirical experience

OPF is “easy” in practice

- Convex relaxations often exact
- Local optima often globally optimal



ELSEVIER

Electric Power Systems Research

Volume 189, December 2020, 106688

Proving global optimality of ACOPF solutions

S. Gopinath [a], H.L. Hijazi [a] ✉, T. Weisser [a], H. Nagarajan [a], M. Yetkin [b], K. Sundar [a], R.W. Bent [a]

[a] Los Alamos National Laboratory, Los Alamos, NM, USA

[b] Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

Case	SDP-RT root gap	SDP-RT final gap	SDP-RT time	SDP-RT iter	Lasserre relaxation gap1	Lasserre relaxation time1	Lasserre relaxation gap2	Lasserre relaxation time2
pglib_opf_case3_lmbd	0.00	0.00	0.08	0	0.38	0.01	0.00	0.23
pglib_opf_case5_pjm	**0.09**	**0.09**	**0.13**	**0**	**5.22**	**0.01**	**0.00**	**20.87**
pglib_opf_case14_ieee	0.00	0.00	0.50	0	0.00	0.22	0.00	141.99
pglib_opf_case24_ieee_rts	0.00	0.00	0.86	0	0.00	0.19	-	-
pglib_opf_case30_as	0.00	0.00	0.85	0	0.00	0.14	0.00	5924.40
pglib_opf_case30_fsr	0.00	0.00	0.71	0	0.01	0.23	-	-
pglib_opf_case30_ieee	0.00	0.00	0.68	0	0.02	0.17	-	-
pglib_opf_case39_epri	0.00	0.00	1.47	0	0.01	0.23	0.00	1336.31
pglib_opf_case57_ieee	0.00	0.00	1.62	0	0.01	0.40	-	-
pglib_opf_case73_ieee_rts	0.00	0.00	3.54	0	0.00	0.60	-	-
pglib_opf_case89_pegase	0.29	0.29	12.96	0	0.34	1.97	-	-
pglib_opf_case118_ieee	0.03	0.03	3.18	0	0.07	1.29	-	-
pglib_opf_case162_ieee_dtc	**1.57**	**0.45**	**5723.00**	**1**	**1.78**	**5.83**	**-**	**-**
pglib_opf_case179_goc	0.07	0.07	6.98	0	0.07	1.64	-	-
pglib_opf_case200_tamu	0.00	0.00	7.11	0	0.00	1.47	-	-
pglib_opf_case300_ieee	**0.10**	**0.10**	**18.99**	**0**	**1.56**	**3.45**	**-**	**-**
pglib_opf_case3_lmbd__api	**0.93**	**0.93**	**0.10**	**0**	**4.99**	**0.01**	**0.00**	**0.20**
pglib_opf_case5_pjm__api	0.01	0.01	0.20	0	0.30*	0.04*	0.00	19.33
pglib_opf_case14_ieee__api	0.01	0.01	0.62	0	0.02	0.06	0.00	143.71
pglib_opf_case24_ieee_rts__api	**1.03**	**0.03**	**21.42**	**1**	**2.07**	**0.21**	**-**	**-**
pglib_opf_case30_as__api	0.72	0.72	0.82	0	16.19*	0.21*	0.39	9807.31
pglib_opf_case30_fsr__api	0.27	0.27	2.24	0	0.52	0.20	-	-
pglib_opf_case30_ieee__api	0.02	0.02	0.74	0	0.34*	0.17*	-	-
pglib_opf_case39_epri__api	0.16	0.16	0.71	0	0.46	0.26	0.01	1973.40
pglib_opf_case57_ieee__api	0.00	0.00	3.34	0	0.02	0.51	-	-
pglib_opf_case73_ieee_rts__api	**2.13**	**0.75**	**261.14**	**1**	**2.92**	**0.65**	**-**	**-**
pglib_opf_case89_pegase__api	**11.70**	**0.93**	**6013.07**	**3**	**12.12**	**2.27**	**-**	**-**
pglib_opf_case118_ieee__api	**8.44**	**0.99**	**2030.47**	**5**	**11.20**	**1.26**	**-**	**-**
pglib_opf_case162_ieee_dtc__api	**1.26**	**0.26**	**16277.76**	**1**	**1.44**	**5.33**	**-**	**-**
pglib_opf_case179_goc__api	0.54	0.54	9.01	0	0.55	1.35	-	-
pglib_opf_case200_tamu__api	0.00	0.00	39.17	0	0.00	2.05	-	-
pglib_opf_case300_ieee__api	0.07	0.07	21.87	0	0.21	3.45	-	-
pglib_opf_case3_lmbd__sad	0.10	0.10	0.08	0	0.62	0.01	0.00	0.21
pglib_opf_case5_pjm__sad	0.00	0.00	0.20	0	0.00	0.03	0.00	18.36
pglib_opf_case14_ieee__sad	0.11	0.11	0.36	0	0.09	0.10	0.00	148.34
pglib_opf_case24_ieee_rts__sad	**3.54**	**0.11**	**21.69**	**1**	**2.52**	**0.17**	**-**	**-**
pglib_opf_case30_as__sad	0.21	0.21	0.95	0	0.16	0.25	0.00	6529.10
pglib_opf_case30_fsr__sad	0.02	0.02	0.69	0	0.02	0.19	-	-
pglib_opf_case30_ieee__sad	0.00	0.00	0.84	0	0.00	0.16	-	-
pglib_opf_case39_epri__sad	0.02	0.02	1.36	0	0.02	0.24	-	-
pglib_opf_case57_ieee__sad	0.04	0.04	4.14	0	0.04	0.63	-	-
pglib_opf_case73_ieee_rts__sad	**2.13**	**0.33**	**228.97**	**1**	**1.48**	**0.58**	**-**	**-**
pglib_opf_case89_pegase__sad	0.29	0.29	12.35	0	0.32	1.95	-	-
pglib_opf_case118_ieee__sad	**2.49**	**0.18**	**471.40**	**1**	**1.83**	**1.21**	**-**	**-**
pglib_opf_case162_ieee_dtc__sad	**1.38**	**0.28**	**3666.39**	**1**	**1.79**	**5.63**	**-**	**-**
pglib_opf_case179_goc__sad	0.94	0.94	15.01	0	0.91	1.59	-	-
pglib_opf_case200_tamu__sad	0.00	0.00	8.50	0	0.00	1.64	-	-
pglib_opf_case300_ieee__sad	**0.12**	**0.12**	**13.98**	**0**	**1.40**	**3.28**	**-**	**-**

Case	SDP-RT root gap	SDP-RT final gap	SDP-RT time	SDP-RT iter	Best of [20] OBBT+cuts gap	Best of [20] time1	Best of [20] final gap	Best of [20] time2
nesta_case3_lmbd	0.00	0.00	0.03	0	0.10	0.95	0.09	0.95
nesta_case4_gs	0.00	0.00	0.03	0	0.00	0.03		
nesta_case5_pjm	**0.11**	**0.11**	**0.05**	**0**	**2.11**	**3.26**	**0.10**	**108.39**
nesta_case6_c	0.00	0.00	0.03	0	-	-	-	-
nesta_case6_ww	0.00	0.00	0.06	0	0.01	1.08		
nesta_case9_wscc	0.00	0.00	0.05	0	0.00	0.09		
nesta_case14_ieee	0.00	0.00	0.09	0	0.00	2.70		
nesta_case24_ieee_rts	0.00	0.00	0.16	0	-	-	-	-
nesta_case29_edin	0.00	0.00	0.73	0	0.01	33.99		
nesta_case30_as	0.00	0.00	0.15	0	0.06	0.11		
nesta_case30_fsr	0.01	0.01	0.17	0	0.07	14.49		
nesta_case30_ieee	0.02	0.02	0.16	0	0.03	14.55		
nesta_case39_epri	0.01	0.01	0.25	0	0.05	0.25		
nesta_case57_ieee	0.00	0.00	0.40	0	0.06	0.22		
nesta_case73_ieee_rts	0.00	0.00	2.32	0	-	-		
nesta_case118_ieee	0.02	0.02	0.97	0	0.14	355.50		
nesta_case162_ieee_dtc	**0.88**	**0.88**	**8.00**	**0**	**1.57**	**948.30**		
nesta_case189_edin	0.05	0.05	1.08	0	0.04	63.15		
nesta_case300_ieee	0.07	0.07	3.41	0	0.09	520.50		
nesta_case3_lmbd__api	0.32	0.32	0.02	0	0.81	1.05		
nesta_case4_gs__api	0.02	0.02	0.03	0	0.03	0.55		
nesta_case5_pjm__api	0.00	0.00	0.04	0	0.05	0.81		
nesta_case6_c__api	0.01	0.01	0.04	0	-	-	-	-
nesta_case6_ww__api	0.02	0.02	0.06	0	0.00	3.39		
nesta_case9_wscc__api	0.00	0.00	0.05	0	0.00	0.06		
nesta_case14_ieee__api	0.05	0.05	0.08	0	0.04	13.18		
nesta_case24_ieee_rts__api	0.54	0.54	0.20	0	-	-	-	-
nesta_case29_edin__api	0.00	0.00	2.38	0	0.04	136.83		
nesta_case30_as__api	0.29	0.29	0.17	0	0.09	62.09		
nesta_case30_fsr__api	**4.93**	**2.66**	**71.69**	**6**	**5.15**	**90.56**	**0.83**	**1802.18**
nesta_case30_ieee__api	0.10	0.10	0.19	0	0.06	60.03		
nesta_case39_epri__api	0.00	0.00	0.53	0	0.01	26.33		
nesta_case57_ieee__api	0.09	0.09	0.43	0	0.06	125.44		
nesta_case73_ieee_rts__api	0.35	0.35	1.10	0	-	-	-	-
nesta_case118_ieee__api	**17.50**	**1.91**	**2981.79**	**12**	**7.83**	**911.90**	**7.83**	**1834.74**
nesta_case162_ieee_dtc__api	**0.84**	**0.84**	**8.03**	**0**	**1.03**	**2007.66**	**1.03**	**2007.68**
nesta_case189_edin__api	0.12	0.12	1.10	0	0.91	592.86	0.12	663.19
nesta_case300_ieee__api	0.00	0.00	53.02	0	0.10	1048.07		
nesta_case3_lmbd__sad	0.11	0.11	0.02	0	0.09	1.29	0.03	1.29
nesta_case4_gs__sad	0.05	0.05	0.03	0	0.01	0.66		
nesta_case5_pjm__sad	0.04	0.04	0.04	0	0.07	0.94		
nesta_case6_c__sad	0.01	0.01	0.04	0	-	-	-	-
nesta_case6_ww__sad	0.00	0.00	0.08	0	0.00	1.53		
nesta_case9_wscc__sad	0.03	0.03	0.06	0	0.01	1.14		
nesta_case14_ieee__sad	0.00	0.00	0.09	0	0.06	0.16		
nesta_case24_ieee_rts__sad	**5.13**	**0.34**	**17.95**	**1**	-	-	-	-
nesta_case29_edin__sad	**23.21**	**0.81**	**215.20**	**2**	**0.70**	**325.68**	**0.67**	**1837.01**
nesta_case30_as__sad	0.47	0.47	0.25	0	0.09	38.85		
nesta_case30_fsr__sad	0.10	0.10	0.18	0	0.09	26.57		
nesta_case30_ieee__sad	0.03	0.03	0.15	0	0.02	26.78		
nesta_case39_epri__sad	0.05	0.05	0.28	0	0.02	11.54		
nesta_case57_ieee__sad	0.04	0.04	0.42	0	0.07	36.75		
nesta_case73_ieee_rts__sad	**3.42**	**0.73**	**202.15**	**1**	-	-	-	-
nesta_case118_ieee__sad	**5.93**	**0.29**	**1062.24**	**2**	**3.35**	**748.42**	**3.07**	**1804.74**
nesta_case162_ieee_dtc__sad	**3.23**	**0.29**	**T.L.**	**2**	**3.76**	**1741.94**		
nesta_case189_edin__sad	**1.23**	**0.25**	**562.08**	**1**	**1.41**	**315.67**	**1.06**	**1814.79**
nesta_case300_ieee__sad	0.09	0.09	3.57	0	0.10	1226.36		

- Relaxation often exact
- Enhancements close almost all gaps (<1%)

Summary

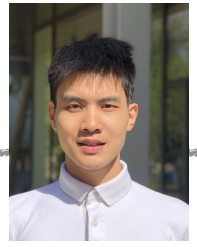
OPF is hard in theory, but “easy” in practice

- Convex relaxations often exact
- Local optima often globally optimal

Result from this paper:

1468 IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, VOL. 9, NO. 3, SEPTEMBER 2022

Conditions for Exact Convex Relaxation and No Spurious Local Optima



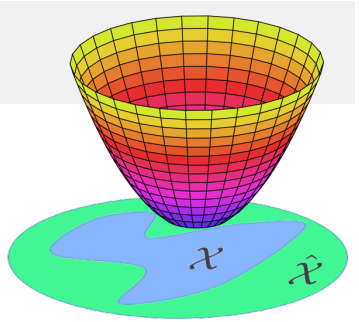
Fengyu Zhou, *Student Member, IEEE*, and Steven H. Low, *Fellow, IEEE*

Optimization and relaxation

Consider

Nonconvex optimization P1: $\min_x \; f(x) \quad \text{s.t.} \quad x \in X \subseteq \mathbb{R}^n$

Convex relaxation P2: $\min_x \; f(x) \quad \text{s.t.} \quad x \in \hat{X} \subseteq \mathbb{R}^n$



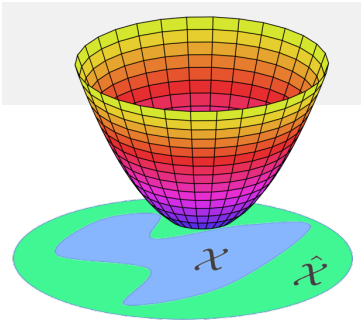
- X : nonempty, compact (not necessarily convex)
- $\hat{X}$: compact and convex superset $\hat{X} \supseteq X$
- $f : \mathbb{R}^n \to \mathbb{R}$ is convex (and hence continuous) function on $\mathbb{R}^n$

Optimal solutions exist for both problems P1 and P2

Exact relaxation

Definition

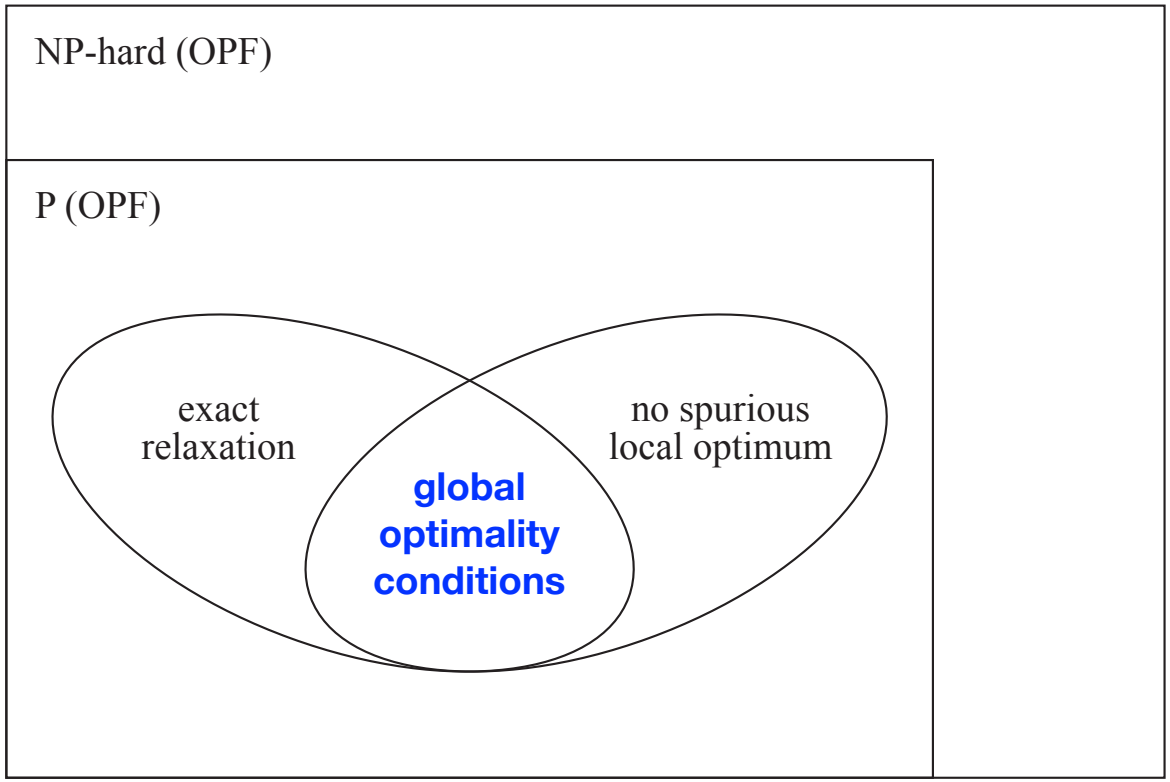
1. $x^* \in X$ is a local optimum of P1 if $\exists \delta > 0$ s.t. $f(x^*) \leq f(x)$ for all $x \in X$ with $\|x - x^*\| < \delta$
2. $x^* \in X$ is a global optimum of P1 if $f(x^*) \leq f(x)$ for all $x \in X$
3. P2 is exact wrt P1 if every optimal x^* of P2 is feasible (and hence globally optimal) for P1

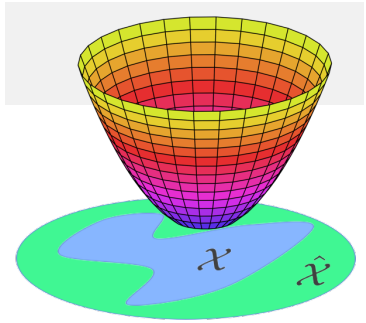


Global optimality condition

Summary

Lyapunov-like conditions for exact relaxation and no spurious local optimal

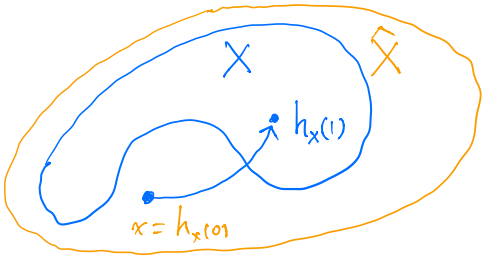




Path

Definition

1. A path in $Y \subseteq \mathbb{R}^n$ connecting a to be b in Y is a continuous function $h : [0,1] \to Y$ s.t. $h(0) = a$ and $h(1) = b$
2. An arbitrary set $\{h_i : i \in I\}$ of paths in Y is called
   - uniformly bounded if $\exists$ finite H s.t. $\|h_i(t)\|_\infty \leq H$ for all $t \in [0,1]$ and $i \in I$
   - uniformly equicontinuous if for any $\epsilon > 0$, $\exists\ \delta > 0$ s.t. $\|h_i(t_2) - h_i(t_1)\|_\infty \leq \epsilon$ for all $i \in I$ whenever $|t_2 - t_1| < \delta$

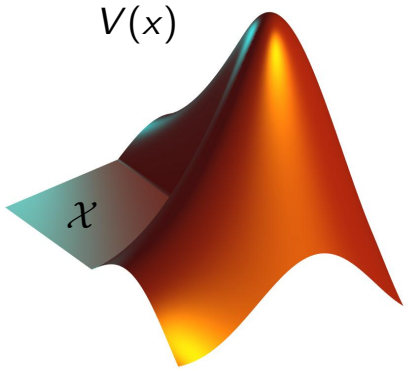


Example: If all paths in $\{h_i : i \in I\}$ are linear, then $\{h_i : i \in I\}$ is both uniformly bounded and uniformly equicontinuous

Lyapunov-like function

Definition

A Lyapunov-like function associated with problems P1 and P2 is a continuous function $V : \hat{X} \to \mathbb{R}_+$ s.t. $V(x) = 0$ if $x \in X$ and $V(x) > 0$ if $x \in \hat{X} \backslash X$



Global optimality

Optimality conditions

1. There is a Lyapunov-like function V and, for every infeasible point $x \in \hat{X} \backslash X$, $\exists$ path h_x s.t.

   (a) $h_x(0) = x,\ h_x(1) \in X,\ f(h_x(1)) < f(x)$

   Every infeasible pt x can be brought back to X with a lower cost

   (b) Both $f(h_x(t))$ and $V(h_x(t))$ are nonincreasing for $t \in [0,1]$

   Nonincreasing cost or certificate along path to feasibility

2. The set $\{h_x : x \in \hat{X} \backslash x\}$ of paths in 1 is uniformly bounded and uniformly equicontinuous

3. At least one of the following holds:

   (a) All local optima of P1 are isolated (i.e., every local optimum has a neighborhood with no other local optimum)

   (b) For $\{h_x : x \in \hat{X} \backslash x\}$ in 1, $\exists \alpha > 0$ s.t. for all $x \in \hat{X} \backslash X$ and all $0 \leq s < t \leq 1$,

   $$f(h_x(s)) - f(h_x(t)) \geq \alpha \| h_x(s) - h_x(t) \|$$

   Cost must decrease sufficiently along path to feasibility

   for some norm $\| \cdot \|$

Global optimality

Theorem [Sufficiency]

Suppose conditions 1, 2, 3 hold.

1. The convex relaxation P2 is exact wrt P1
2. Every local optimum of P1 is a global optimum

Moreover if condition 3(a) holds, then the optimal point is unique

Remarks

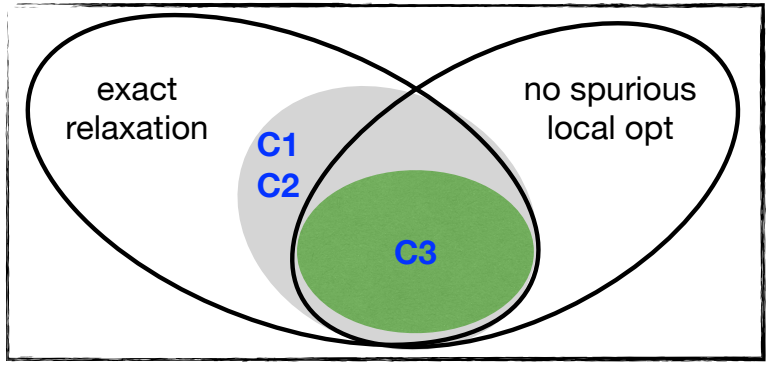
- Exactness $\iff$ existence of $\{h_x : x \in \hat{X} \backslash x\}$ that satisfies condition 1
- Other conditions are to prove that there is no spurious local optimum

Global optimality

A set $Y \subseteq \mathbb{R}^n$ is semianalytic if every $x \in Y$ has a neighborhood U s.t. $Y \cap U$ can be represented as a finite Boolean combination of sets $\{x : g(x) = 0\}$ and $\{x : h(x) < 0\}$ for some analytic functions g, h (usually satisfied by engineering problems)

Global optimality

A set $Y \subseteq \mathbb{R}^n$ is semianalytic if every $x \in Y$ has a neighborhood U s.t. $Y \cap U$ can be represented as a finite Boolean combination of sets $\{x : g(x) = 0\}$ and $\{x : h(x) < 0\}$ for some analytic functions g, h (usually satisfied by engineering problems)



Theorem [Necessity]

Suppose X is semianalytic and f is analytic. If

1. The convex relaxation P2 is exact wrt P1, and
2. Every local optimum of P1 is a global optimum

then ∃ Lyapunov-like function V and a family of paths $\{h_x : x \in \hat{X} \backslash x\}$ that satisfy conds 1 and 2

Outline

1. OPF: basic formulation (10 mins)
2. OPF: unbalanced radial network (15 mins)
3. Solution approaches (10 mins)
4. New challenge: uncertainty (15 mins)
   - Overview
   - Scenario opt
   - Two-stage opt

$\Longrightarrow$ machine learning for OPF

Uncertainty

Optimal power flow (OPF) is fundamental because it underlies numerous power system applications

- Unit commitment, optimal dispatch, state estimation, contingency analysis, voltage control, …

Deterministic OPF is a constrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x) \qquad \text{s.t.} \qquad h(x, \zeta) \leq 0$$

where ζ is a given fixed parameter

New challenge: uncertainty

- Uncertainty in future demand
- Uncertainty in renewable generations
- Random contingencies such as generator/transmission outages
- … …

Stochastic OPF

Stochastic OPF is a constrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x) \qquad \text{s.t.} \qquad h(x, \zeta) \leq 0$$

where ζ is a random parameter

- Optimal generation schedule x^* to meet random demand ζ
- Optimal inverter setpoint x^* to stabilize voltage in response to random PV generation ζ
- Optimal EV charging schedule x^* for DR on a distribution feeder with unknown admittance matrix ζ

Deterministic optimization suffices if the decision x^* can be made

- after observing the realization of the random parameter ζ, or
- using a forecast $\hat{\zeta}$ of ζ in the constraint $h(x, \hat{\zeta}) \leq 0$

Otherwise, need to take uncertain ζ into account more explicitly in decision making

Stochastic OPF

4 main ideas

Choose optimal x^* s.t.

- Robust opt: x^* satisfies constraints for all ζ in an uncertainty set Z
- Chance constrained opt: x^* satisfies constraints with high probability (instead of with probability 1)
- Scenario opt: x^* satisfies constraints for K random samples of $\zeta \in Z$
- Two-stage opt: 2nd-stage decision $y(x^*, \zeta)$ adapts to realized parameter ζ, given 1st-stage decision x^*

Many methods are combinations of these 4 ideas, e.g.

- Distributional robust opt: robust + chance constrained
- Adaptive robust opt: two-stage + robust (as opposed to expected) 2nd-stage cost
- Adaptive robust affine control: two-stage + robust (or avg) + affine policy for 2nd-stage decision $y(x^*, \zeta)$

Stochastic OPF

Consider

$$\min_{x \in \mathbb{R}^n} \; f(x) \qquad \text{s.t.} \qquad h(x, \zeta) \leq 0$$

where ζ is a random parameter

Brief introduction to theory of stochastic optimization

- Most stochastic optimization problems are intractable (e.g., nonconvex, nonsmooth)
- Explains 2 ideas to deal with uncertainty: scenario optimization, two-stage optimization

Scenario opt: motivation

Robust optimization

Consider

$$\min_{x \in X \subseteq \mathbb{R}^n} \quad c^\mathsf{T} x \quad \text{s.t.} \quad h(x, \zeta) \leq 0, \quad \forall \zeta \in Z \subseteq \mathbb{R}^k$$

- x : opt var; $X \subseteq \mathbb{R}^n$: nonempty closed convex set
- ζ : uncertain parameter; $Z \subseteq \mathbb{R}^k$: uncertainty set
- $c^\mathsf{T} x$: linear cost function without uncertainty (wlog)
- $h(x, \zeta) : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$: uncertain inequality constraint function
- Assumption: $h(x, \zeta)$ convex (and hence continuous) in x for every uncertain parameter $\zeta \in Z$

Interpretation: Choose an optimal x^* that satisfies the inequality constraint $h(x^*, \zeta) \leq 0$ for all possible uncertainty realization $\zeta \in Z$

Scenario opt: motivation

Robust optimization

Consider

$$\min_{x \in X \subseteq \mathbb{R}^n} \quad c^\mathsf{T} x \quad \text{s.t.} \quad h(x, \zeta) \leq 0, \quad \forall \zeta \in Z \subseteq \mathbb{R}^k$$

- Semi-infinite program: finite #optimization variables $x \in X$, possibly infinite #constraints
- Generally intractable
- For special cases of uncertainty set Z, robust program has finite convex reformulation which is tractable, e.g. robust LP, robust SOCP, robust SDP

Scenario opt: motivation

Chance constrained optimization

Consider

$$\min_{x \in X \subseteq \mathbb{R}^n} \quad c^\mathsf{T} x \quad \text{s.t.} \quad \mathbb{P}\left(\zeta : h(x,\zeta) \leq 0\right) \;\geq\; 1 - \epsilon$$

- $\mathbb{P}$: probability measure, $\epsilon \in [0,1]$

- Less conservative than robust optimization: allows constraint violation with probability $< 1 - \epsilon$
- Need to know distribution $\mathbb{P}$ of ζ
- Generally intractable

Scenario opt: motivation

Consider

$$\text{RCP}: \quad c^*_{\text{RCP}} := \min_{x \in X \subseteq \mathbb{R}^n} c^{\mathsf{T}} x \quad \text{s.t.} \quad h(x, \zeta) \leq 0,\ \zeta \in Z \subseteq \mathbb{R}^k$$

$$\text{CCP}(\epsilon): \quad c^*_{\text{CCP}}(\epsilon) := \min_{x \in X \subseteq \mathbb{R}^n} c^{\mathsf{T}} x \quad \text{s.t.} \quad \mathbb{P}\left(h(x, \zeta) \leq 0\right) \geq 1 - \epsilon$$

- X : nonempty closed convex set
- $h : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$: convex (and hence continuous) in x for every uncertain parameter $\zeta \in Z$

Scenario opt: motivation

Consider

$$\text{RCP}: \quad c^*_{\text{RCP}} := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad h(x,\zeta) \leq 0,\ \zeta \in Z \subseteq \mathbb{R}^k$$

$$\text{CCP}(\epsilon): \quad c^*_{\text{CCP}}(\epsilon) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad \mathbb{P}\left(h(x,\zeta) \leq 0\right) \geq 1-\epsilon$$

$$\text{CSP}(N): \quad c^*_{\text{CSP}}(N) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad h(x,\zeta^i) \leq 0,\ i = 1,\ldots,N$$

- $\left(\zeta^1, \ldots, \zeta^N\right)$: independent random samples each according to $\mathbb{P}$

Convex scenario opt

Comparison

Consider

$$\text{RCP}: \quad c^*_{\text{RCP}} := \min_{x \in X \subseteq \mathbb{R}^n} c^{\mathsf{T}} x \quad \text{s.t.} \quad h(x,\zeta) \leq 0,\ \zeta \in Z \subseteq \mathbb{R}^k$$

$$\text{CCP}(\epsilon): \quad c^*_{\text{CCP}}(\epsilon) := \min_{x \in X \subseteq \mathbb{R}^n} c^{\mathsf{T}} x \quad \text{s.t.} \quad \mathbb{P}\left(h(x,\zeta) \leq 0\right) \geq 1-\epsilon$$

$$\text{CSP}(N): \quad c^*_{\text{CSP}}(N) := \min_{x \in X \subseteq \mathbb{R}^n} c^{\mathsf{T}} x \quad \text{s.t.} \quad h(x,\zeta^i) \leq 0,\ i = 1,\ldots,N$$

- RCP : deterministic, semi-infinite, generally computational hard, conservative (safe)
- CCP(ϵ) : deterministic, generally computationally hard, less conservative, need $\mathbb{P}$
- CSP(N) : randomized, finite convex program for each realization of $\zeta := \left(\zeta^1, \ldots, \zeta^N\right)$, less conservative, only needs samples under $\mathbb{P}$ (not necessarily $\mathbb{P}$ itself), much more practical

Convex scenario opt

Consider

$$\text{RCP}: \quad c^*_{\text{RCP}} := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad h(x,\zeta) \leq 0,\ \zeta \in Z \subseteq \mathbb{R}^k$$

$$\text{CCP}(\epsilon): \quad c^*_{\text{CCP}}(\epsilon) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad \mathbb{P}\left(h(x,\zeta) \leq 0\right) \geq 1-\epsilon$$

$$\text{CSP}(N): \quad c^*_{\text{CSP}}(N) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad h(x,\zeta^i) \leq 0,\ i = 1,\ldots,N$$

Study 3 questions on CSP(N):

- Violation probability : how likely is the random solution x^*_N of CSP(N) feasible for CCP(ϵ)?
- Sample complexity : what is min N for x^*_N to be feasible for CCP(ϵ) in expectation or probability?
- Optimality guarantee : how close is the min cost $c^*_{\text{CSP}}(N)$ to the min costs $c^*_{\text{CCP}}(\epsilon)$ and c^*_{RCP} ?

Assumption

$$\text{CSP}(N): \qquad c^*_{\text{CSP}}(N) \;:=\; \min_{x \in X \subseteq \mathbb{R}^n} \; c^{\mathsf{T}} x \quad \text{s.t.} \quad h(x, \zeta^i) \leq 0,\; i = 1, \ldots, N$$

Assumption 1

- For each $\zeta \in Z, h(x, \zeta)$ is convex and continuous in x so that X_ζ is a closed convex set
- For each integer $N \geq n$ and each realization of $\zeta := \left(\zeta^1, \ldots, \zeta^N\right)$, feasible set of CSP$(N)$ has a nonempty interior. Moreover CSP(N) has a unique optimal solution x^*_N

Violation probability

Definition

Let $X_\zeta := \{x \in X \subseteq \mathbb{R}^n : h(x,\zeta) \leq 0\}$

Violation probability: $V(x) := \mathbb{P}\left(\left\{\zeta \in Z : x \notin X_\zeta\right\}\right)$

- For fixed $x \in X$, $V(x)$ is a deterministic value in $[0,1]$
- CCP(ϵ) is: $c^*_{\text{CCP}}(\epsilon) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad V(x) \leq \epsilon$
- For CSP(N), optimal solution x_N^* is a random variable under product measure $\mathbb{P}^N$
- Violation probability $V\left(x_N^*\right)$ of x_N^* is therefore a random variable under $\mathbb{P}^N$, taking value in $[0,1]$
- $V\left(x_N^*\right)$ may be smaller or greater than ϵ, i.e., x_N^* may or may not be feasible for CCP(ϵ)
- Goal: derive tight upper bounds on expected value and tail probability of $V\left(x_N^*\right)$

Violation probability

Uniformly supported problem

Definition

Consider CSP(N)

1. Given a realization $(\zeta^1, \ldots, \zeta^N) \in Z^N$, a constraint X_{ζ^i} is a support constraint for CSP(N), wrt $(\zeta^1, \ldots, \zeta^N)$, if its removal changes the optimal solution, i.e., $c^\mathsf{T} x_N^* \neq c^\mathsf{T} x_{N\setminus i}^*$
2. CSP(N) is uniformly supported with $s \geq 0$ support constraints if every realization of $(\zeta^1, \ldots, \zeta^N) \in Z^N$ contains exactly s support constraints for CSP(N) a.s. It is fully supported if $s = n$.

Violation probability

Uniformly supported problem

Definition

Consider CSP(N)

1. Given a realization $(\zeta^1, \ldots, \zeta^N) \in Z^N$, a constraint X_{ζ^i} is a support constraint for CSP(N), wrt $(\zeta^1, \ldots, \zeta^N)$, if its removal changes the optimal solution, i.e., $c^\top x_N^* \neq c^\top x_{N\setminus i}^*$
2. CSP(N) is uniformly supported with $s \geq 0$ support constraints if every realization of $(\zeta^1, \ldots, \zeta^N) \in Z^N$ contains exactly s support constraints for CSP(N) a.s. It is fully supported if $s = n$.

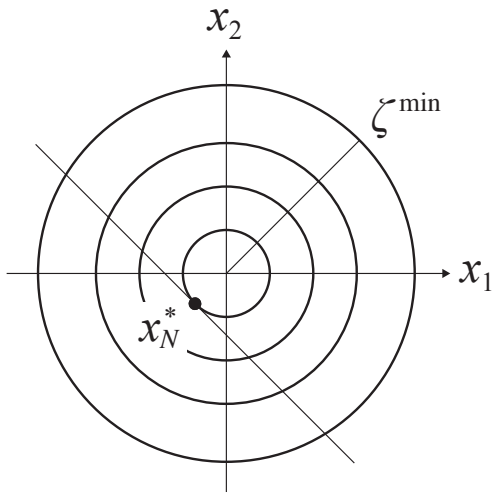
- A support constraint must be active at x_N^*; the converse may not hold.
- For uniformly supported problem with $s \geq 1$ support constraints, $\mathbb{P}(\zeta^i = \zeta^j) = 0$
- Since optimal solutions are unique, $c^\top x_N^* \neq c^\top x_{N\setminus i}^*$ if and only if $x_N^* \neq x_{N\setminus i}^*$
- **Lemma**: The number of support constraints for CSP(N) is at most n

Examples

Uniformly supported problem

CSP(N):

$$\min_{x\in\mathbb{R}^n} \sum_i x_i \quad \text{s.t.} \quad \|x\|_2^2 \le \zeta^i, \quad i = 1,\dots,N$$



For almost every realization of $\zeta := \left(\zeta^i, i = 1,\dots,N\right)$

#support constraints $= 1 < n$

Unique optimal solution $x_N^* := \left(-\sqrt{\zeta^{\min}/n}, \dots, -\sqrt{\zeta^{\min}/n}\right)$

where $\zeta^{\min} := \min_i \zeta^i$

Violation probability

Expected value

Theorem [Calafiore & Campi 2005; Calafiore 2009]

Suppose Assumption 1 holds. For any $N \geq n$

1. Then $$E^N\left(V\left(x_N^*\right)\right) = \mathbb{P}^{N+1}\left(x_N^* \notin X_{\zeta^{N+1}}\right) \leq \frac{n}{N+1}$$

2. If CSP$(N+1)$ is uniformly supported with $0 \leq s \leq n$ support constraints then

$$E^N\left(V\left(x_N^*\right)\right) = \mathbb{P}^{N+1}\left(x_N^* \notin X_{\zeta^{N+1}}\right) = \frac{s}{N+1}$$

- Upper bound is tight for uniformly supported problems
- Improved bound: $E^N(V(x_N^*)) \leq \frac{s_{N+1}^{\max}}{N+1}$

G. C. Calafiore and M. C. Campi, "Uncertain convex programs: Randomized solutions and confidence levels," *Math. Program.*, 2005
G. C. Calafiore. "A note on the expected probability of constraint violation in sampled convex programs." *IEEE CCA & ISIC*, 2009.

Violation probability

Tail probability

Theorem [Campi, Garatti 2008]

Suppose Assumption 1 holds. For any $N \geq n$

1. Then $\mathbb{P}^N\left(V\left(x_N^*\right) > \epsilon\right) \leq \sum_{i=0}^{n-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}$ Binomial tail

2. If CSP$(N+1)$ is uniformly supported with $1 \leq s \leq n$ support constraints then

$$\mathbb{P}^N\left(V\left(x_N^*\right) > \epsilon\right) = \sum_{i=0}^{s-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}$$

- Upper bound is tight for uniformly supported problems
- Improved bound: $\mathbb{P}^N(V(x_N^*) > \epsilon) \leq \sum_{i=0}^{s_*^{\max}-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}$

M. C. Campi and S. Garatti. "The exact feasibility of randomized solutions of uncertain convex programs." *SIAM J. Optim.*, 200

Violation probability

Summary: improved bounds

Suppose Assumption 1 holds.

1. $E^N\left(V\left(x_N^*\right)\right) \leq \frac{s_{N+1}^{\max}}{N+1}$

2. $\mathbb{P}^N\left(V\left(x_N^*\right) > \epsilon\right) \leq \sum_{i=0}^{s_*^{\max}-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}$ Binomial tail

- Binomial tail decreases rapidly as N increases
- Bounds are tight for uniformly supported problems with $0 \leq s \leq n$ support constraints
- Bounds depend only on (n, N) and ϵ.
- Not on details of cost function $c^\mathsf{T} x$, constraint function $h(x, \zeta)$, probability measure $\mathbb{P}$; they determine if the problem is fully supported and hence tightness of the bounds

Sample complexity

Corollary

Suppose Assumption 1 holds. For any ϵ, β in $[0,1]$

1. $E^N\left(V(x_N^*)\right) \leq \beta$ if $N \geq (n/\beta) - 1$

2. $\mathbb{P}^N\left(V(x_N^*) > \epsilon\right) \leq \beta$ if $N \geq N(\epsilon, \beta)$ where

$$N(\epsilon, \beta) \;:=\; \min\left\{ N \;:\; \sum_{i=0}^{n-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \;\leq\; \beta \right\}$$

Optimality guarantee

Consider

$$\text{RCP}: \quad c^*_{\text{RCP}} := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad h(x,\zeta) \leq 0,\ \zeta \in Z \subseteq \mathbb{R}^k$$

$$\text{CCP}(\epsilon): \quad c^*_{\text{CCP}}(\epsilon) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad \mathbb{P}\left(h(x,\zeta) \leq 0\right) \geq 1-\epsilon$$

$$\text{CSP}(N): \quad c^*_{\text{CSP}}(N) := \min_{x \in X \subseteq \mathbb{R}^n} c^\mathsf{T} x \quad \text{s.t.} \quad h(x,\zeta^i) \leq 0,\ i = 1,\ldots,N$$

Study 3 questions on CSP(N):

- Violation probability : how likely is the random solution x^*_N of CSP(N) feasible for CCP(ϵ)?
- Sample complexity : what is min N for x^*_N to be feasible for CCP(ϵ) in expectation or probability?
- Optimality guarantee : how close is the min cost $c^*_{\text{CSP}}(N)$ to the min costs $c^*_{\text{CCP}}(\epsilon)$ and c^*_{RCP} ?

Stochastic OPF

Consider

$$\min_{x \in \mathbb{R}^n} \; f(x) \qquad \text{s.t.} \qquad h(x, \zeta) \leq 0$$

where ζ is a parameter

Brief introduction to theory of stochastic optimization

- Most stochastic optimization problems are intractable (e.g., nonconvex, nonsmooth)
- Explains 2 ideas to deal with uncertainty: scenario optimization, two-stage optimization

Stochastic linear program

With fixed recourse

$$\min_{x \in \mathbb{R}^{n_1}} \quad f(x) + Q(x) \qquad \text{s.t.} \qquad Ax = b,\ x \in K$$

where $Q(x) := E_\zeta \left(\min_{y(\omega) \geq 0} q^\mathsf{T}(\omega) y(\omega) \quad \text{s.t.} \quad Wy(\omega) = h(\omega) - T(\omega)\, x \right)$

1st-stage problem

- Cost function $f : \mathbb{R}^{n_1} \to \mathbb{R}$ is real-valued convex, K is closed convex cone
- Parameters (f, A, b, K) are certain

2nd-stage (semi-infinite) problem: linear program for each ω

- Recourse action $y(\omega)$ adapts to each realized $\omega \in \Omega$
- Recourse matrix W is independent of ω (i.e., fixed recourse)
- Uncertain parameter $\zeta := \zeta(\omega) := ((q(\omega), T(\omega), h(\omega)) \in \mathbb{R}^k$
- uncertainty set $Z := \{\zeta(\omega) \in \mathbb{R}^k : \omega \in \Omega\}$

Stochastic linear program

With fixed recourse

$$\min_{x \in \mathbb{R}^{n_1}} \quad f(x) + Q(x) \qquad \text{s.t.} \qquad Ax = b,\ x \in K$$

where $Q(x) := E_\zeta \left(\min_{y(\omega) \geq 0} q^\mathsf{T}(\omega) y(\omega) \quad \text{s.t.} \quad Wy(\omega) = h(\omega) - T(\omega)\, x \right)$

- $Q(x)$: recourse function (or 2nd-stage expected value function)
- $Q(x)$ can be extended real-valued function and nondifferentiable
- $Q(x) = \infty$ if second-stage problem is infeasible (e.g., day-ahead schedule leads to insufficient supply when outages occur in real time)

This would have been a simple conic program, but for the recourse function $Q(x)$

Example

Generation scheduling

Schedule 2 generators with same generation capacity $[0,a]$ to meet random demand $\zeta(\omega)$

1. Slow but cheap generator must be scheduled before $\zeta(\omega)$, at level $x \in [0,a]$ at unit cost c_1
2. Fast but expensive generator can be scheduled after $\zeta(\omega)$, at level $y(\omega) := y(\zeta(\omega)) \in [0,a]$ at unit cost $c_2 > c_1$
3. Suppose $\zeta(\omega) = a + \epsilon$ with prob. p, and $\zeta(\omega) = a - \epsilon$ with prob. $1 - p$

Goal: choose $(x, y(\omega))$ to meet random demand $\zeta(\omega)$ at minimum expected cost:

$$f^* := \min_{x \in \mathbb{R}} c_1 x + Q(x) \quad \text{s.t.} \quad 0 \leq x \leq a$$

where $Q(x) := E_\zeta \tilde{Q}(x, \zeta)$ and

$$\tilde{Q}(x, \zeta) := \min_{0 \leq y(\omega) \leq a} c_2 y(\omega) \quad \text{s.t.} \quad x + y(\omega) = \zeta(\omega)$$

What is the optimal solution ?

Example

Generation scheduling

Therefore

$$f^* \quad := \quad \min_{x \in \mathbb{R}} \ (c_1 - c_2)x + c_2(a + \epsilon(2p - 1)) \quad \text{s.t.} \quad \epsilon \leq x \leq a - \epsilon$$

Solution:

Since $c_2 > c_1$, optimal solution is:

$$x^* = a - \epsilon, \qquad f^* = c_1(a - \epsilon) + 2c_2\epsilon p$$

Therefore

1. The cheap generator always produces at the lower level $a - \epsilon$ of the random demand
2. The expensive generator will pick up the slack, 2ϵ with probability p

Takeaway

OPF underlies numerous power system planning, operation, market applications

- It is nonconvex, NP-hard
- Traditional solution approaches: approximations, local algorithms, relaxations

New challenges: uncertainty

- Generally intractable
- Large number of constraints and vars even with convex reformulation (e.g., scenario opt, two-stage opt)
- Uncertainty $\Longrightarrow$ need to close the loop, real-time decisions

This motivates a 4th solution approach: ML for OPF (Parts 2-4)

Main reference

Power System Analysis

Analytical tools and structural properties

STEVEN H. LOW

California Institute of Technology
slow@caltech.edu

DRAFT: https://netlab.caltech.edu/book_reg/

1. OPF: basic formulation [Ch 9]
2. OPF: unbalanced radial network [Ch 17, 18]
3. Solution approaches [Ch 9]
4. New challenge: uncertainty [Ch 13]

Outline

- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems

- Concluding remarks and open challenges

Observation

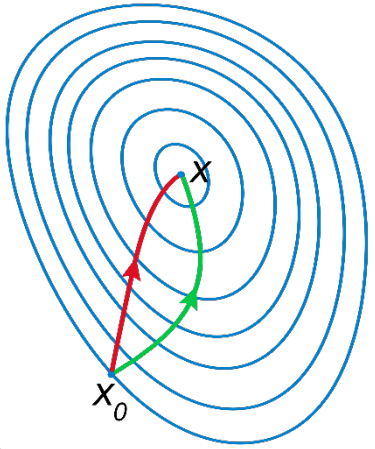
- OPFs are important, but hard to solve in real time
  - They can be non-convex, NP-hard, and involve 1M variables
- Even simplified DC-OPF formulations may incur high complexity
  - e.g., solving DC-OPF by interior-point methods can require $O(N^4)$ time
- How to solve important, hard problems fast?

Machine learning for constrained optimization

Constrained Optimization

$$\min_{x} \quad f(x, z)$$
$$s.t. \quad g_i(x, z) = 0, \ i = 1, \ldots . n$$
$$\mathrm{h}_j(x, z) \leq 0, \ j = 1, \ldots . m$$

z: input parameter vector
x: decision variable vector



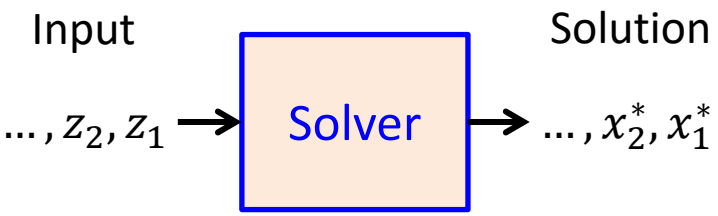
Gradient descent (green)
Newton's method (red)

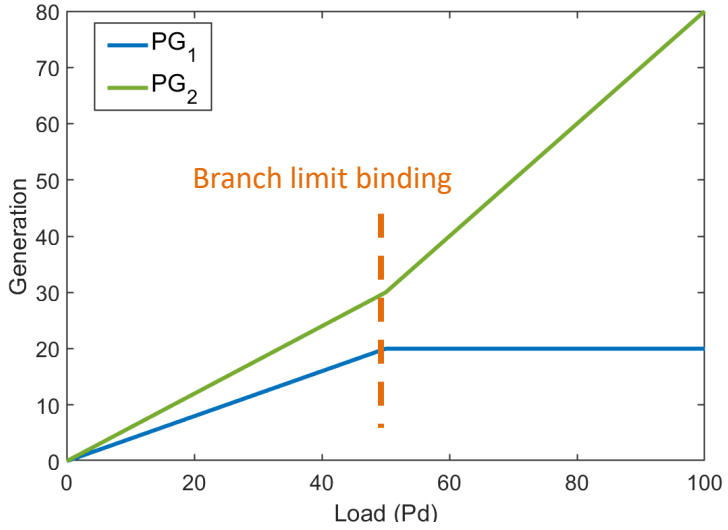
- Tremendous applications; many off-the-shelf solvers
- Given z, solvers apply iterative strategies to pursue opt
  - E.g., the gradient descent method (with projecti
    $$x(t+1) = x(t) - \alpha$$
  - E.g., the Newton-Raphson method that u curvature information

Universal, optimal, feasible but high run-time complexity

Picture source: wiki

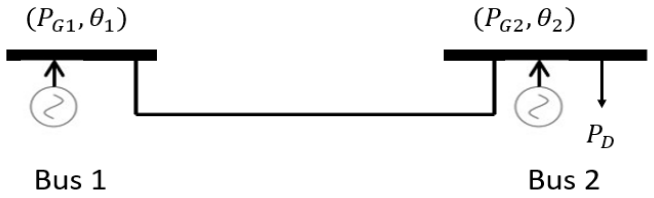
An Input-Solution Mapping Perspective



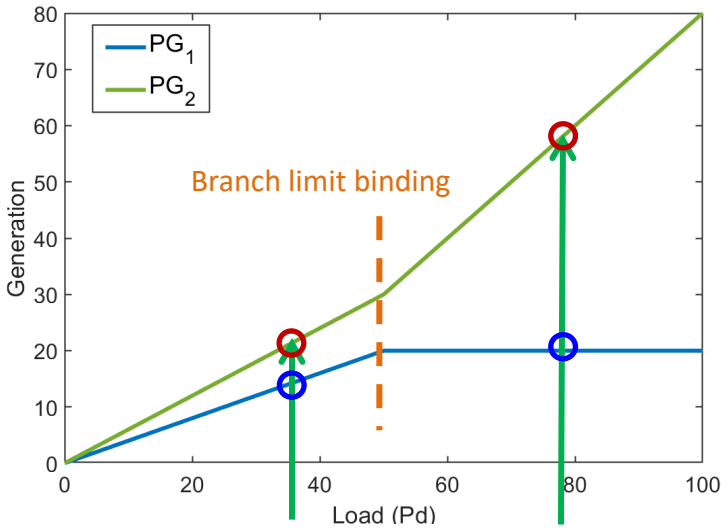


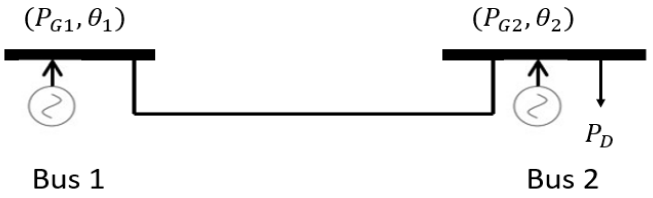
- A solver implicitly characterizes an input-solution mapping for a problem

- Example: The load-generation mapping for a DC-OPF problem over a 2-bus instance



New Machine Learning Viewpoint





- Learn the input-solution mapping for a given problem
- Pass inputs through the learned mapping for solutions
  - No iterative updates needed
  - Trade learning complexity for low run-time complexity
- Learn once, solve many times
- **Q**: can we learn such a mapping?

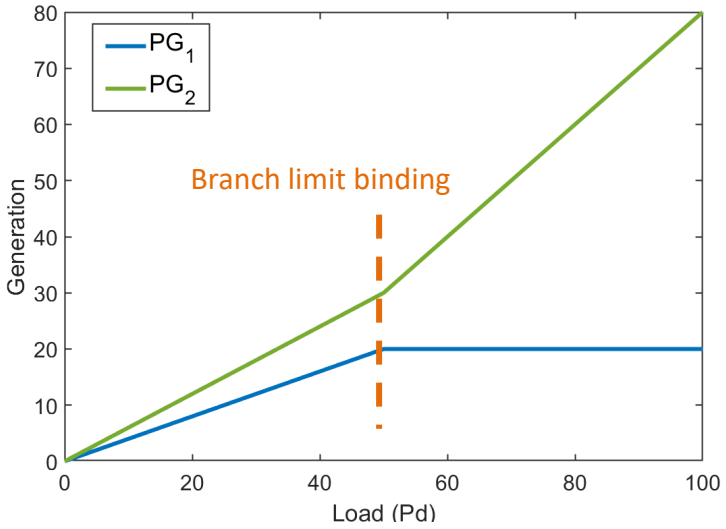
[1] X. Pan, T. Zhao and M. Chen, "DeepOPF: Deep Neural Network for DC Optimal Power Flow", SmartGridComm, 2019. (arXiv:1905.04479, May 11th, 2019) The Journal version for SC-DCOPF appears in IEEE Transactions on Power Systems in 2021.

Continuous Mapping upon Unique Solution

$$\min_{x} \quad f(x,z)$$
$$s.t. \quad g_i(x,z)=0, \ i=1,\dots.n$$
$$\mathrm{h}_j(x,z)\le 0, \ j=1,\dots.m$$

z: input parameter vector
x: decision variable vector

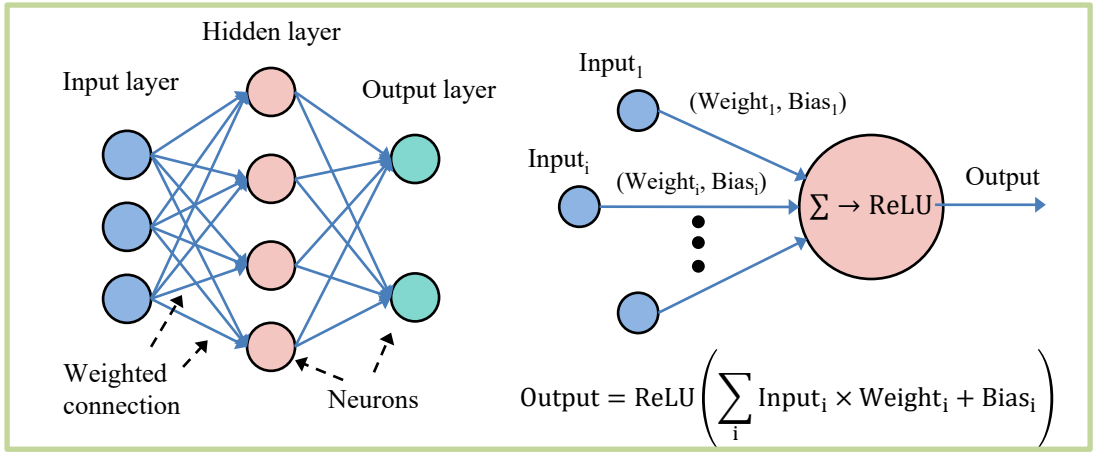
Continuous Mapping Theorem. *[1][2]*
For continuous f, g, h, if the input domain D is compact and the optimal solution $x^*(z)$ is unique for every z $\in$ D, then the input-solution mapping $z \to x^*(z)$ is continuous.

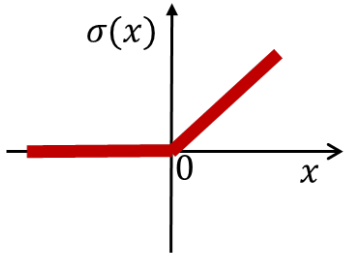


DC-OPF is a quadratic problem with a unique optimal solution

[1] Maximum Theorem in Chapter 6, Section 3, *Claude Berge, "Topological Spaces". Oliver and Boyd. p. 116. (1963)*
[2] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," arXiv preprint arXiv:2007.01002, 2020. Journal version in IEEE Systems Journal, 2023.

NN Can Approximate Continuous Mapping





- **Theorem** [1-3]: With suitable non-polynomial activation functions, feedforward networks can approximate "any" function/mapping arbitrarily well, given enough neurons.
  - Any function whose p-th power of the absolute value is Lebesgue integrable
  - Activation function is bounded, non-constant, and continuous
  - Even by using single (hidden) layer NNs
- Results extended to RNN, CNN, ResNet, Transformers, etc.

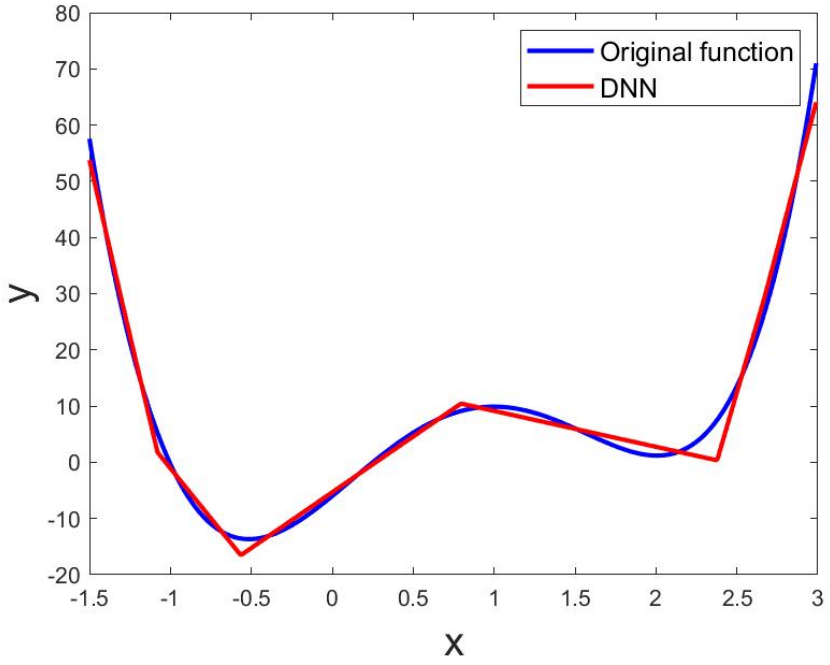
[1] K. Hornik, "Approximationcapabilitiesofmultilayerfeedforwardnetworks," Neural networks, vol.4, no.2, pp. 251–257,1991.
[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function", Math. Control Signals Systems, 2(4):303–314, 1989.
[3] Pinkus, Allan. "Approximation theory of the MLP model in neural networks." Acta numerica, 8 :143-195, 1999.

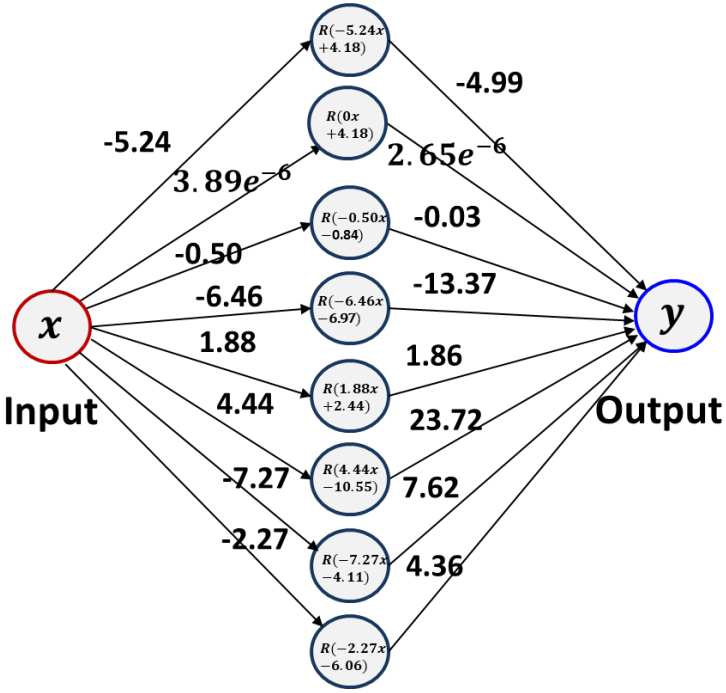
An Illustrative Example

- Approximating a nonlinear function with a small ReLU neural network

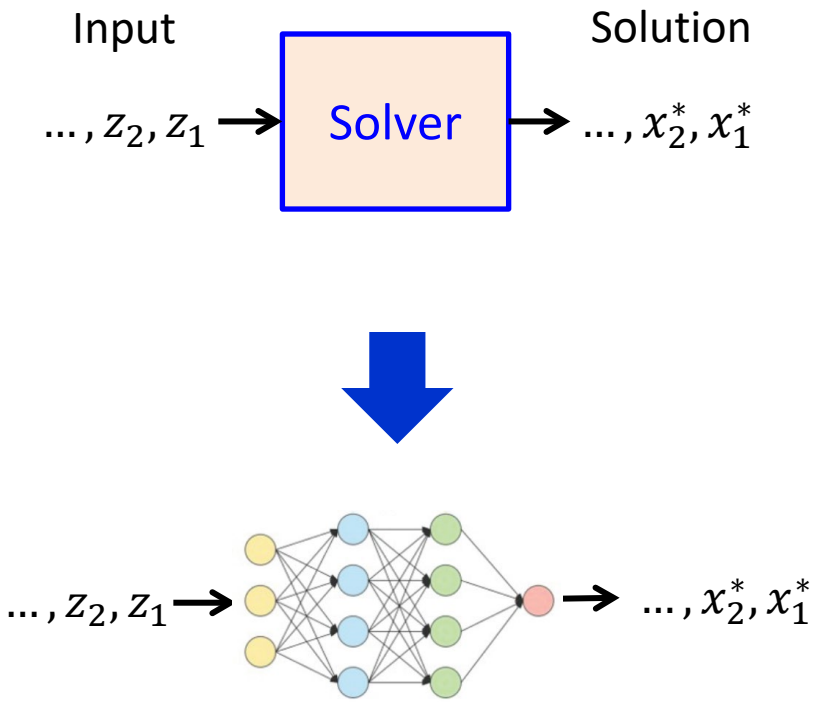
$$y = 0.3x^5 + 4.8x^4 - 18.8x^3 + 6x^2 + 23.6x - 6$$

using an 8-node, single hidden-layer ReLU NN





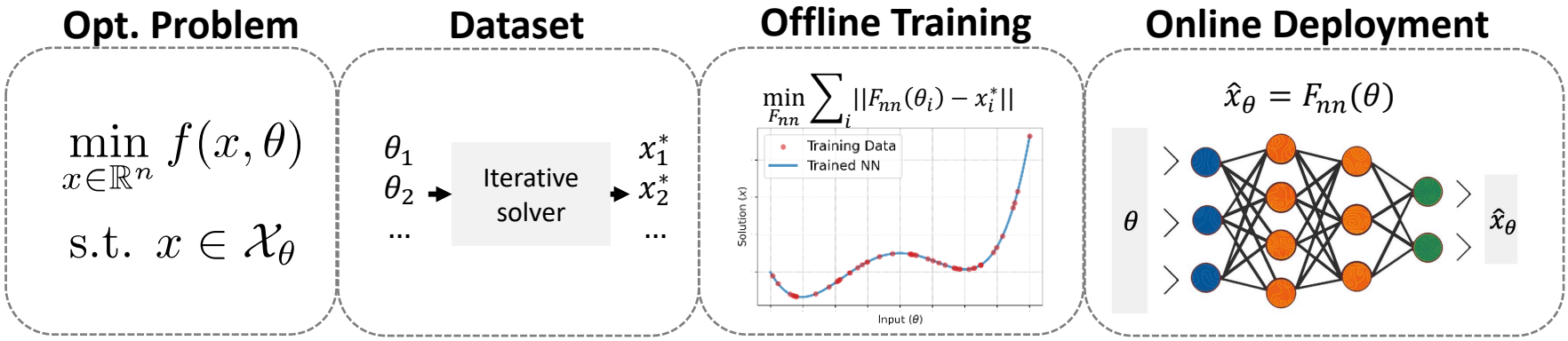
Machine Learning as an Amortized Solver!



- Learn the input-solution mapping for a given problem
- Pass inputs through the learned mapping for solutions
  - No iterative updates needed
  - Trade learning complexity for low run-time complexity
- **Yes, NN can learn the input-solution mapping**
  - Learning complexity is amortized if the problem is solved repeatedly, e.g., OPF

[1] X. Pan, T. Zhao and M. Chen, "DeepOPF: Deep Neural Network for DC Optimal Power Flow", SmartGridComm, 2019. (arXiv:1905.04479, May 11th, 2019) The Journal version for SC-DCOPF appears in IEEE Transactions on Power Systems in 2021.

One-time Training Cost for Fast Inference

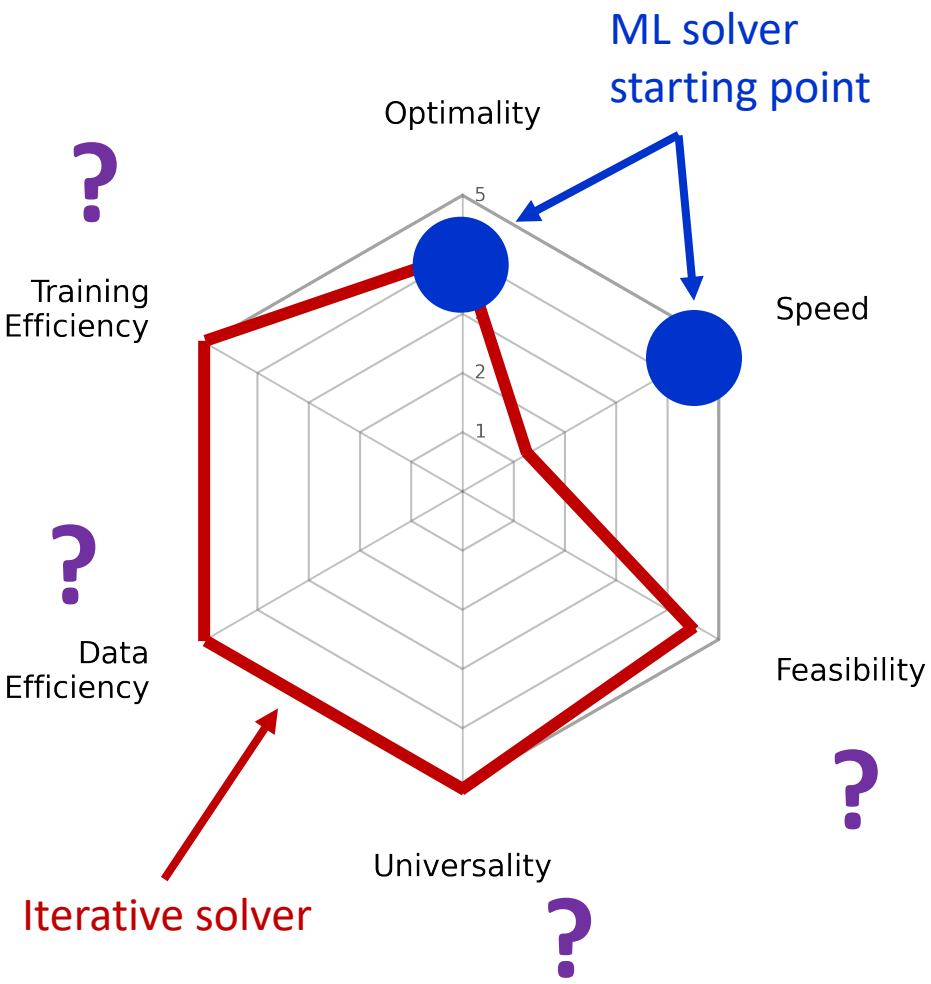


- **Offline training:** Learn the input-to-solution mapping from solved problem instances

- **Online deployment:** For a new input, use the learned mapping to generate a solution directly

[1] X. Pan, T. Zhao and M. Chen, "DeepOPF: Deep Neural Network for DC Optimal Power Flow", SmartGridComm, 2019. (arXiv:1905.04479, May 11th, 2019) The Journal version for SC-DCOPF appears in IEEE Transactions on Power Systems in 2021.

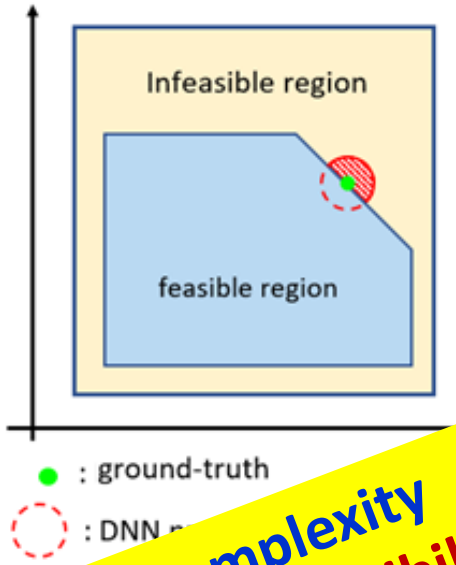
Six Dimensions for Comparing Solvers



- **Feasibility**: solution feasibility
- **Optimality**: solution optimality
- **Speed**: run-time solution speed
- **Universality**: one solver for all problems
- **Data efficiency**: minimum data-preparation effort
- **Training efficiency**: minimum training effort

Challenge and Limitation

- NN solutions may not be feasible
  - can be critical; e.g., violating branch flow limit can cause cascading failure in power grids
- May incur high data/training complexity
- NN may not generate the right solution (even with perfect training)
- One mapping per problem



Low run-time complexity but not universal, no feasibility guarantee, and high training/data complexity

ML for Constrained Optimization in Power System Operation

- Application in OPF
  - Assist conventional solvers, e.g., [1,2]
  - Directly generate solutions (e.g., [3], and upcoming slides)
- Other Applications
  - Frequency, voltage, and stability control [4-6]
  - Network reconfiguration and restoration [7, 8]
  - Economic dispatch, unit commitment, and scheduling [9-11]

[1] Y. Ng, S. Misra, L. A. Roald and S. Backhaus, “Statistical Learning for DC Optimal Power Flow”, in Proc. IEEE PSCC, Dublin, Ireland, Jun. 11 - 15, 2018.
[2] T. Falconer, L. Mones, “Leveraging Power-Grid Topology in ML-Assisted OPF,” IEEE TPWRS, 38(3):2234–2246, 2023.
[3] X. Pan, T. Zhao, and M. Chen, “Deepopf: Deep neural network for DC optimal power flow,” in Proc. IEEE SmartGridComm, Beijing, China. 2019.
[4] W. Cui and B. Zhang, "Lyapunov-Regularized Reinforcement Learning for Power System Transient Stability," in IEEE Control Systems Letters, vol. 6, pp. 974-979, 2022.
[5] W. Cui, Y. Jiang, and B. Zhang, “Reinforcement learning for optimal primary frequency control: A Lyapunov approach,” IEEE TPWRS, 38(2), 2022.
[6] T. Zhao, J. Wang, X. Lu, and Y. Du, “Neural Lyapunov control for power system transient stability: A deep learning-based approach,” IEEE Trans. Power Syst., vol. 37, no. 2, pp. 955–966, Mar. 2022.
[7] Y. Gao, J. Shi, W. Wang and N. Yu, "Dynamic Distribution Network Reconfiguration Using Reinforcement Learning," in Proc. IEEE SmartGridComm, Beijing, China. 2019.
[8] H. Gao, R. Wang, S. He, L. Wang, J. Liu, Z. Chen, “A Cloud-Edge Collaboration Solution for Distribution Network Reconfiguration Using Multi-Agent Deep Reinforcement Learning”, IEEE TPWRS, 39(2), 2024.
[9] F. Hasan, A. Kargarian, "Topology-aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch", TPWRS, 2022.
[10] A. Xavier, F. Qiu, and S. Ahmed, “Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems”, Journal of Computing, 2022
[11] T. B. Lopez-Garcia, J. A. Dominguez-Navarro, “Optimal Power Flow With Physics-Informed Typed Graph Neural Networks, IEEE TPWRS, 2025

Outline

- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

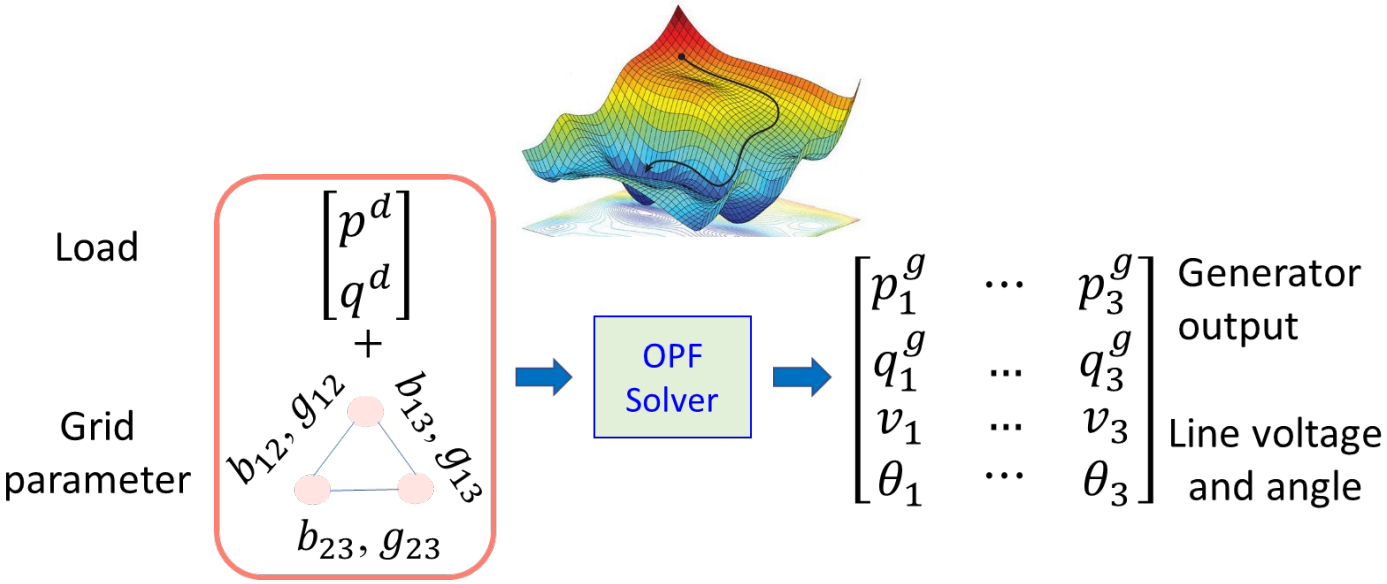
- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems

- Concluding remarks and open challenges

Machine Learning for Solving OPF Problems

Input: load, renewable, grid topology and state
Solution: generation, voltage, and phase

Recall: OPF for Grid Operating Set Points



- Recall: The Optimal Power Flow (OPF) Problem is to determine the outputs of generators to
  - Satisfy the power-balance constraints (reliability)
  - Minimize the overall generation cost (efficiency)
  - Satisfy the line and voltage limits (safety)

[1] J. Carpentier, "Contribution to the economic dispatch problem," Bulletin de la Societe Francoise des Electriciens, vol. 3, no. 8, 1962.
[2] Cain M B, O'neill R P, Castillo A. History of optimal power flow and formulations. Federal Energy Regulatory Commission, 2012, 1: 1-36.

Recall: Solving AC-OPF is Challenging

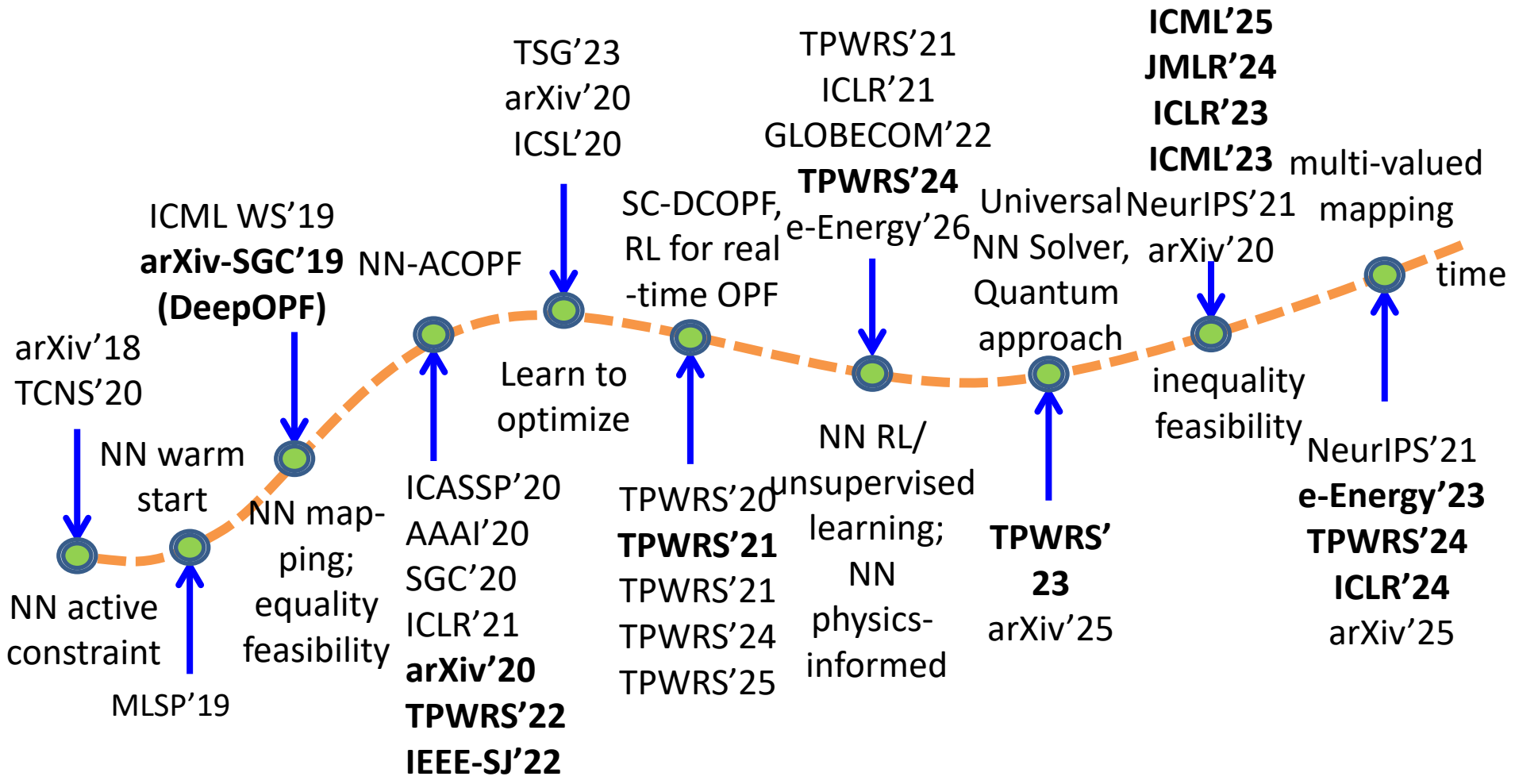
- AC-OPF problem is non-convex and NP-hard, difficult to solve in real-time
  - Practical OPF can involve more than 1M variables
- To accommodate renewable, market operators need to solve OPFs every 5 minutes
  - Previously, every half day or every 2 hours
  - In the future, every one minute
- Operators often terminate iterative methods early, or resort to solve DC-OPF, both giving sub-optimal results
- 5% solution improvement saves $36B/year globally

[1] Bienstock D, Verma A. Strong NP-hardness of AC power flows feasibility. Operations Research Letters, 2019.
[2] Reddy S S, Bijwe P R. Day-ahead and real time optimal power flow considering renewable energy resources. International Journal of Electrical Power & Energy Systems, 2016, 82: 400-408.

Approaches

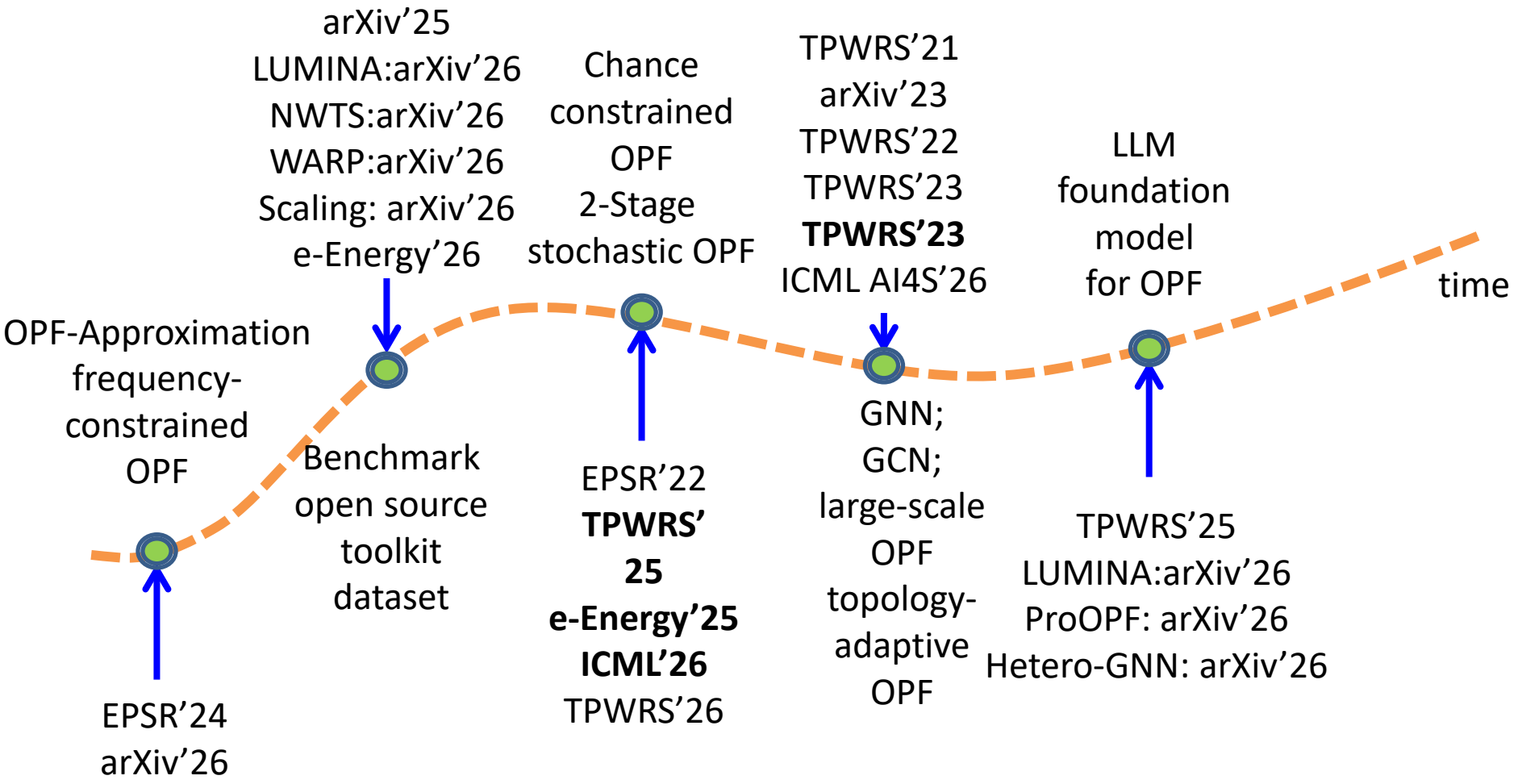
- General Newton-like iterative algorithms — Not scalable
- Linearization to solve OPF approximately — Inaccurate
- Convexification to solve OPF optimality — Only applicable to special cases
- Machine learning to solve OPF problems directly
  - Sub-percentage optimality loss (better than linearized OPF)
  - 1,5000x speedup for AC-OPF over a 2000-bus network
  - Approaches evaluated over actual RTE networks with 9,241 buses, actual Korea-4492 system, also realistic load profiles with 42% variation
- Machine learning to assist existing iterative solvers

Historical Roadmap



- Our works in bold font
- Wiki: https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki

Historical Roadmap (Cont.)



- Our works in bold font
- Wiki: https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki

Wiki and Overview Webpage

- A wiki page hosted by ACM SIGEnergy
  - https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki
- A wiki page hosted by Climate Change AI (on more general topics)
  - https://wiki.climatechange.ai/wiki/Welcome_to_the_Climate_Change_AI_Wiki
- An overview webpage by Letif Mones
  - https://invenia.github.io/blog/2021/10/11/opf-nn/
- Dataset or data generators for training NN for OPF problems
  - https://github.com/NREL/OPFLearn.jl
  - https://github.com/invenia/OPFSampler.jl/
- Machine learning for AC-OPF tutorial (with walk-through exercise)
  - https://colab.research.google.com/github/climatechange-ai-tutorials/optimal-power-flow/blob/main/AI_for_Optimal_Power_Flow.ipynb

Happenings at e-Energy'26

Self-Supervised Learning for Energy Informatics – Morning Session 1
Srinivasan Keshav, University of Cambridge
08:00 – 10:00 • KC306

Session C: Machine learning I
Chair: Wenqi Cui
15:45 – 17:30 • MaxBell
click for papers ▸

Session G: Optimal power flow
Chair: Nicolas Christianson
08:30 – 10:00 • MaxBell
click for papers ▸

Session H: Foundation models
Chair: Priya Donti
10:30 – 12:15 • MaxBell
click for papers ▸

Session F: Machine learning II
Chair: Hao Zhu
15:45 – 17:45 • MaxBell
click for papers ▸

Session I: Foundation models / Learning
Chair: Yize Chen
13:45 – 15:30 • MaxBell
click for papers ▸

Self-Supervised Learning for Energy Informatics – Morning Session 2
Srinivasan Keshav, University of Cambridge
10:30 – 12:00 • KC306

Machine Learning for Solving DC-OPF and SC-DCOPF Problems

(sharing in the backup slides)

Machine Learning for Solving AC-OPF Problems

- X. Pan, M. Chen, T. Zhao and S. H. Low, "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems", arXiv preprint arXiv:2007.01002, 2020.
- G. Neel, Z. Wang and A. Majumdar, "Machine Learning for AC Optimal Power Flow", In Proceedings of the 36th International Conference on Machine Learning Workshop, Long Beach, CA, USA, 2019.
- W. Huang, X. Pan, M. Chen, and S. H. Low, "DeepOPF-V: Solving AC-OPF Problems Efficiently", IEEE Transactions on Power Systems, vol. 37, no. 1, pp. 800 - 803, Jan. 2022.
- F. Fioretto, T. Mak, and P. V. Hentenryck, "Predicting AC Optimal Power Flows: Combining D
Learning and Lagrangian Dual Methods", AAAI, 2020.
- A. Zamzam and K. Baker, "Learning Optimal Solutions for Extremely Fast AC
IEEE SmartGridComm, 2020.
- P. L. Donti, D. Rolnick and J. Z. Kolter, "DC3: a learning meth
ICLR, 2021.
- W. Huang and M. Chen, "DeepOPF-NGT: A Fast U
Problems without Ground Truth", In Proceedings of
Learning Workshop, virtual conference, Jul. 23, 2021.
- M. Chatzos, T. W. K. Mak and P. Vanhentenryck, "Spa
AC-OPF Learning," in *IEEE Trans. on Power Systems, 2022*

Mainly focus on DNN here
GNN/LLM in Part III

Standard (Nonconvex) AC-OPF Formulation

- Minimizing generation cost to serve the load, with an accurate AC model

$$\min \quad \sum_{i\in\mathcal{N}_g} \left(\lambda_{i,2}P_{gi}^2 + \lambda_{i,1}P_{gi} + \lambda_{i,0}\right)$$

generation cost

$$\text{s.t.} \quad P_{gi} - P_{di} = \sum_{j:(i,j)\in\mathcal{E}} V_i V_j (g_{ij}\cos\theta_{ij} + b_{ij}\sin\theta_{ij}), \quad \forall i \in \mathcal{N},$$

$$Q_{gi} - Q_{di} = \sum_{j:(i,j)\in\mathcal{E}} V_i V_j (g_{ij}\sin\theta_{ij} - b_{ij}\cos\theta_{ij}), \quad \forall i \in \mathcal{N},$$

Nonconvex AC power flow equations

$$P_{gi}^{\min} \le P_{gi} \le P_{gi}^{\max}, \quad \forall i \in \mathcal{N}_g,$$

$$Q_{gi}^{\min} \le Q_{gi} \le Q_{gi}^{\max}, \quad \forall i \in \mathcal{N}_g,$$

$$V_i^{\min} \le V_i \le V_i^{\max}, \quad \forall i \in \mathcal{N}_g,$$

$$\theta_{ij}^{\min} \le \theta_{ij} \le \theta_{ij}^{\max}, \quad \forall (i,j) \in \mathcal{E},$$

Generation and voltage limit constraints

$$0 \le P_{ij}^2 + Q_{ij}^2 \le \left(S_{ij}^{\max}\right)^2, \quad \forall (i,j) \in \mathcal{E},$$

$$P_{ij} = g_{ij}\left(V_i^2 - V_i V_j \cos\theta_{ij}\right) - b_{ij} V_i V_j \sin\theta_{ij}, \quad \forall (i,j) \in \mathcal{E},$$

$$Q_{ij} = b_{ij}\left(-V_i^2 + V_i V_j \cos\theta_{ij}\right) - g_{ij} V_i V_j \sin\theta_{ij}, \quad \forall (i,j) \in \mathcal{E},$$

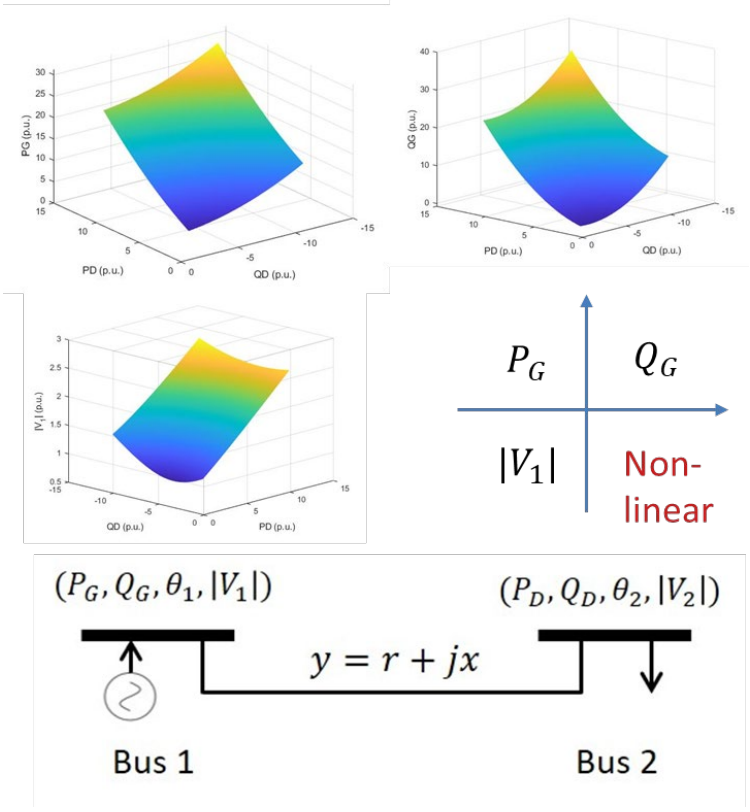
Nonconvex branch flow limit constraints

$$\text{var.} \quad P_{gi}, Q_{gi}, \forall i \in \mathcal{N}_g; V_i, \theta_i, \forall i \in \mathcal{N}.$$

J. Carpentier, "Contribution to the economic dispatch problem," Bulletin de la Societe Francoise des Electriciens, vol. 3, no. 8, pp. 431–447, 1962.

Load-Solution Mapping of AC-OPF

- **Theorem** [4]: Assume the load domain is compact, and the constraint and objective functions are continuous, then the input-solution mapping (can be single-valued or multi-valued) is continuous almost everywhere, with respect to corresponding distance metrics
  - AC-OPF has a unique solution in "typical" load regions, or radial network under certain conditions [1], or with monotonic power flow equations [2,3]



[1] S. H. Low, "Convex relaxation of optimal power flow—Parts I: Formulations and equivalence," IEEE Trans. Control Netw. Syst., vol. 1, no. 1, pp. 15–27, Mar. 2014.

[2] Park, S., Zhang, R.Y., Lavaei, J. and Baldick, R., '' Uniqueness of power flow solutions using monotonicity and network topology,'' IEEE Transactions on Control of Network Systems, 8(1), pp.319-330, 2020

[3] Dvijotham, K., Low, S. and Chertkov, M., ''Solving the power flow equations: A monotone operator approach,'' arXiv:1506.08472, 2015

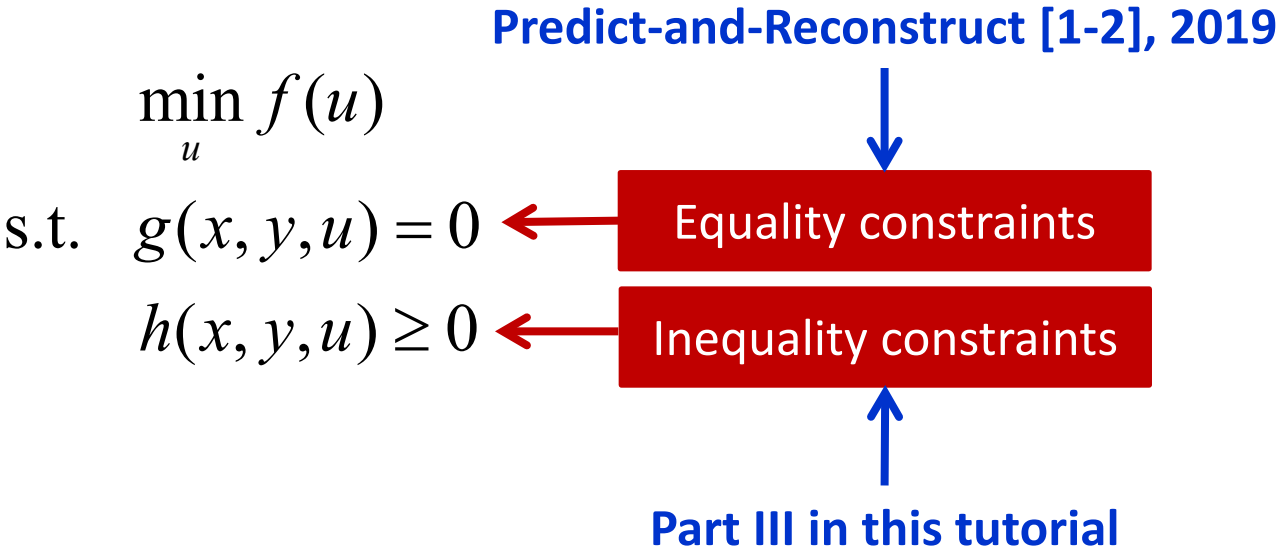
[4] X. Pan, M. Chen, T. Zhao and S. H. Low, "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems", arXiv preprint arXiv:2007.01002, 2020. Also in IEEE Systems Journal 2023.

Glance of the Challenges and Approaches

- NN solutions may be infeasible
  - Predict-and-reconstruct (PR2) / equation completion
  - Primal-dual training (utilizing KKT conditions)
  - Differentiable NN layers
  - More advanced techniques to be covered in **Part III**
- High training/data complexity for large-scale OPF problems
  - Un-/self- supervised learning, GNN, grid decomposition, compact learning, approximate data labels
- Non-universal: need one DNN per system configurati..., topology, admittance, generator on/off...
  - **Will be discussed in Part I**

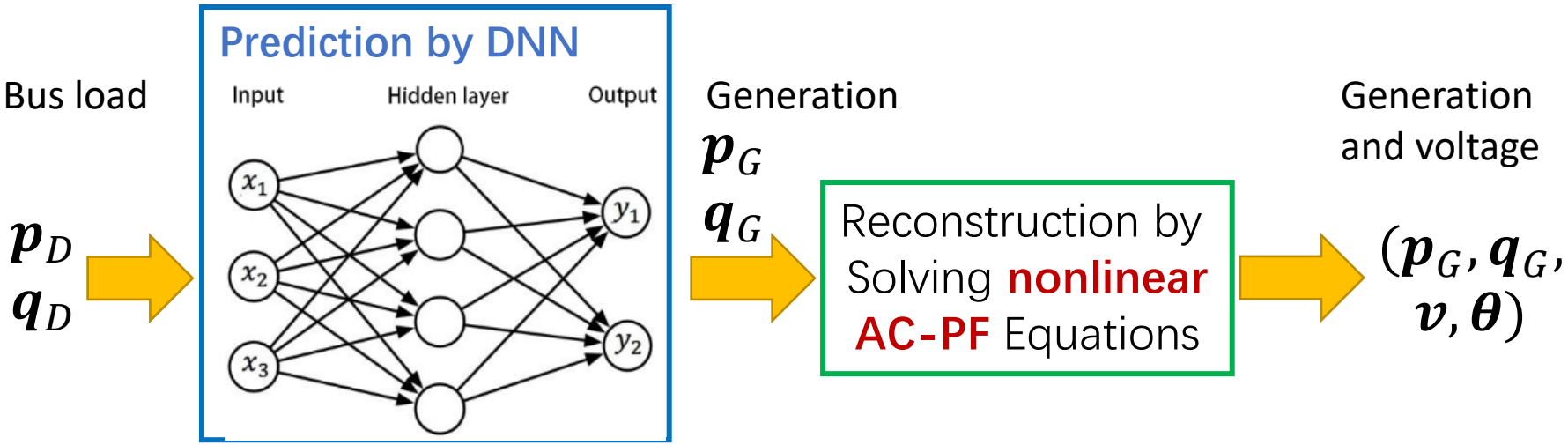
We will discuss the topics in color blue in this part

Ensuring NN Solution Feasibility



[1] X. Pan, T. Zhao and M. Chen, "DeepOPF: Deep Neural Network for DC Optimal Power Flow", SmartGridComm, 2019. (arXiv:1905.04479, May 11th, 2019) The Journal version for SC-DCOPF appears in IEEE Transactions on Power Systems in 2021.
[2] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems", IEEE Systems Journal, 2022. arXiv version in 2020.
[3] E. Liang, M. Chen, and S. H. Low, "Low Complexity Homeomorphic Projection to Ensure Neural-Network Solution Feasibility for Optimization over (Non-)Convex Set", ICML, 2023.
[4] E. Liang, M. Chen, and S. H. Low, "Homeomorphic Projection to Ensure Neural-Network Solution Feasibility for Constrained Optimization", JMLR, 2024

Predict and Reconstruct (PR2)

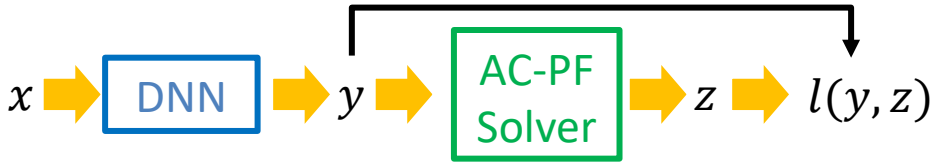


- **Predict-and-Reconstruct** (PR2) [1, 2]: predict $\boldsymbol{p}_G, \boldsymbol{q}_G$ and reconstruct $\boldsymbol{v}, \boldsymbol{\theta}$ by solving power flow equations (aka equation completion in [3])
  - Ensure power flow equality constraints (reliability)
  - Reduce #variables to predict: 99% reduction in a 2000-bus case [2]
  - Applicable to general constrained optimization problems

[1] X. Pan, T. Zhao, M. Chen, and S. Zhang, "DeepOPF: A Deep Neural Network Approach for Security-Constrained DC Optimal Power Flow", IEEE Transactions on Power Systems, 2021.
[2] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," arXiv preprint arXiv:2007.01002, 2020; also in IEEE Systems Journal 2023
[3] P. L. Donti, D. Rolnick and J. Z. Kolter, "DC3: a learning method for optimization with hard constraints", in Proc. ICLR, 2021.

Incorporate Line Limit Violations into Loss

- Ensure box constraints for $\boldsymbol{p}_G, \boldsymbol{q}_G$, e.g., $p_{Gi} = \alpha_i\left(P_{Gi}^{max} - P_{Gi}^{min}\right) + P_{Gi}^{min}, \alpha \in [0,1]$



$x \in R^{\mathrm{d}}$: load input, $y \in R^m$: independent variables, $z \in R^n$: dependent variables, l: penalty function

- Loss function:
  - $w_1 \cdot loss_{pred} + w_2 \cdot loss_{pen}$
- Computing penalty gradient by the chain rule:
  - The mapping between y and z does not admit an explicit form

$$\nabla l(y) = \begin{bmatrix} \frac{\partial l(y,z)}{\partial y_1} \\ \vdots \\ \frac{\partial l(y,z)}{\partial y_m} \end{bmatrix}^T + \begin{bmatrix} \frac{\partial l(y,z)}{\partial z_1} \\ \vdots \\ \frac{\partial l(y,z)}{\partial z_n} \end{bmatrix}^T \cdot \begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \cdots & \frac{\partial z_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial y_1} & \cdots & \frac{\partial z_n}{\partial y_m} \end{bmatrix}$$

[1] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," arXiv preprint arXiv:2007.01002, 2020; also in IEEE Systems Journal 2023

[2] P. L. Donti, D. Rolnick and J. Z. Kolter, "DC3: a learning method for optimization with hard constraints", in Proc. ICLR, 2021.

Computing Penalty Gradient Directly

- The AC power flow equations implicitly encode the y–z mapping
  - Penalty gradient can be computed by exploring implicit function theorem

Denote AC–PF equations by:

$$h_i(y, z) = 0, i = 1, \dots n$$

$$\begin{bmatrix} \frac{\partial h_1}{\partial y_1} & \cdots & \frac{\partial h_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial y_1} & \cdots & \frac{\partial h_n}{\partial y_m} \end{bmatrix} + \begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \cdots & \frac{\partial h_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial z_1} & \cdots & \frac{\partial h_n}{\partial z_n} \end{bmatrix} \begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \cdots & \frac{\partial z_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial y_1} & \cdots & \frac{\partial z_n}{\partial y_m} \end{bmatrix} = 0$$

$$\begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \cdots & \frac{\partial z_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_n}{\partial y_1} & \cdots & \frac{\partial z_n}{\partial y_m} \end{bmatrix} = - \left(\begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \cdots & \frac{\partial h_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial z_1} & \cdots & \frac{\partial h_n}{\partial z_n} \end{bmatrix} \right)^{-1} \begin{bmatrix} \frac{\partial h_1}{\partial y_1} & \cdots & \frac{\partial h_1}{\partial y_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial y_1} & \cdots & \frac{\partial h_n}{\partial y_m} \end{bmatrix}$$

[1] P. L. Donti, D. Rolnick and J. Z. Kolter, “DC3: a learning method for optimization with hard constraints”, in Proc. ICLR, 2021.

Estimating Penalty Gradient

- The penalty function is a composite function of y:
- Two-point gradient estimation
  - Estimate gradient by perturbating y and computing the penalty twice
  - Better empirical performance than the implicit function theorem-based method

Inputs	Penalty
$y + \mu\delta$	$l(y + \mu\delta)$
$y - \mu\delta$	$l(y - \mu\delta)$

δ: smooth parameter, m: the input dimensions, $\mu \in R^m$: a uniformly-sampled vector from the unit ball

$$\nabla l(y) \approx \frac{l(y + \mu\delta) - l(y - \mu\delta)}{2\delta} m \cdot \mu$$

[1] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," arXiv preprint arXiv:2007.01002, 2020, also in IEEE Systems Journal 2023.

Simulation Settings

- Test cases: IEEE 30-/118-/300-bus and a synthetic 2000-bus mesh power network [1]

#Bus	#P-V Bus	#P-Q bus	#Branch	#Hidden layers	#Neurons per layer
30	5	24	41	2	64/32
118	53	63	231	2	256/128
300	68	231	411	2	512/256
2000	177	1822	3693	2	2048/1024

- Workstation: CentOS 7.6 with quad-core (i7-3770@3.40G Hz) CPU and 16GB RAM
- Datasets: (i) synthetic dataset with $\pm$10% variation; (ii) California demand curve with up to 40% variation; 10,000 training samples and 2,500 for testing
- Schemes: DeepOPF(-AC), Pypower, DNN-warm start [2], DNN-E [3]

[1] Powergrid Lib 2000-bus synthetic test case," 2022, https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg2000/
[2] W. Dong, Z. Xie, G. Kestor, and D. Li, "Smart-PGSim: Using Neural Network to Accelerate AC-OPF Power Grid Simulation," in Proc. SC20, St. Louis, MO, USA, 2020
[3] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow, IEEE SmartGridComm, 2020.

Simulations for Realistic Load w. 40% Variation

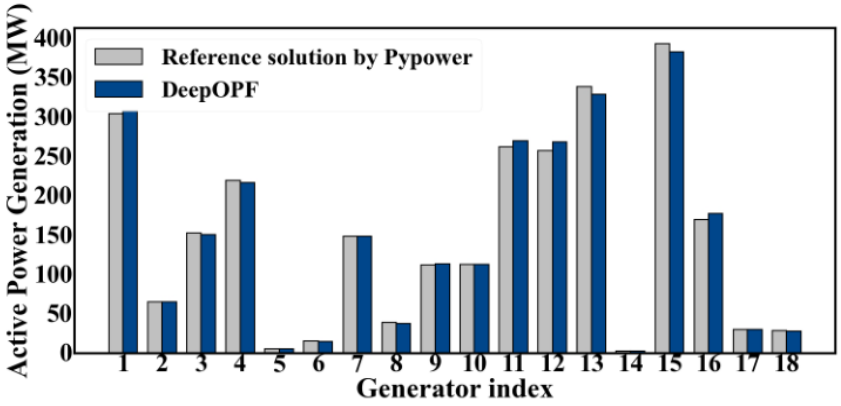
Test case	Feasibility rate (%) before feasibility-recovery*			Average cost difference (%)			Average speedup		
	DNN-E	DNN-W	DeepOPF	DNN-E	DNN-W	DeepOPF	DNN-E	DNN-W	DeepOPF
IEEE Case30	36	100	100	< 0.1	0	< 0.1	$\times 7.3$	$\times 1.0$	$\times 13$
IEEE Case118	80	100	99	< 0.1	0	< 0.1	$\times 11$	$\times 1.1$	$\times 12$
IEEE Case300	49	100	100	< 0.2	0	< 0.2	$\times 16$	$\times 1.7$	$\times 33$
IEEE Case2000	60	100	100	< 0.2	0	< 0.2	$\times 44$	$\times 0.9$	$\times 70$

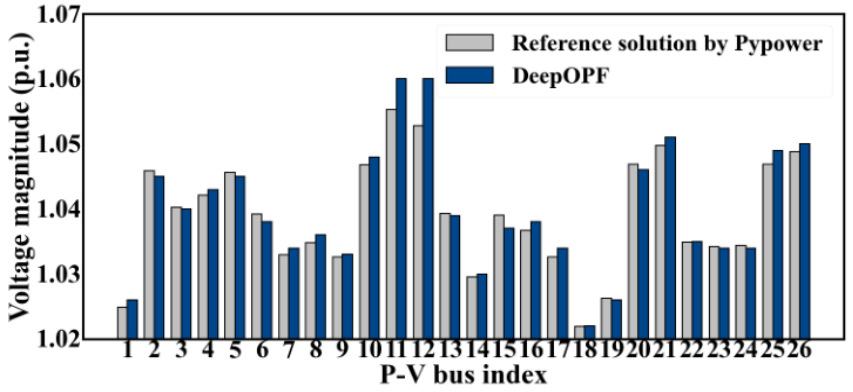
- Speedups are higher for DNN-E and DeepOPF than DNN-W
- **Remark**: the speedup is roughly the same if use the truncated iterative solver's running time upon achieving the same optimality gap as DNN approaches

Simulations for Synthetic Load: ±10% Variation

- Speedup higher than realistic loads with 40% variation

Test case	Feasibility rate (%) before feasibility-recovery*			Average cost difference (%)			Average speedup		
	DNN-E	DNN-W	DeepOPF	DNN-E	DNN-W	DeepOPF	DNN-E	DNN-W	DeepOPF
IEEE Case30	42	100	100	<0.1	0	<0.1	×1[illegible]	×1.1	×2[illegible]
IEEE Case118	22	100	100	<0.1	0	<0.1	×4.2	×1.3	×22
IEEE Case300	21	100	99	<0.1	0	<0.1	×5.4	×1.8	×20
IEEE Case2000	29	100	99	<0.1	0	<0.1	×24	×1.0	×123





Comparison of DeepOPF and Pypower solutions for IEEE Case118 test case

A Brief Summary

- DeepOPF speedups AC-OPF solving time by ~100x with <0.2% optimality loss, over a 2000-bus system
  - PR2 with a penalty approach can guarantee equality constraints and promote inequality constraint feasibility
- We Can further improve the speed-up by employing alternative PR2 design [1]
  - 1,5000x speedup for AC-OPF over a 2000-bus network, with sub-percentage optimality loss
  - At the expense of minor load and generation adjustment

[1] W. Huang, X. Pan, M. Chen, and S. H. Low, "DeepOPF-V: Solving AC-OPF Problems Efficiently", IEEE Transactions on Power Systems, Jan. 2022

Glance of the Challenges and Approaches

- NN solutions may be infeasible
  - Predict-and-reconstruct (PR2) / equation completion
  - Primal-dual training (utilizing KKT conditions)
  - Differentiable NN layers
  - More advanced techniques to be covered in **Part III**
- High training/data complexity for large-scale OPF problems
  - Un-/self- supervised learning, GNN, grid decomposition, compact learning, approximate data labels
- Non-universal: need one DNN per system
  - **Will be discussed in Part III**

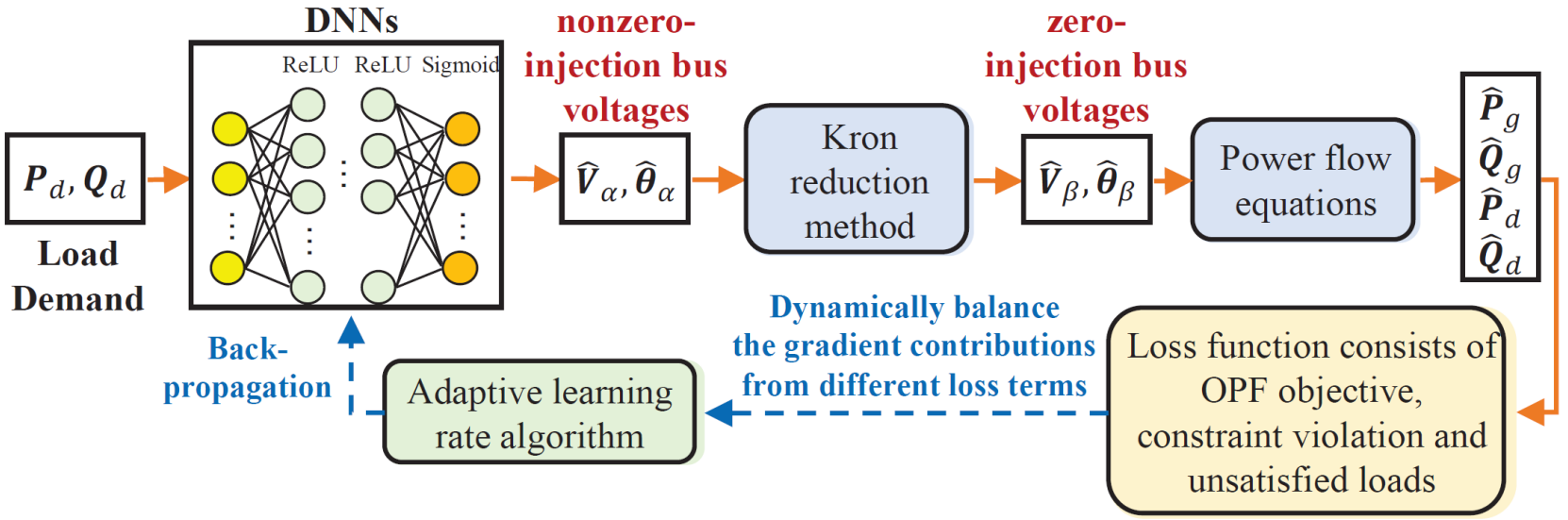
We will discuss the topics in color blue in this part

Unsupervised Learning for AC-OPF

- Solving 10,000 AC-OPF instances on a 2742-bus system takes 3+ days [3]
  - Workstation, dual Intel 2.10GHz CPUs and 128GB RAM
- Approach: unsupervised training [1, 2]
  - No training data ground truth (OPF solutions) needed
  - Use the OPF objective and constraint violation to guide the DNN training
- Note: also other methods and GNN-based approaches (check the SIGEnergy wiki page for a more complete list)

[1] W. Huang and M. Chen, "DeepOPF-NGT: A Fast Unsupervised Learning Approach for Solving AC-OPF Problems without Ground Truth", In Proceedings of the 38th International Conference on Machine Learning Workshop, virtual conference, Jul. 23, 2021.
[2] P. L. Donti, D. Rolnick and J. Z. Kolter, "DC3: a learning method for optimization with hard constraints", ICLR, 2021.
[3] S. Babaeinejadsarookolaee, et al. (23 authros), "The power grid library for benchmarking ac optimal power flow algorithms", arXiv preprint arXiv:1908.02788, 2019.

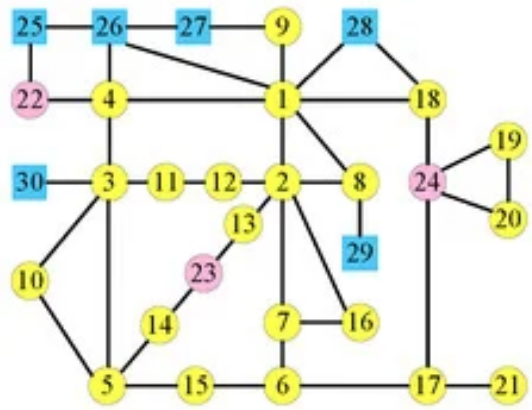
DeepOPF-NGT: DeepOPF w/o Ground Truth



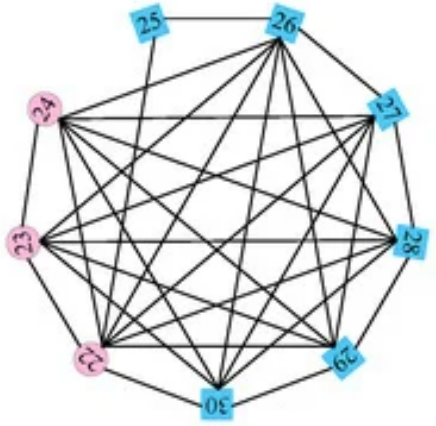
□ Contributions

- Employ Kron Reduction to simplify the ACOPF problem (~1/3 smaller)
- Train DNN to solve AC-OPF without ground truth
- Learn legitimate input-solution mapping under a learnability condition (avoid the multi-valued mapping issue in supervised learning)

Kron Reduction to Simplify ACOPF



(a) Topology of the IEEE 30-bus system



(b) Topology of the reduced IEEE 30-bus system

- **Kron Reduction** [1]: Voltages at (zero-injection) internal buses can be computed from voltages at PV and PQ buses by solving linear equations
- NN only needs to predict voltages at PV and PQ buses; AC power flow equations at internal buses are guaranteed to be satisfied

[1] G. Kron, Tensor Analysis of Networks. New York, NY, USA: Wiley,1939.

[2] H. Song, L. Han, Y. Wang, W. Wen, and Y. Qu, "Kron Reduction Based on Node Ordering Optimization for Distribution Network Dispatching with Flexible Loads", Energies, 2022

Use Objective and Constraint Violation to Guide DNN Training

$$L = k_o L_o(x, \varphi) + k_c L_c(x, \varphi) + k_d L_d(x, \varphi)$$

OPF objective

constraint violation

load mismatch

- We use the following adaptive learning rate in DeepOPF-NGT

$$k_d^t = \min\{\frac{k_o \mathcal{L}_o(x, \varphi)}{\mathcal{L}_d(x, \varphi)}, \overline{k}_d\}$$

$$k_c^t = \min\{\frac{k_o \mathcal{L}_o(x, \varphi)}{\mathcal{L}_c(x, \varphi)}, \overline{k}_c\}$$

$\overline{k}_d$ and $\overline{k}_c$: upper bounds for penalty coefficients k_d and k_c.

- Benefit: balance impact of different terms in the loss function

[1] W. Huang and M. Chen, "DeepOPF-NGT: A Fast Unsupervised Learning Approach for Solving AC-OPF Problems without Ground Truth", In Proceedings of the 38th International Conference on Machine Learning Workshop, virtual conference, Jul. 23, 2021.

DeepOPF-NGT Learns Legitimate Mapping

- **Theorem**: under a general learnability condition, DeepOPF-NGT learns a legitimate input-solution mapping for ACOPF upon good-enough training

- Avoid the multi-valued mapping issue in supervised learning

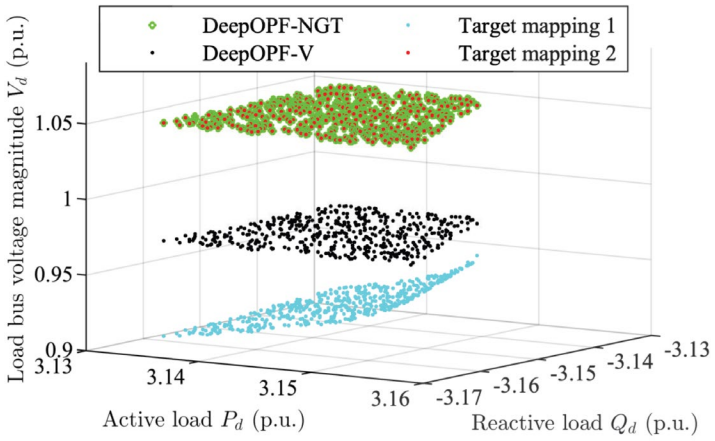
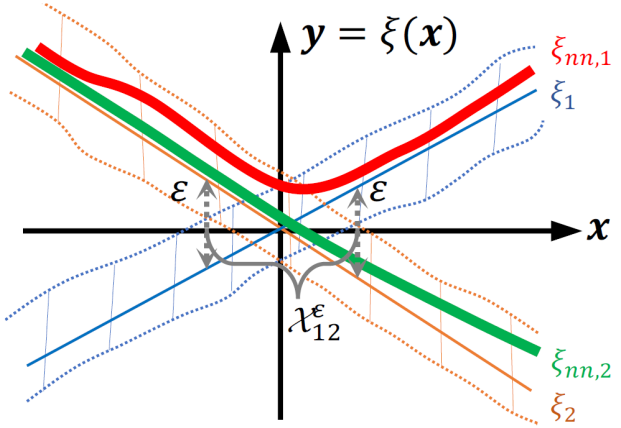


Fig. 2. Load-solution mappings in an illustrating two-bus system.



Unsupervised Learning Works

- IEEE Case118 test case; training/testing samples: 600/40,000
- Training time: 3 hrs for DeepOPF-V, 1 hr for DeepOPF-NGT, 10 min for EACOPF

Metric	IEEE 118-bus system			
	DeepOPF-NGT	DeepOPF-V	DeepOPF-AC	EACOPF
$\eta_{opt}(\%)$	<1.0	<0.1	<3.6	<10.7
$\eta_{\boldsymbol{V}}(\%)$	-	-	**99.2**	**94.5**
$\eta_{\boldsymbol{P_g}}(\%)$	100.0	100.0	100.0	99.3
$\eta_{\boldsymbol{Q_g}}(\%)$	100.0	99.9	99.9	100.0
$\eta_{\boldsymbol{S_l}}(\%)$	99.6	100.0	100.0	99.5
$\eta_{\boldsymbol{\theta_l}}(\%)$	100.0	100.0	100.0	100.0
$\eta_{\boldsymbol{P_d}}(\%)$	99.8	99.9	-	-
$\eta_{\boldsymbol{Q_d}}(\%)$	99.9	100.0	-	-
$t_{dnn}(s)$	1.9e-4	7.2e-4	1.3e-2	3.2e-2
η_{sp}	×3589	×1995	×50	×21

Ground Truth Data Help

- A small amount of ground truth data can improve performance

N_{label}	0	50	150	250
$\eta_{opt}(\%)$	< 0.3	< 0.2	< 0.1	< 0.1
$\eta_{\boldsymbol{V}}(\%)$	-	-	-	-
$\eta_{\boldsymbol{P_g}}(\%)$	**99.1**	99.4	99.7	**99.8**
$\eta_{\boldsymbol{Q_g}}(\%)$	99.9	100.0	100.0	100.0
$\eta_{\boldsymbol{S_l}}(\%)$	100.0	100.0	100.0	100.0
$\eta_{\boldsymbol{\theta_l}}(\%)$	100.0	100.0	100.0	100.0
$\eta_{\boldsymbol{P_d}}(\%)$	**99.4**	99.5	99.6	**99.8**
$\eta_{\boldsymbol{Q_d}}(\%)$	**98.8**	99.2	99.4	**99.7**

Glance of the Challenges and Approaches

- NN solutions may be infeasible
  - Predict-and-reconstruct (PR2) / equation completion
  - Primal-dual training (utilizing KKT conditions)
  - Differentiable NN layers
  - More advanced techniques to be covered in **Part III**

- High training/data complexity for large-scale OPF problems
  - Un-/self- supervised learning, GNN, grid decomposition, compact learning, approximate data labels

- Non-universal: need one DNN per syst
  topology, admittance, gen
  - **Will be discussed in Part I**

We will discuss the topics in color blue in this part

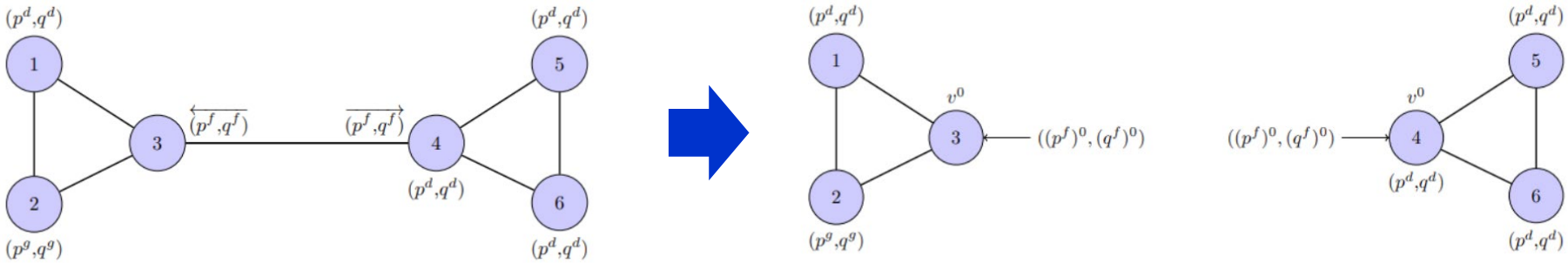
High Training Complexity for Large AC-OPF

- Training DNN to solve large-scale AC-OPF problems incurs high complexity [1]
  - Large DNN output dimension: 28,180 for a 9241-bus system
  - Long training time: 7 hours for AC-OPF problems over a 3500-bus system
- Complexity may increase exponentially in the grid size
- Next: decomposition approach; GNN approach to be discussed in part III

[1] M. Chatzos, T. W. K. Mak and P. Vanhentenryck, "Spatial Network Decomposition for Fast and Scalable AC-OPF Learning," in *IEEE Trans. on Power Systems, 2022*,

Grid Decomposition

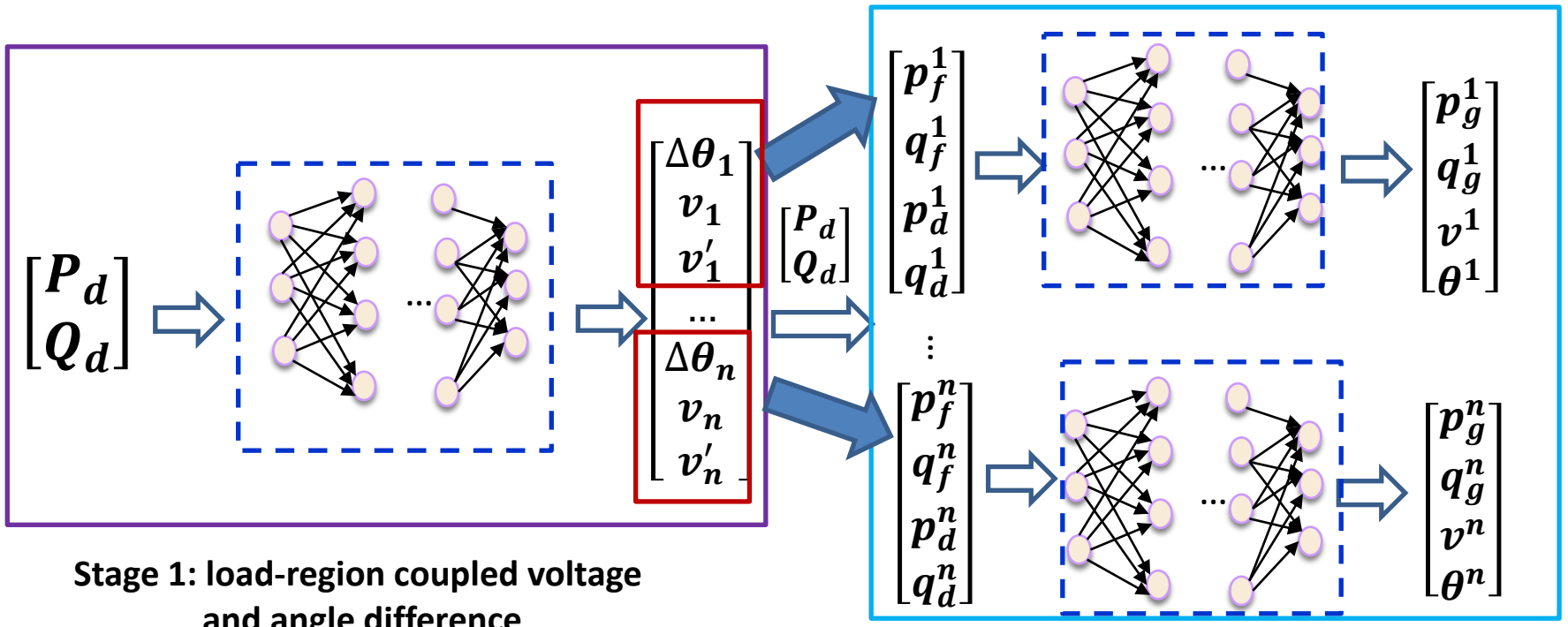
- Decompose a power grid into disjoint regions [1]
  - Regions are connected via coupling branches
  - The coupling branch flows are sufficient statistics to separate regions
- Keep the training complexity linear in the grid size



[1] M. Chatzos, T. W. K. Mak and P. Vanhentenryck, "Spatial Network Decomposition for Fast and Scalable AC-OPF Learning," in *IEEE Trans. on Power Systems, 2022*

Two-stage Learning for AC-OPF

- Use a two-stage approach to solve large-scale AC-OPF problems
  - Stage 1: predict load to the coupled voltage and angle
  - Stage 2: predict (load, coupled flow) to OPF solutions in each region



Stage 1: load-region coupled voltage and angle difference

Stage 2: (load, region coupled flow)-OPF solution

Evaluation over an RTE 9241-bus Network

Speedup	Optimality gap	Feasibility rate
x10	0.03%	$> 98.5\%$

- 9,241 buses, 16,049 branches, 4,895 load and 1,445 generator buses
- Load profile: $\pm 7.75\%$ variation
- Training/test dataset: 8K/2K samples [1]
- Training time: 30/60 minutes for $1^{st}/2^{nd}$ stage
- Baseline: IPOPT solver

[1] C.Josz, S. Fliscounakis, J. Maeght, P. Panciatici, "AC power flow data in MATPOWER and QCQP format: iTesla, RTE snapshots, and PEGASE", *arXiv preprint arXiv:1603.01533*.

Glance of the Challenges and Approaches

- NN solutions may be infeasible
  - Predict-and-reconstruct (PR2) / equation completion
  - Primal-dual training (utilizing KKT conditions)
  - Differentiable NN layers
  - More advanced techniques to be covered in **Part III**
- High training/data complexity for large-scale OPF problems
  - Un-/self- supervised learning, GNN, grid decomposition, compact learning, approximate data labels
- The multi-valued mapping issue for non-convex (AC-OPF) problems
  - Augmented learning, data preparation/selection
  - Generative learning (learning the input-dependent solution distributions)
- Non-universal: need one DNN per system
  admittance, generator on/off, et
  - **Will be discussed in Part III of th**

We will discuss the topics in color blue in this part

Outline

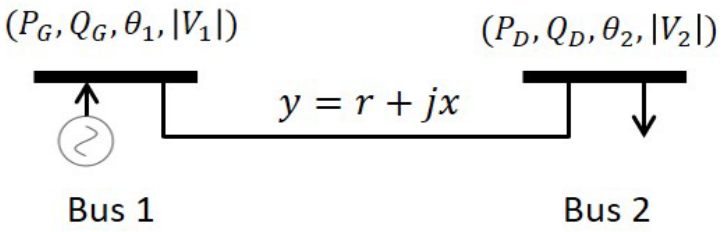
- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

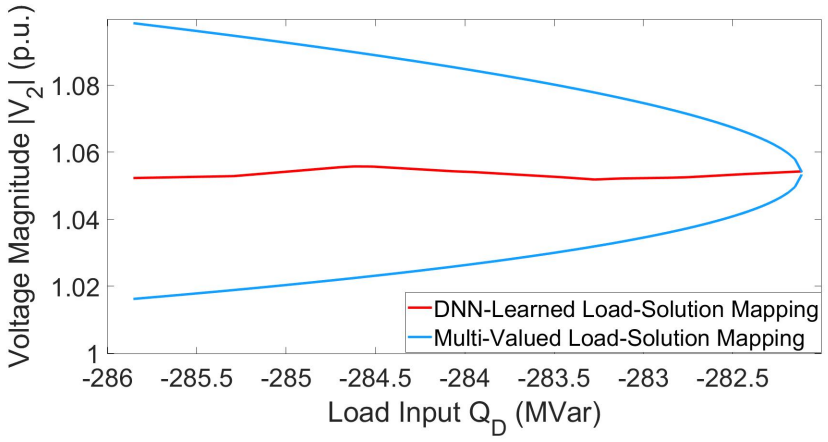
- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems

- Concluding remarks and open challenges

AC-OPF Problem Admits Multi-Valued Mapping



A toy 2-bus example.



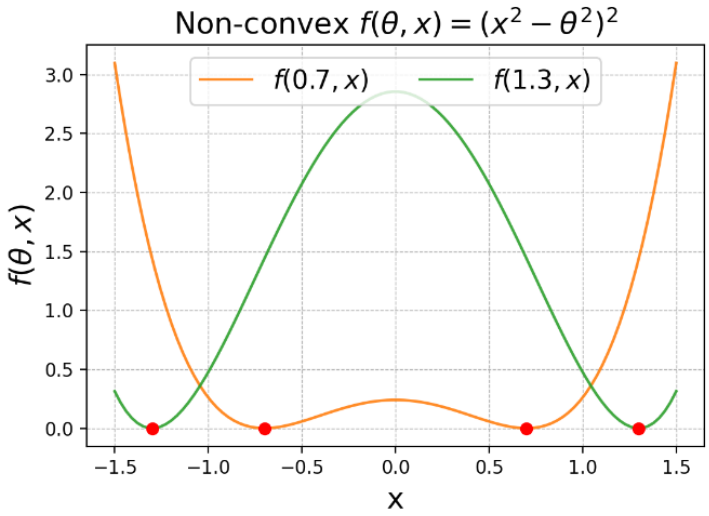
DNN's mapping vs. target mapping.

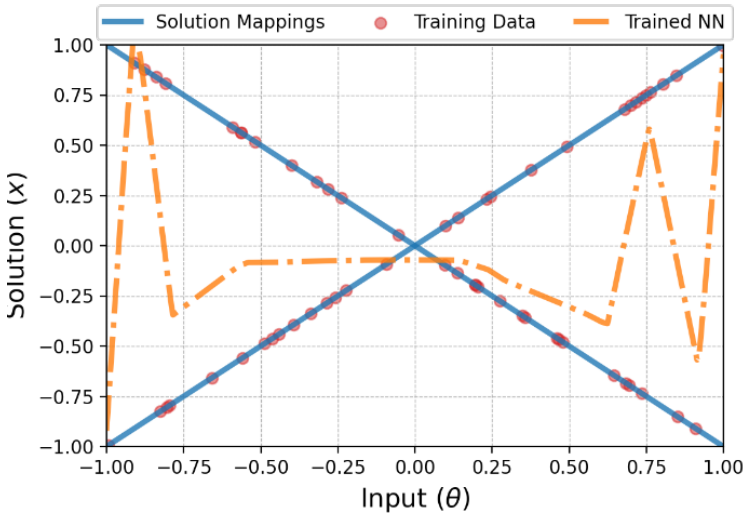
- AC-OPF problem is non-convex and can admit multiple optimal or near-optimal solutions [1-3]
- A well-trained DNN with (standard) supervised learning fails to learn a target mapping [3]

[1] W. A. Bukhsh, A. Grothey, K. I. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," IEEE Trans. Power Syst., vol. 28, no. 4, 2013.
[2] J. Kotary, F. Fioretto, and P. Van Hentenryck, "Learning hard optimization problems: A data generation perspective," NeurIPS 2021.
[3] W. Huang, M. Chen, and S. H. Low, "Unsupervised Learning for Solving AC Optimal Power Flows: Design, Analysis, and Experiment", IEEE Transactions on Power Systems, vol. 39, issue 6, pp. 7102 - 7114, November 2024.

The Observation is General

- Non-convex problems may admit multi-valued mapping





- NN struggles to learn the multi-valued mapping

ML methods for Multi-valued Mapping

Existing Work	Learning target	Optimality guarantee	Low run-time complexity	Training techniques
Data generation	Mapping	✓	✓	Data pre.
Data selection	Mapping	✗	✓	Data pre.
Data augmentation	Mapping	✓	✓	Data pre.
Unsupervised learning	Mapping	✗	✓	Loss design
Hindsight loss	Mappings	✗	✓	Loss design
Clustering approach	Mappings	✗	✓	Clustering
Generative Learning	Distribution	✓	✓	Generative

Data generation [1]: generate training data from a single mapping

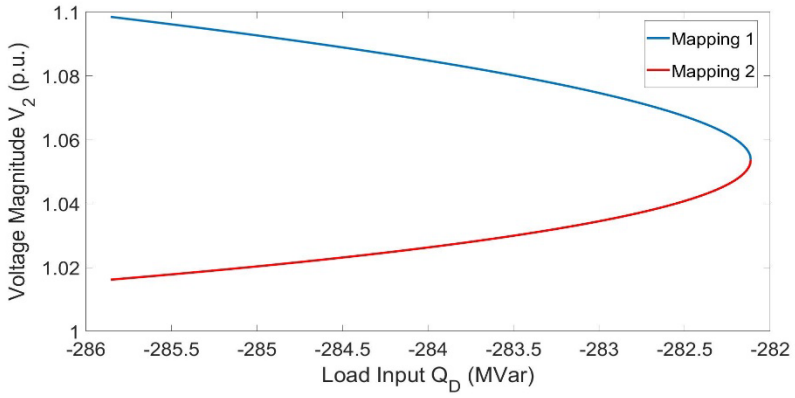
Data selection [2]: use another NN to pick data from training dataset

Unsupervised learning [3]: not rely on input-solution data, may avoid this issue

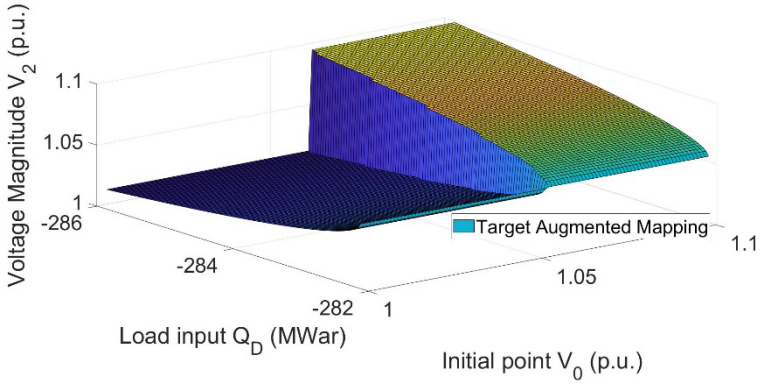
- They lack general guarantee of learning one legitimate mapping

[1] Kotary, J., Fioretto, F., & Van Hentenryck, P. Learning hard optimization problems: A data generation perspective. NeurIPS 2021.
[2] Nandwani, Y., Jindal, D., & Singla, P. Neural learning of one-of-many solutions for combinatorial problems in structured output spaces. ICLR 2021
[3] Huang, W., Chen, M., & Low, S. H. Unsupervised Learning for Solving AC Optimal Power Flows: Design, Analysis, and Experiment. IEEE Trans. Power Syst. 2024

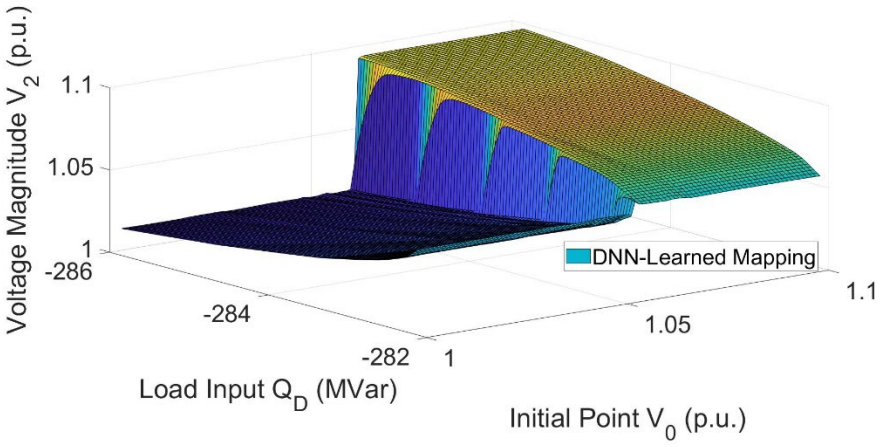
Approach #1: Augmented Learning



Embedding

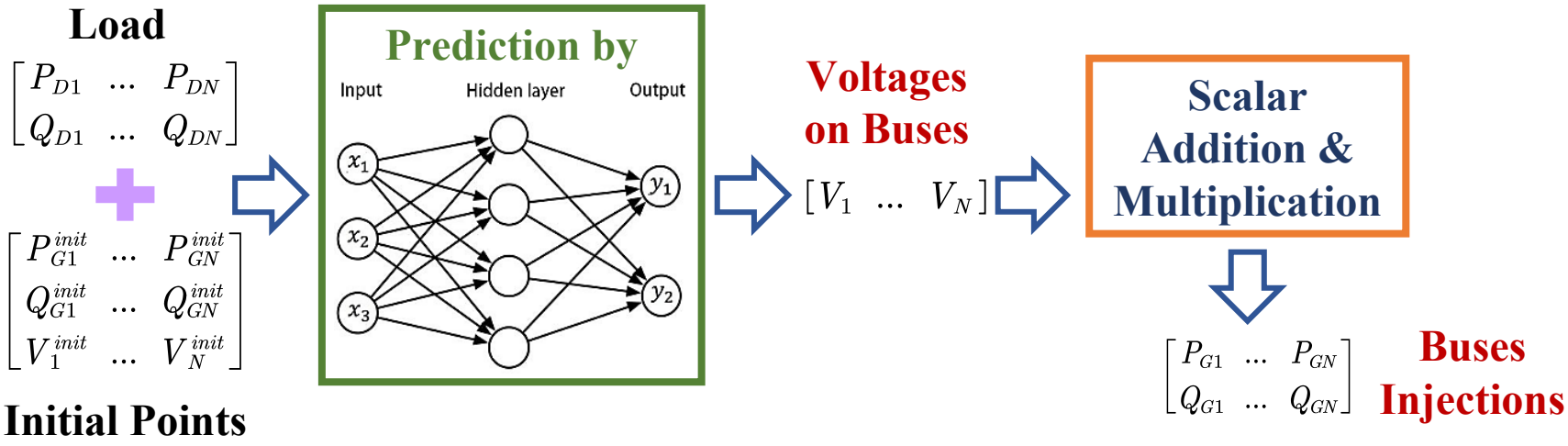


- Augment the load with the initial point in training data generation
- The augmented mapping is unique and can be learned by DNN



DeepOPF-AL[1]: A Simple Design

- Follow prediction-and-reconstruction in DeepOPF-V [2]
  - Learn the unique augmented mapping from (load, initial point) to optimal solution



[1] X. Pan, W. Huang, M. Chen, and S. H. Low, "DeepOPF-AL: Augmented Learning for Solving AC-OPF Problems with a Multi-Valued Load-Solution Mapping", ACM e-Energy, 2023

[2] W. Huang, X. Pan, M. Chen, and S. H. Low, "Deepopf-V: Solving AC-OPF problems efficiently," IEEE Trans. Power Syst., 2021

Simulation for Learning 2-valued Mapping

- Compare DeepOPF-AL with DeepOPF-V over IEEE Case39 with realistic load profile (40% variation)
  - Each load corresponds to two solutions

Metric	Case39-V2 with Avg. Cost Diff. = 30%			
	Balanced Dataset		Unbalanced Dataset	
	DeepOPF-AL	DeepOPF-V	DeepOPF-AL	DeepOPF-V
$\eta_{opt}(\%)$	**0.48**	-8.56	**0.66**	-5.98
$\eta_{P_G}(\%)/\eta_{Q_G}(\%)$	97.5/**94.6**	99.3/91.7	97.4/**98.4**	99.8/96.7
$\eta_{S_l}(\%)$	100	100	100	100
$\eta_{P_D}(\%)/\eta_{Q_D}(\%)$	**0.19**/**6.47**	0.61/27.6	**0.09**/**0.49**	0.32/12.9
t_{mips} (ms)	2808	2808	2676	2676
t_{dnn} (ms)	1.4	1.3	1.3	1.2
η_{speed}	×2006	×2160	×2058	×2230

Approach #2: Learning Input-dependent Solution Distribution Instead!

- Learning solution mapping by NN:

CO

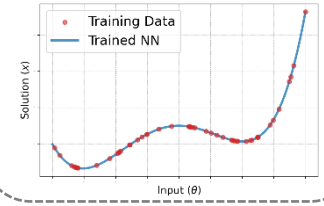
$$\min_{x \in \mathbb{R}^n} f(x, \theta)$$
$$\text{s.t. } x \in \mathcal{X}_\theta$$

Dataset

$\theta_1, \theta_2, \ldots$ → Iterative solver → $x_1^*, x_2^*, \ldots$

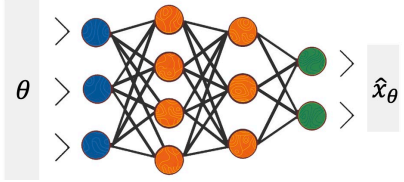
Training

$$\min_{F_{nn}} \sum_i ||F_{nn}(\theta_i) - x_i^*||$$



Deployment

$$\hat{x}_\theta = F_{nn}(\theta)$$



- New approach : Generative Learning

CO

$$\min_{x \in \mathbb{R}^n} f(x, \theta)$$
$$\text{s.t. } x \in \mathcal{X}_\theta$$

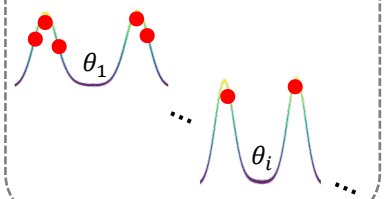
Dataset

$\boldsymbol{x_\theta} \sim p(x|\theta)$

$\theta_1, \theta_2, \ldots$ —sampling→ $\tilde{x}_{1,1}, \tilde{x}_{1,2}, \ldots$; $\tilde{x}_{2,1}, \tilde{x}_{2,2}, \ldots$; …

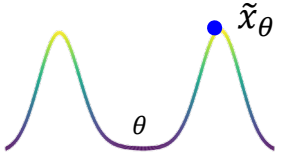
Training

$\boldsymbol{x_\theta} = F(\theta, \boldsymbol{z})$



Deployment

$\tilde{x}_\theta = F(\theta, \hat{z})$

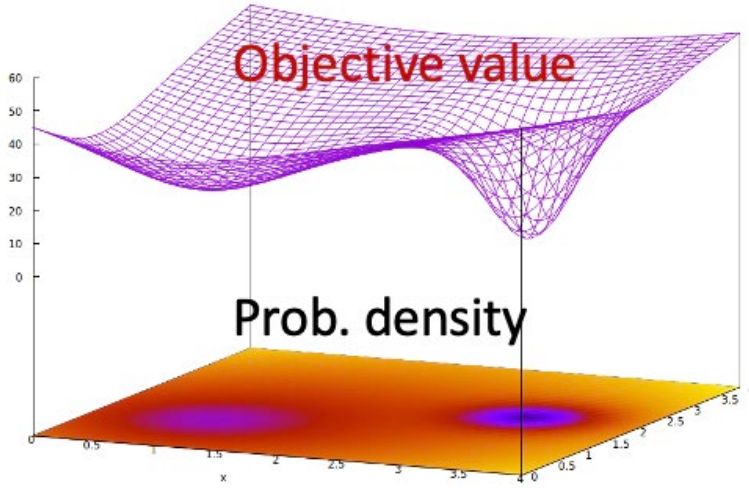


Prepare Training Data from Solution Distribution

- Requirements of solution distribution $p(x|\theta)$
  - Continuous probability density
  - Concentrate around high-quality solutions
  - Can apply existing solvers to sample from it

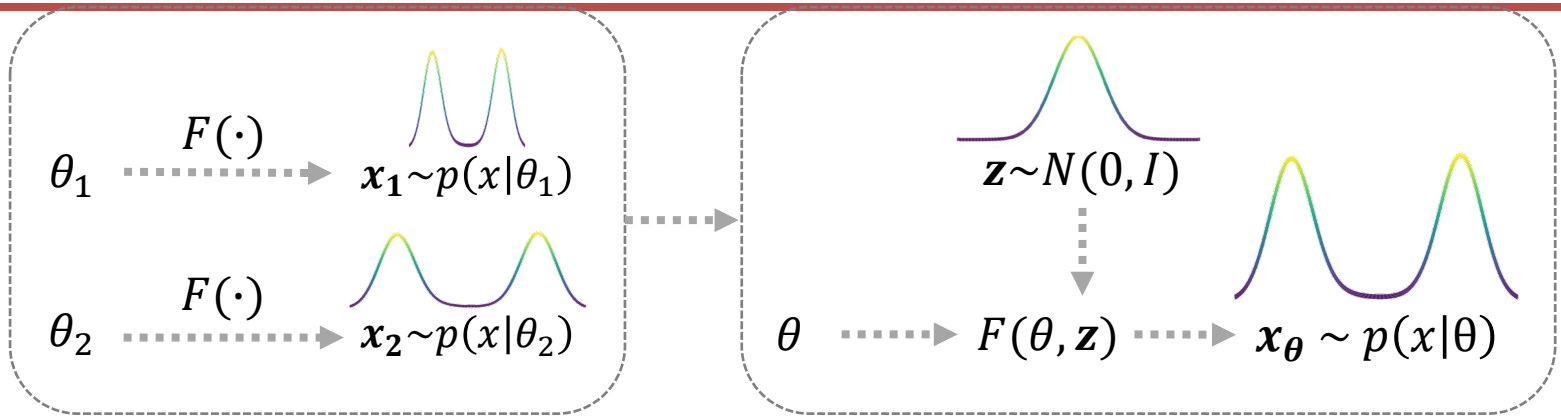
- E.g., Boltzmann distribution:
  - $p(x|\theta) \propto \exp\big(-f(x,\theta)\big)$
  - Markov Chain Monte-Carlo [1]

- Other distributions [2-3]



[1] Salakhutdinov, R. Learning deep Boltzmann machines using adaptive MCMC. ICML 2010
[2] Xu, P., Chen, J., Zou, D., & Gu, Q. Global convergence of Langevin dynamics based algorithms for nonconvex optimization. NeurIPS 2018
[3] Villani, C. Optimal transport: old and new (Vol. 338, p. 23). Berlin: springer. 2009.

Learn an Input Solution-Distribution Mapping



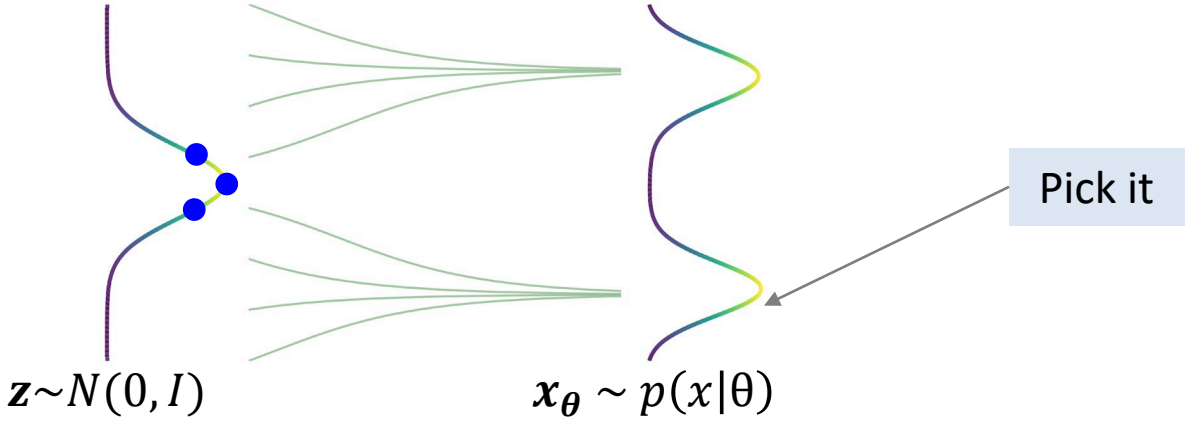
- Generative approach: $\boldsymbol{x_\theta} = F(\theta, \boldsymbol{z})$
  - Map (input θ, a Gaussian sample $\boldsymbol{z}$) to a sample from $p(x|\theta)$
  - E.g., VAE, GAN, Diffusion, RectFlow [1], …

$$F(\theta,\boldsymbol{z}) = \boldsymbol{z} + \int_0^1 u(x_t,t,\theta)dt \text{ with } x_0 = \boldsymbol{z} \text{ and } dx_t/dt = u(x_t,t,\theta)$$

- For continuous $p(x|\theta)$, there exists a continuous explicit-form u and can be approximated by NN
- Training NN v_{nn} (e.g., MLP) to learn u with samples from Gaussian and solution distributions

[1] Liu, X., Gong, C., & Liu, Q. (2022). Flow straight and fast: Learning to generate and transfer data with rectified flow. ICLR 2023

Generate Solutions for New Inputs



- Given new θ, use trained v_{nn} to sample solutions
  - **Step 1**: sample $\hat{z}$ from Gaussian
  - **Step 2**: pass θ and $\hat{z}$ to NN mapping
    - Initialize $x_0 = \hat{z}$
    - k-step iteration: $x_{t+1/k} = x_t + 1/k \cdot v_{nn}(x_t, t, \theta)$
    - Output x_1 as a solution
  - **Step 3**: select the best one from multiple solutions

Benign Optimality Gap and Run-time Complexity

Theorem 2. Given $\boldsymbol{m}$-layer NN vector v_{nn} and generate $\boldsymbol{M}$ solutions in $\boldsymbol{n}$-dim with $\boldsymbol{k}$-step integration as $\left\{\hat{x}_{\theta,i}^{k}\right\}_{i=1}^{M}$,

- Probability of the best solution with optimality gap larger than δ decrease exponentially with M as:

Best sampled solution

δ-optimum

$$\Pr\left(\min_{i=1,\ldots,M}\left\{f\left(\hat{x}_{\theta,i}^{k},\theta\right)\right\}\geq f_{\theta}^{*}+\delta\right)\leq\left(1-\Pr\left(S_{\theta}^{\delta}\mid\theta\right)+C_{1}\epsilon_{nn}^{1/4}+C_{2}k^{-1/2}\right)^{M}$$

Concentration of $p(x|\theta)$ on optimum

Average NN Errors

Steps of integration

- The run-time complexity is: $M\cdot O(kmn^{2})$

Outline

- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

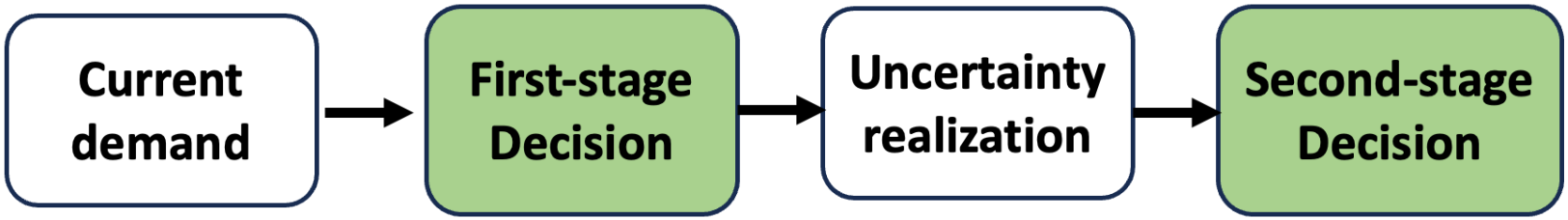
- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems

- Concluding remarks and open challenges

Machine Learning for Solving 2-Stage Stochastic AC-OPF Problems

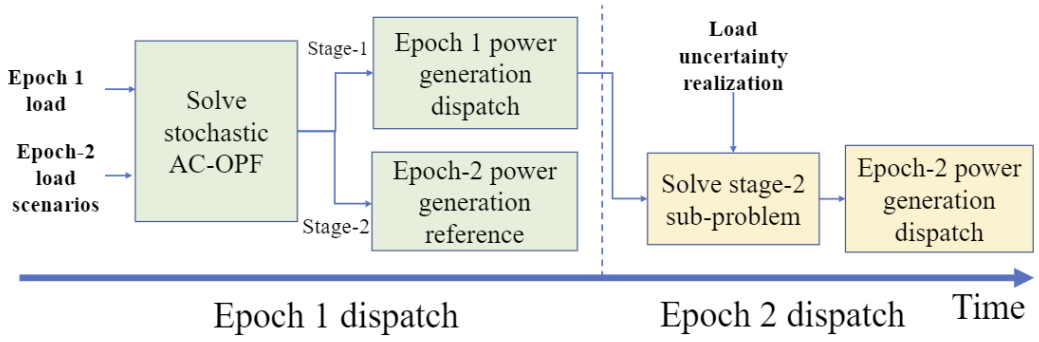
- Min Zhou, Enming Liang, Minghua Chen, and Steven H. Low, “Partially Permutation-Invariant Neural Network for Solving Two-Stage Stochastic AC-OPF Problem,” IEEE Transactions on Power Systems, vol. 41, no. 2, pp. 1246–1263, 2026.
- Ali Rajaei, Olayiwola Arowolo, and Jochen L. Cremer, “Learning-Accelerated ADMM for Stochastic Power System Scheduling With Numerous Scenarios,” IEEE Transactions on Sustainable Energy, 2025.
- Shishir Lamichhane, Abodh Poudyal, Nicholas R. Jones, Bala Krishnamoorthy, and Anamika Dubey, “Scalable Two-Stage Stochastic Optimal Power Flow via Separable Approximation,” arXiv:2504.13933, 2025.
- Ling Zhang, Daniel Tabas, and Baosen Zhang, “An Efficient Learning-Based Solver for Two-Stage DC Optimal Power Flow with Feasibility Guarantees,” arXiv:2304.01409, 2024.
- Sarthak Gupta, Sidhant Misra, Deepjyoti Deka, and Vassilis Kekatos, “DNN-Based Policies for Stochastic AC OPF,” Electric Power Systems Research, vol. 213, article 108563, 2022.
- Zhentong Shao, Jingtao Qin, and Nanpeng Yu, “Neural Two-Stage Stochastic Volt-VAR Optimization for Three-Phase Unbalanced Distribution Systems with Network Reconfiguration,” arXiv:2510.23867, 2025.
- Justin Dumouchelle, Rahul Patel, Elias B. Khalil, and Merve Bodur, “Neur2SP: Neural Two-Stage Stochastic Programming,” NeurIPS, 2022.

Stochastic AC-OPF for Managing Uncertainty



- First stage
  - Set generator output based on the current demand
  - The model actively anticipates future uncertainty when making this decision
- Second stage
  - Uncertainty realized
  - Re-dispatch the decisions to maintain grid stability

2-stage AC-OPF Problem



$$\min \quad f(\boldsymbol{u}_0) + \frac{1}{K}\sum_{k\in\mathcal{S}} f(\boldsymbol{u}_k)$$

$$\text{s.t.} \quad g(\boldsymbol{u}_k, \boldsymbol{\zeta}_k, \boldsymbol{d}_k) = 0, \quad \forall k \in \{0\} \cup \mathcal{S},$$

$$h(\boldsymbol{u}_k, \boldsymbol{\zeta}_k, \boldsymbol{d}_k) \le 0, \quad \forall k \in \{0\} \cup \mathcal{S},$$

$$|\boldsymbol{u}_k - \boldsymbol{u}_0| \le \boldsymbol{\Delta}, \quad \forall k \in \mathcal{S},$$

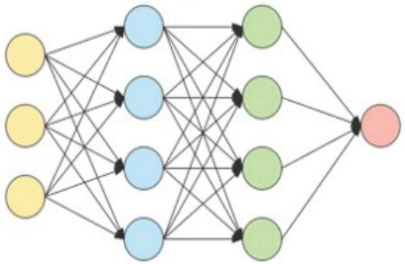
$$\text{var.} \ \boldsymbol{\zeta}_k, \boldsymbol{u}_k, \quad \forall k \in \{0\} \cup \mathcal{S}.$$

- 2-stage AC-OPF is important for grid operation under uncertainty, but it is challenging
  - $d_1, d_2, \ldots, d_K$: load/renewable inputs for K scenarios
  - Decisions are coupled
  - Non-convex and NP-hard
  - Huge problem size, e.g., for a 300-bus test system with 1,000 scenarios, #variables > 1 million

Research Gap

- Iterative algorithms are not scalable
  - e.g. it takes more than 30 mins to solve the 118-bus 2-stage AC-OPF problem with 100 scenarios using Newton's method on a E5-2630@2.40GHz CPU

- Vanilla NN design suffers from dimension explosion
  - NN input dimension increases linearly in #scenarios K
  - #Training data needed grows exponentially in K

$(d_0, d_1, d_2, \dots d_K)$ →  → $(u_0, u_1, u_2, \dots, u_k)$

Partially Permutation-Invariance Property

$$\begin{aligned}
\min \quad & f(\boldsymbol{u}_0) + \frac{1}{K}\sum_{k\in\mathcal{S}} f(\boldsymbol{u}_k) \\
\text{s.t.} \quad & g(\boldsymbol{u}_k, \boldsymbol{x}_k, \boldsymbol{d}_k) = 0, \quad \forall k \in \{0\}\cup\mathcal{S}, \\
& h(\boldsymbol{u}_k, \boldsymbol{x}_k, \boldsymbol{d}_k) \le 0, \quad \forall k \in \{0\}\cup\mathcal{S}, \\
& |\boldsymbol{u}_k - \boldsymbol{u}_0| \le \Delta, \quad \forall k \in \mathcal{S}. \\
\text{var.} \quad & \boldsymbol{x}_k, \boldsymbol{u}_k, \quad \forall k \in \{0\}\cup\mathcal{S}.
\end{aligned}$$

- **Partially Permutation-Invariance**: Optimal stage-1 solution u_0^* remain unchanged upon permutation of inputs $d_1, d_2, \ldots, d_K$

- **Theorem**: under mild conditions, for the 2-stage AC-OPF problem,

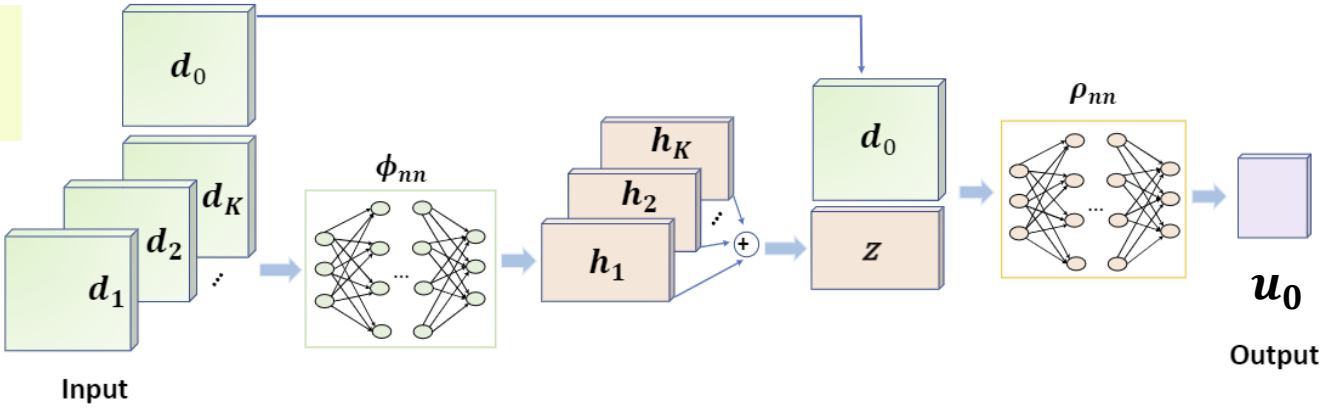
$$u_0^* = \rho\left(d_0, \sum_{k=1}^{K}\phi(d_k)\right) \text{ and } u_k^* = \psi\left(u_0^*, d_k\right), 1 \le k \le K.$$

where ρ, ϕ, and ψ are all continuous a.e. mappings.

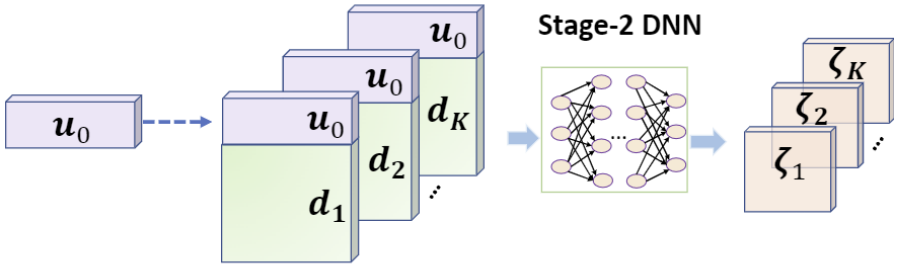
[1] Min Zhou, Enming Liang, Minghua Chen, and Steven H. Low, "Partially Permutation-Invariant Neural Network for Solving Two-Stage Stochastic AC-OPF Problem," IEEE Transactions on Power Systems, vol. 41, no. 2, pp. 1246–1263, 2026.

DeepOPF-Stoc: A Partially Permutation-invariant NN (PPNN) Design

Obtaining stage-1 Solution:



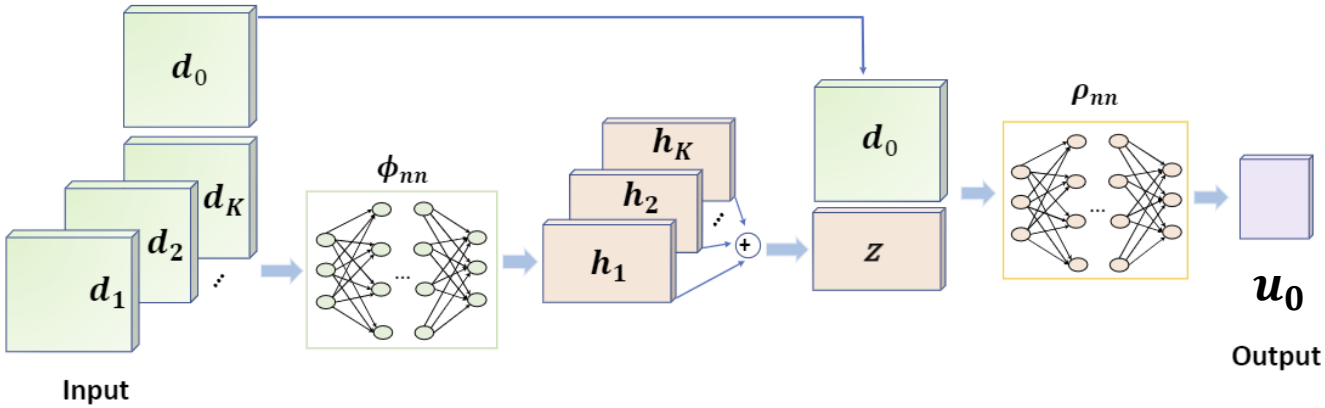
Obtaining stage-2 solutions:



- We design a PPNN to learn the load-to-stage-1 solution mapping
  - Learn ρ and ϕ with NNs
  - Inputs $d_1, \ldots, d_K$ through ϕ_{NN} parallelly
- Learn the stage-2 mapping ψ as a standard AC-OPF one

[1] Min Zhou, Enming Liang, Minghua Chen, and Steven H. Low, "Partially Permutation-Invariant Neural Network for Solving Two-Stage Stochastic AC-OPF Problem," IEEE Transactions on Power Systems, vol. 41, no. 2, pp. 1246–1263, 2026.

PPNN Design Significantly Alleviates Dimensionality Explosion



- The PPNN design: learn ϕ and ρ with neural networks
  - Naturally satisfies the partial permutation-invariance
- Addressing the dimensionality explosion (due to K)
  - PPNN input dimension is independent of K
- **Theorem [1]:** $O(\log K)$ latent dimension (output dimension of ϕ_{nn}) sufficient for universal approximation

[1] M. Zhou, E. Liang, and M. Chen, "O(log N) Latent Dimension Suffices for Universal Approximation of Permutation-invariant Function", ICML, 2026

PPNN Approximation Error

- **Theorem**: There exists a PPNN with neural networks ρ_{nn} and ϕ_{nn}, such that its approximation error to $\mathcal{P}^*$ is bounded by:

$$\epsilon(b_\phi) \leq \epsilon_p \leq \epsilon(b_\phi) + K\,\mathrm{Lip}(\rho^*)\epsilon_{\phi_{\mathrm{nn}}} + \epsilon_{\rho_{\mathrm{nn}}},$$

- Universal Approximation
  - Given sufficient NN size, PPNN can approximate the target mapping arbitrarily well
- The scaling advantage
  - PPNN Error Growth: Increases linearly with K
  - Standard NN Error Growth: Increases exponentially with K

[1] Min Zhou, Enming Liang, Minghua Chen, and Steven H. Low, "Partially Permutation-Invariant Neural Network for Solving Two-Stage Stochastic AC-OPF Problem," IEEE Transactions on Power Systems, vol. 41, no. 2, pp. 1246–1263, 2026.

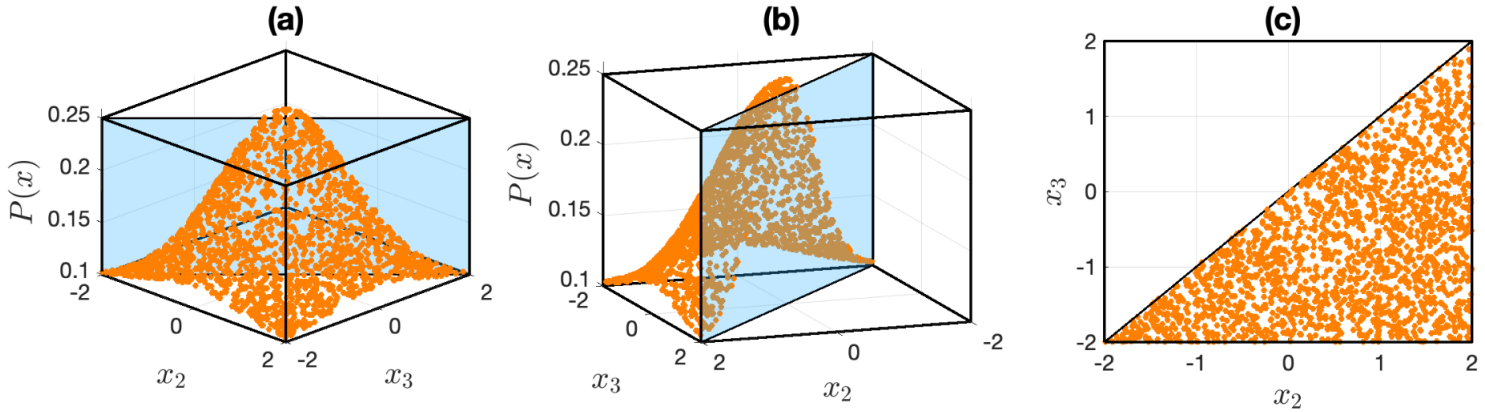
Efficient PPNN-aware Sampling

- **Observation**: ordered data suffices for PPNN training
  - E.g., (0.3,0.4,0.5) and (0.3,0.5,0.4) are the "same" for PPNN

$$\mathcal{P}(\boldsymbol{x}) = x_1 + 2(x_2 + x_3)$$

- **PPNN-aware sampling**: we sample from an ordered subspace where $\boldsymbol{d}_1 > \cdots > \boldsymbol{d}_K$

- **Lemma**: Improve the sample efficiency by a factor of $K!$ (factorial) as compared to uniform sampling



[1] Min Zhou, Enming Liang, Minghua Chen, and Steven H. Low, "Partially Permutation-Invariant Neural Network for Solving Two-Stage Stochastic AC-OPF Problem," IEEE Transactions on Power Systems, vol. 41, no. 2, pp. 1246–1263, 2026.

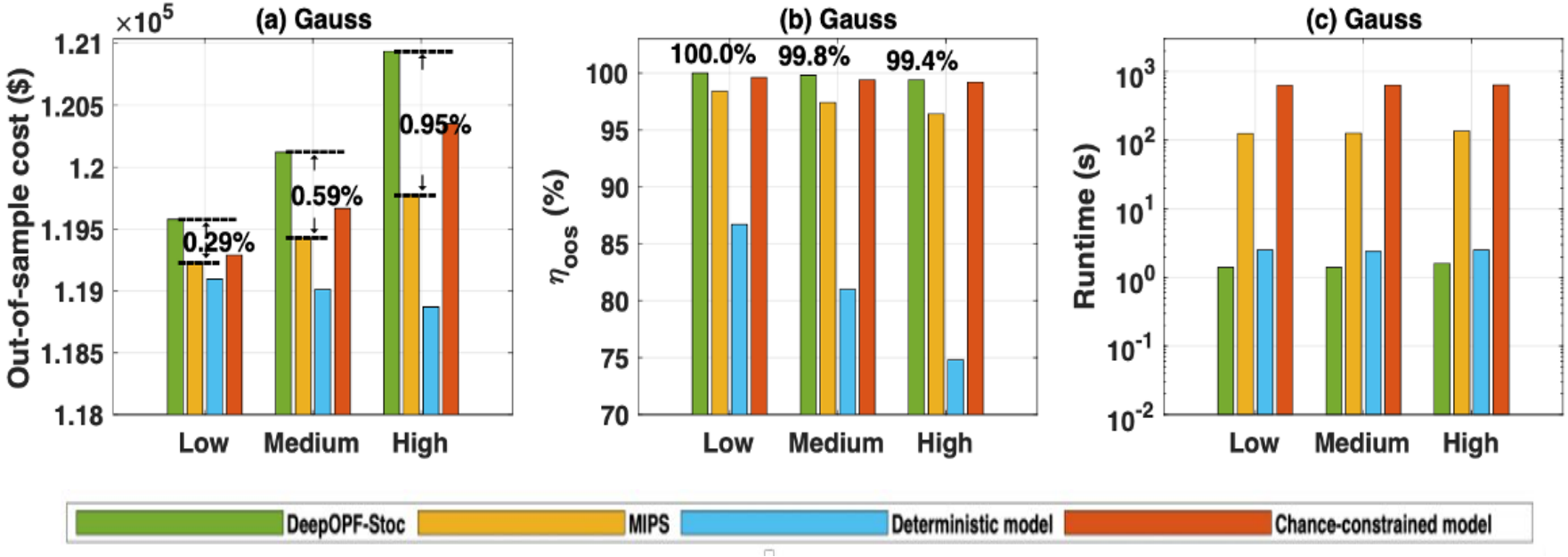
Simulation Setup

- Test Cases
  - **IEEE 118-bus system:** Standard medium-scale benchmark
  - **Synthetic 793-bus system:** Large-scale, highly complex grid to test scalability
- Uncertainty Distributions
  - **Gaussian:** Standard normal fluctuations
  - **Uniform-Laplacian:** Tests heavy-tailed, extreme renewable generation events
  - Each with 3 uncertainty levels (low, medium, high)

Baseline Methods

- Matpower Interior Point Solver (MIPS)
  - Solves the full stochastic problem numerically using interior point methods
- Deterministic AC-OPF
  - Solves the grid dispatch using only the average forecast
- Chance-Constrained AC-OPF (CC-OPF)
  - Method that handles uncertainty by restricting the probability of constraint violations

DeepOPF-Stoc Performance under Different Distributions



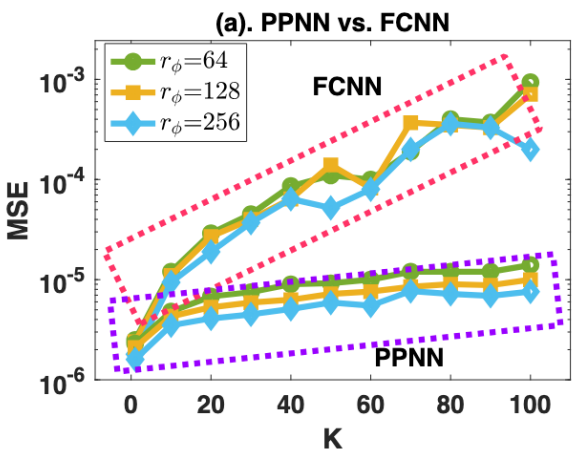
- Over 500x speedup (all over CPU; GPU would bring another 10x)
- Low (within 0.95%) out-of-sample cost difference
  - The generation cost incurred over unseen scenarios
- High (over 99%) out-of-sample feasibility rate
  - The percentage of unseen scenarios where the dispatch decision is feasible

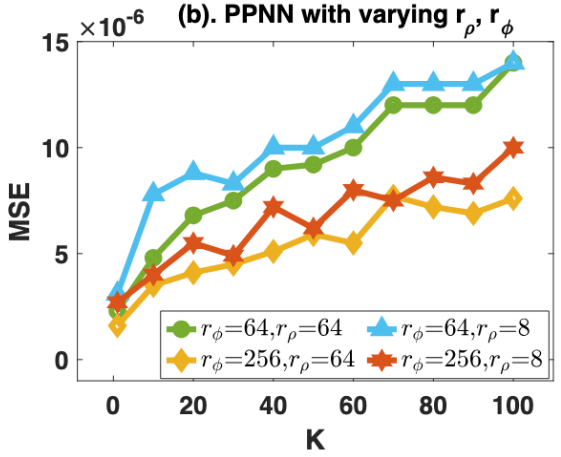
Scalability of DeepOPF-Stoc

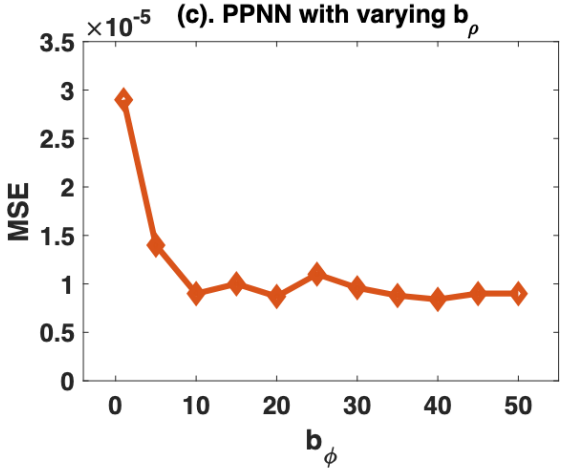
- DeepOPF-Stoc achieves within 0.5% cost gap, over 99% out-of-sample feasibility rate, with up to x230 speedup for the large 793-bus test system

	Metrics	In sample = 5 $\approx$ 20K variables	In sample = 10 $\approx$ 34K variables	In sample = 15 $\approx$ 50K variables
MIPS	C_{oos} ($)	439,230	439,169	439,198
	η_{oos} (%)	93.6	95.6	96.5
	η_{sp}	x1	x1	x1
Deterministic model	C_{oos} ($)	438,261	439,186	439,216
	η_{oos} (%)	84.7	85.8	88.4
	η_{sp}	×8	×29	×61
CC-ACOPF	C_{oos} ($)	439,258	439,180	439,215
	η_{oos} (%)	98.6	98.8	98.8
	η_{sp}	×0.1	×0.1	×0.5
DeepOPF-Stoc	C_{oos} ($)	440,082	441,199	441,461
	η_{oos} (%)	**99.0**	**99.6**	**98.9**
	η_{sp}	**×85**	**×156**	**×230**

PPNN vs. FCNN and Its Latent Dimension





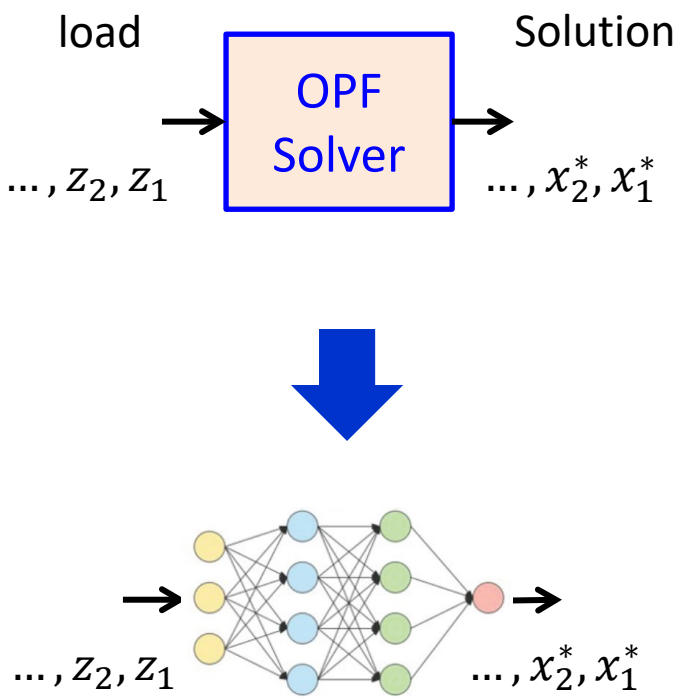


- PPNN vs FCNN: PPNN approximation error is two orders of magnitude lower than a standard FCNN
- Linear Scaling: PPNN error grows linearly with K (echo theoretical analysis)
- Latent Efficiency: small latent dimension suffices to get high accuracy

Summary: ML as an Amortized Solver

- Learn once, solve many times
  - Offline training pays a one-time cost
  - Online deployment replaces iterative updates with fast inference
- AC-OPF is nonconvex, NN solutions must be
  - fast
  - near-optimal
  - physically feasible

ML for Solving OPF is Working



- <0.2% optimality loss AC-OPF simulations over IEEE cases, real-world topology, and loads by various NN schemes
  - Come with theoretical justification
  - 15,000x speedup over a 2000-bus network; 10% load variation
  - Evaluated over actual RTE networks with 9,241 buses, actual Korea-4492 system
  - Can also be used to generate a preliminary evaluation quickly
- Generalizable approaches
  - PR2 to ensure equality feasibility
  - Unsupervised learning to reduce training data complexity
  - Decomposition to improve scalability
  - Permutation-invariant NN design for 2-stage stochastic optimization problems

Beyond Single-Valued Deterministic OPF

A. Multi-valued AC-OPF

Problem
standard supervised NN can average incompatible solution branches

↓

Approach 1: augmented learning
learn from (load, initial point) → solution

↓

Approach 2: generative learning
learn input-dependent solution distributions

Message: learn one legitimate branch or sample high-quality solutions — do not force a single averaged output.

B. Two-stage stochastic AC-OPF

Problem
K uncertainty scenarios cause dimension explosion

↓

Structure
stage-1 decision is invariant to scenario order

↓

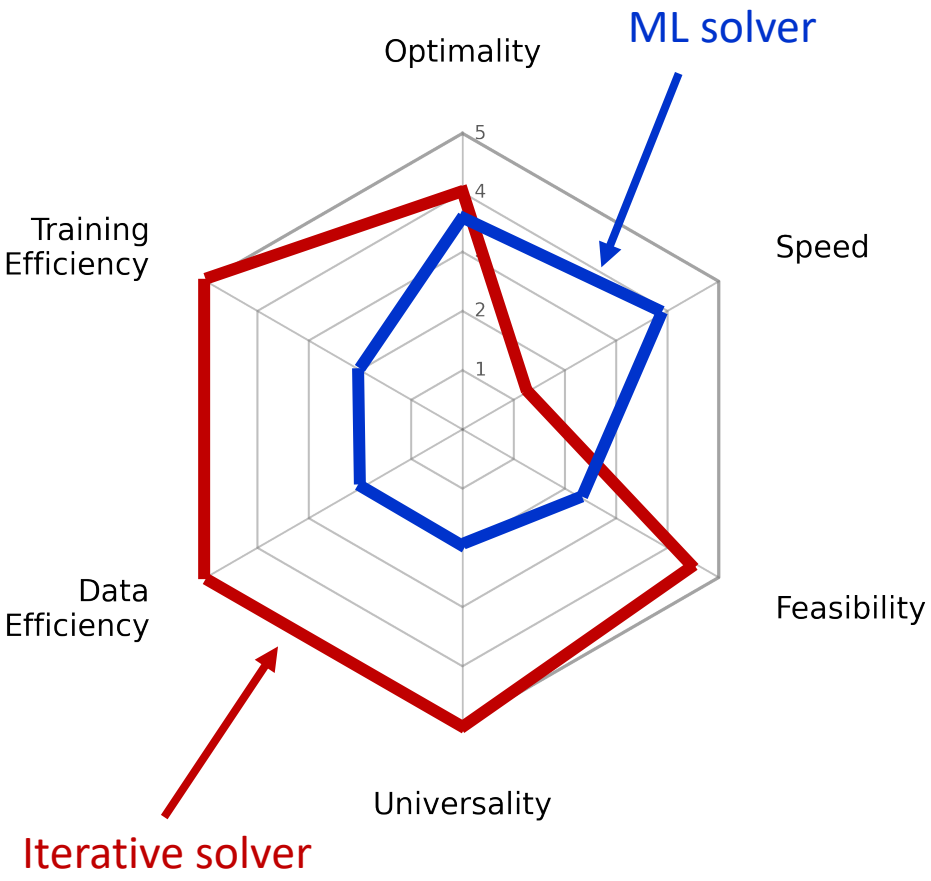
PPNN / DeepOPF-Stoc
input dimension independent of K

Reported behavior: >500× speedup, >99% out-of-sample feasibility, <0.95% cost difference; up to 230× on 793-bus tests.

Summary of Challenges and Approaches

- NN solutions may be infeasible
  - Predict-and-reconstruct (PR2) / equation completion
  - Primal-dual training (utilizing KKT conditions)
  - Differentiable NN layers
  - More advanced techniques to be covered in **Part III**
- High training/data complexity for large-scale OPF problems
  - Un-/self- supervised learning, GNN, grid decomposition, compact learning, approximate data labels
- The multi-valued mapping issue for non-convex (AC-OPF) problems
  - Augmented learning, data preparation/selection
  - Generative learning (learning the input-dependent solution distributions)
- 2-stage stochastic ACOPF problems incurs dimensionality explosion (NN input dimension linear in K)
  - Permutation-invariant design to address dimensionality explosion with desirable theoretical guarantee
  - Input dimension independent of K (#scenarios), laten dimension $O(\log K)$

Six Dimensions for Comparing Solvers



- **Feasibility**: solution feasibility
- **Optimality**: solution optimality
- **Speed**: run-time solution speed
- **Universality**: one solver for all problems
- **Data efficiency**: minimum data-preparation effort
- **Training efficiency**: minimum training effort

Outline

- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems

- Concluding remarks and open challenges

Three Questions After Fast NN Prediction

1. Can the output satisfy OPF constraints?

Prediction errors can violate power balance, voltage, or line limits.
We discuss feasibility enforcement, recovery, and verification.

2. Can the ML model adapt to grid topology changes?

Contingencies, switching, and reconfiguration change the graph.
We discuss topology embedding, GNN and its expressivity.

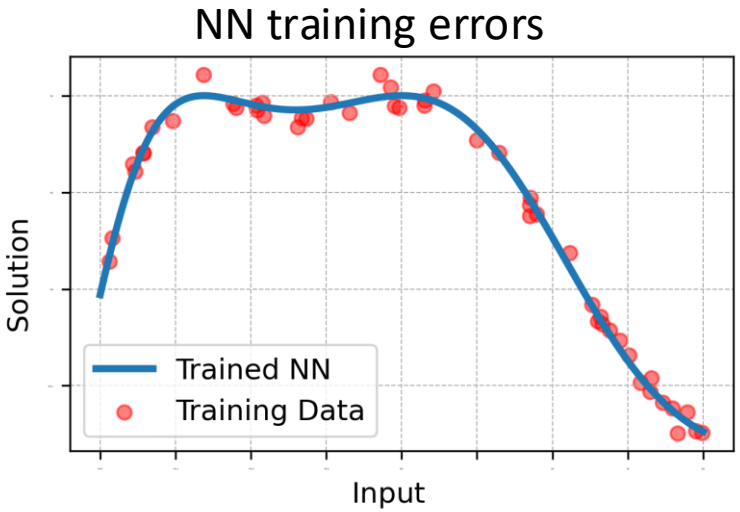
3. Can one ML model generalize across OPF scenarios?

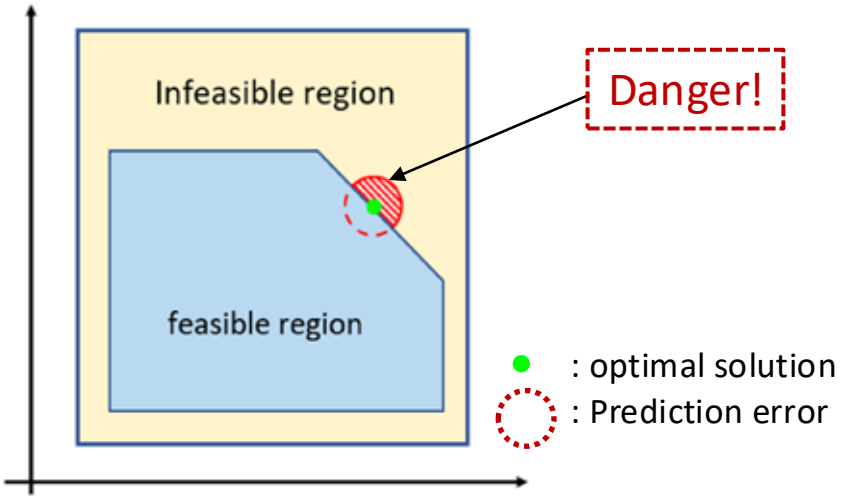
OPF varies by grid scale, formulation, and operating scenario.
We discuss OPF foundation models and LLM tool workflows.

Part I: Ensuring NN Solution Feasibility for OPF

From fast predictions to safe operations

NN Prediction Errors Can Break Feasibility





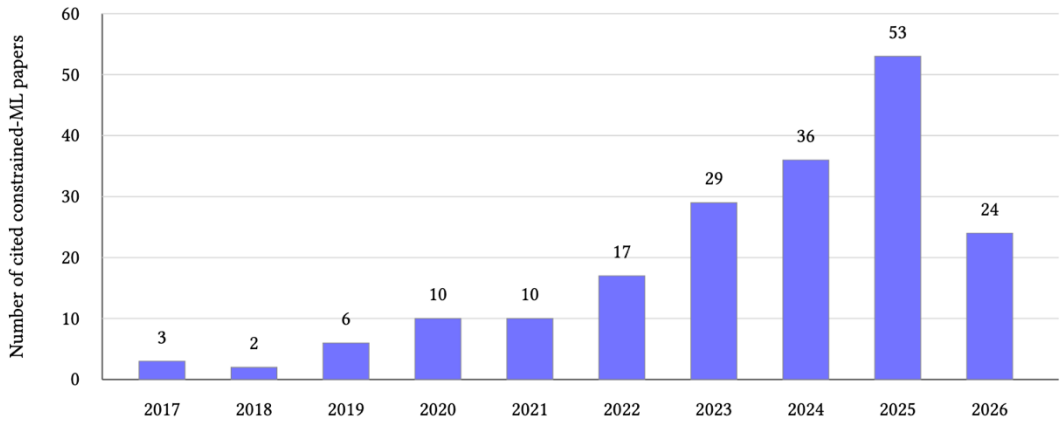
Why can this happen?

- NN outputs are approximated dispatch solutions
- OPF optima lie in active constraint boundaries
- Small prediction errors can cross the feasible boundary
- Violations can invalidate the dispatch for grid operation

How to enforce NN prediction feasibility

NN Feasibility Is a General Constrained ML Problem

Enforcing NN feasibility is not limited to OPF problems. It is an active topic in main ML venues, including both constrained prediction and constrained generation:



Non-exhaustive statistics

Key words:
machine learning;
hard constraints;
neural network; …

Venues:
ICML, ICLR, NeurIPS

More applications:

- Prediction: OPF dispatch, trajectory planning, control actions, …
- Generation: molecule generation, material design, image watermark, …

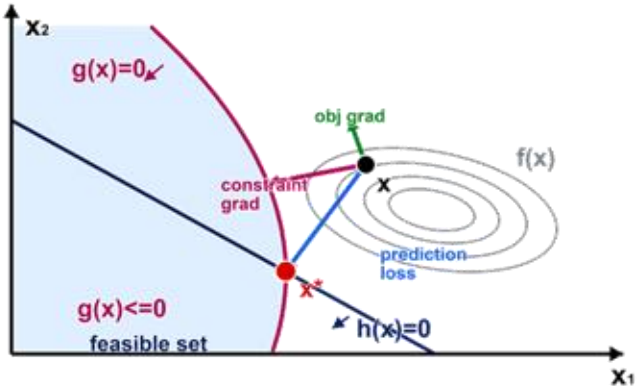
This tutorial focuses on NN feasibility for OPF

Three Ways to Handle Feasibility
Tradeoff among applicability, quality, and complexity

1. Neural Network training

Training to change NN weights to enforce feasibility:

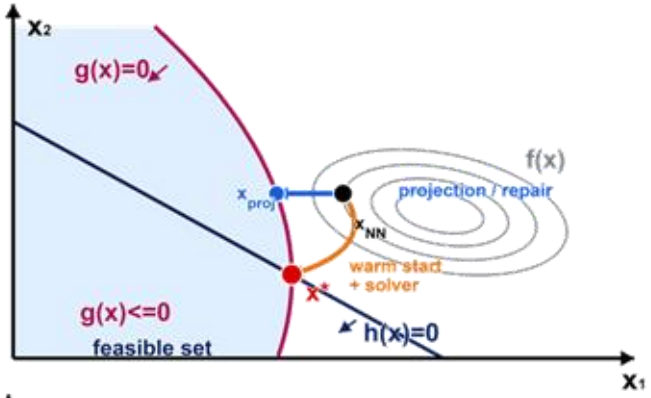
- Soft feasibility : Penalty, primal-dual, physics-informed, …
- Verified feasibility: Preventive learning, NN editing,…



2. Refine after Prediction

Repair NN prediction with a refinement step:

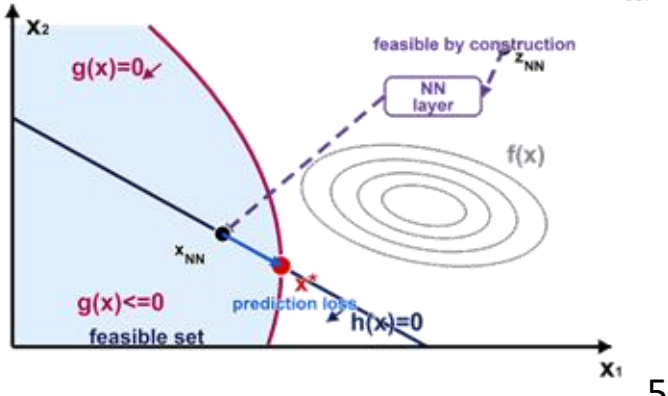
- Expensive: L_2/L_1 projection, solver-based, …
- Cheaper: Homeomorphic projection, bisection, …



3. Feasibility by Construction

Design NN layers so constraints hold by construction.

- Convex: convex hull, gauge-based methods, …
- General: Optimization layers (unrolled or implicit)
  - Equality: Predict-and-Reconstruct, completion, …



Neural Network Training
Penalty, Lagrangian, and KKT Loss Terms

NN training approaches:

Tune NN weights toward feasible and optimal OPF solutions.

- Prediction loss, violation penalties, Lagrangian terms, KKT residuals.

Flexible, but finite data and nonconvex NN training leave **residual errors**

OPF problems

$$\begin{aligned} \min_{x} \quad & f(x) \\ & \text{generation cost} \\ \text{s.t.} \quad & h(x) = 0 \\ & \text{power balance constraints} \\ & g(x) \leq 0 \\ & \text{physical and operational limits} \end{aligned}$$

Training loss options for $\hat{x} = NN(P_D, Q_D)$

$$\mathcal{L}_{\text{pred}} = \underbrace{\|\hat{x} - x^\star\|_2^2}_{\text{MSE prediction}}$$

$$\mathcal{L}_{\text{feas}} = \underbrace{\|h(\hat{x})\|_2^2}_{\text{equality residual}} + \underbrace{\|[g(\hat{x})]_+\|_2^2}_{\text{inequality violation}}$$

$$\mathcal{L}_{\text{dual}} = \underbrace{\mu^\top h(\hat{x})}_{\text{equality dual}} + \underbrace{\nu^\top [g(\hat{x})]_+}_{\text{inequality dual}}$$

$$\mathcal{L}_{\text{KKT}} = \mathcal{L}_{\text{feas}} + \underbrace{\|\nabla f(\hat{x}) + J_h(\hat{x})^\top \mu + J_g(\hat{x})^\top \nu\|_2^2}_{\text{stationarity residual}} + \underbrace{\|\nu \odot g(\hat{x})\|_2^2}_{\text{complementarity}}$$

X. Pan, T. Zhao, and M. Chen, *DeepOPF: Deep Neural Network for DC Optimal Power Flow*, IEEE SmartGridComm, 2019.
F. Fioretto, T. W. K. Mak, and P. Van Hentenryck, *Predicting AC Optimal Power Flows: Combining Deep Learning and Lagrangian Dual Methods*, AAAI, 2020.
L. Zhang, Y. Chen, and B. Zhang, *A Convex Neural Network Solver for DCOPF with Generalization Guarantees*, arXiv, 2020.
R. Nellikkath and S. Chatzivasileiadis, *Physics-Informed Neural Networks for AC Optimal Power Flow*, Electr. Power Syst. Res., 2022.
S. Park and P. Van Hentenryck, *Self-Supervised Primal-Dual Learning for Constrained Optimization*, AAAI, 2023.

Neural Network Training
Hard Feasibility with Verification

Verification approaches:

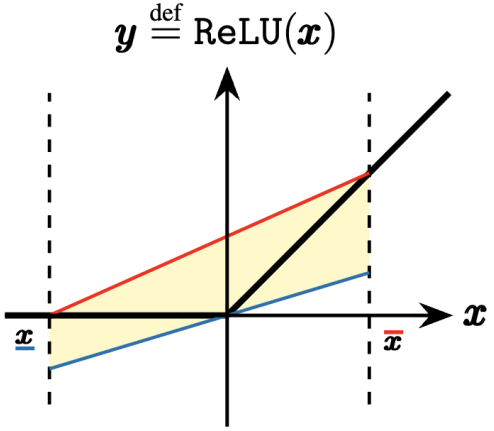
Certify constraint residual errors over input, and *accommodate* the error.

- preventive learning, neural weight editing, …

Lead to **conservative** solution region due to relaxation/tightening, and **limited** to simple constraint set

Neural editing problem for polytope constraints

$$\min \|\dot{\boldsymbol{\theta}} - \theta\| \quad \text{s.t.} \quad \forall \mathbf{x} \in \mathrm{P}.\, \mathcal{N}(\mathbf{x}; \dot{\boldsymbol{\theta}}) \in \mathrm{Q}$$



- Find neural weight near a well-trained NN predictor
- While satisfying output linear constraint over arbitrary input
- Solve it via linear relaxation of non-linear ReLU activation

V. Tjeng, K. Xiao, and R. Tedrake, *Evaluating Robustness of Neural Networks with Mixed Integer Programming*, ICLR, 2019.
A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, *Learning Optimal Power Flow: Worst-Case Guarantees for Neural Networks*, IEEE SmartGridComm, 2020.
T. Zhao, X. Pan, M. Chen, and S. H. Low, *Ensuring DNN Solution Feasibility for Optimization Problems with Convex Constraints*, arXiv, 2021. ICLR 2023.
Z. Tao and A. V. Thakur, *Provable Editing of Deep Neural Networks Using Parametric Linear Relaxation*, NeurIPS, 2024 [verify].

Refine after Prediction: Projection and Warm-start

Solver-based refinement

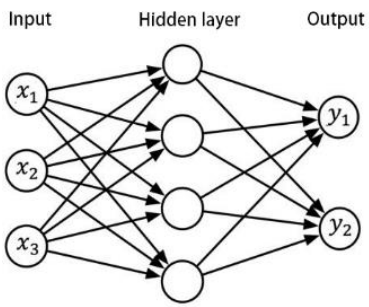
Start from (infeasible) NN output $\hat{x}$, then recover a feasible dispatch.

- L1 Projection for DC-OPF; Warm-start for AC-OPF

High-quality solution, but **runtime** depends a solver call, may be slow for non-convex constraints

DeepOPF for (SC)-DCOPF

Bus load p_D



Generation p_G^{DNN}

$$
\begin{aligned}
\min \quad & \left\| \boldsymbol{p}_G - \boldsymbol{p}_G^{DNN} \right\|_1 \\
\text{s.t.} \quad & \mathbf{B}^c \boldsymbol{\theta}^c = \boldsymbol{p}_G - \boldsymbol{p}_D, \quad \forall c \in \mathcal{C}, \\
& b_{ij}^c \left(\theta_i^c - \theta_j^c \right) \le S_{ij}^{\max}, \quad \forall (i,j) \in \mathcal{E}, c \in \mathcal{C}, \\
& \boldsymbol{P}_G^{\min} \le \boldsymbol{p}_G \le \boldsymbol{P}_G^{\max}, \\
\text{var.} \quad & p_{Gi}, \forall i \in \mathcal{N}, \theta_i^c, \forall i \in \mathcal{N}, c \in \mathcal{C}.
\end{aligned}
$$

- The l_1-projection problem is essentially an LP;
- complexity lower than solving the original QP-based DCOPF

X. Pan, T. Zhao, and M. Chen, *DeepOPF: Deep Neural Network for DC Optimal Power Flow*, IEEE SmartGridComm, 2019.
K. Baker, *Learning Warm-Start Points for AC Optimal Power Flow*, IEEE MLSP, 2019.
B. Taheri and D. K. Molzahn, *Not All Warm Starts Help: Benchmarking Primal-Dual Initializations for ACOPF Algorithms*, arXiv, 2026.

Refine after Prediction: Bisection-based Approaches

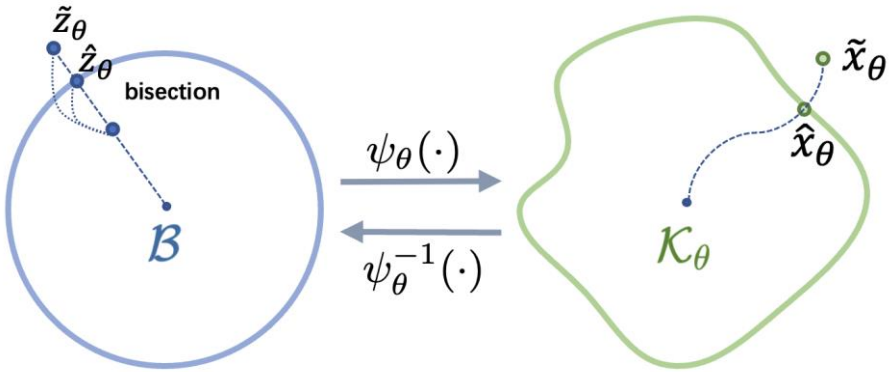
Projection-analogous repair:

Recover feasibility with low-complexity geometry-aware maps.

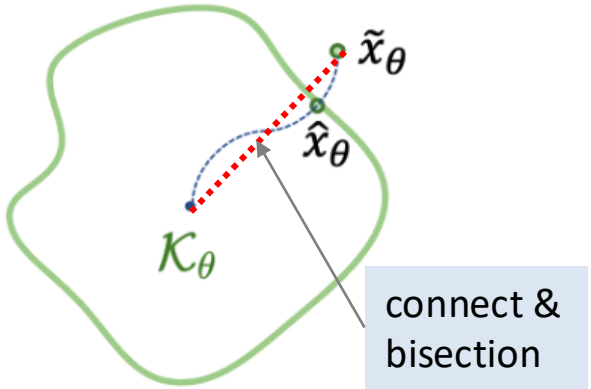
- Project or search along a structured path to the feasible set

Faster repair, but at cost of additional objective **optimality loss** and depend on the constraint geometry

Homeomorphic Projection



Direct Bisection



Bisection-based method for low-complexity feasibility recovery

E. Liang, M. Chen, and S. Low, *Low Complexity Homeomorphic Projection to Ensure NN Solution Feasibility for Optimization over (Non-)Convex Set*, ICML, 2023.
E. Liang, M. Chen, and S. Low, *Homeomorphic Projection to Ensure Neural Network Solution Feasibility for Constrained Optimization*, JMLR, 2024.
E. Liang and M. Chen, *Efficient Bisection Projection to Ensure Neural Network Solution Feasibility over General Set*, ICML, 2025.

Feasibility by Construction: Optimization Layers for Equality

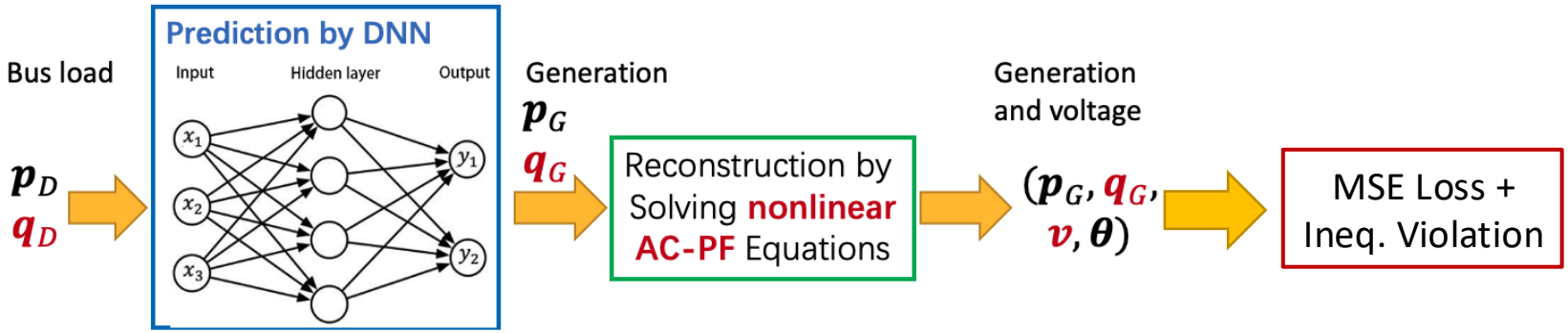
Optimization Layers

Embed a constrained solve inside the NN

- Forward solving feasibility / violation-minimization iterations
- Backpropagate by zero-order estimation, implicit gradients, unrolling, …

Tradeoff between **feasibility/quality** and **complexity** in forward solving settings.

Predict-and-Reconstruct (PR2) for Equality Constraints



- **Forward**: predict partial variable and reconstruct rest via AC-PF solver
- **Backward**: zero-order estimation for gradient backpropagation

X. Pan, T. Zhao and M. Chen, *DeepOPF: Deep Neural Network for DC Optimal Power Flow*, SmartGridComm, 2019.

X. Pan, M. Chen, T. Zhao, and S. H. Low, *DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems*, arXiv, 2020.

Feasibility by Construction: Optimization Layers for Both

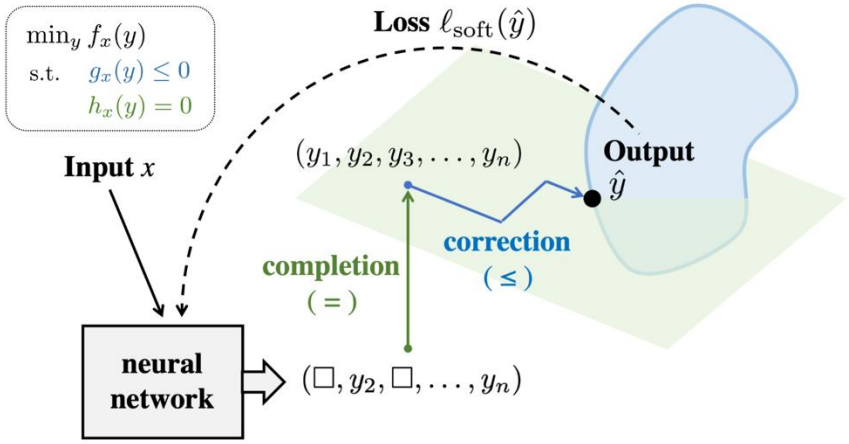
Optimization Layers

Embed a constrained solve inside the NN

- Forward solving feasibility / violation-minimization iterations
- Backpropagate by zero-order estimation, implicit gradients, unrolling, …

Tradeoff between **feasibility/quality** and **complexity** in forward solving settings.

Deep Constraint Completion and Correction (DC3) for Eq. and Ineq. Constraints



- **Equality**: predict partial var. and complete rest via Newton's methods (backward via *implicit gradient*)
- **Inequality**: gradient step for violation minimization of inequality constraint (backward via *unrolled iteration*)

P. L. Donti, D. Rolnick, and J. Z. Kolter, *DC3: A Learning Method for Optimization with Hard Constraints*, ICLR, 2021.
M. Kim and H. Kim, *Self-Supervised Equality Embedded Deep Lagrange Dual for Approximate Constrained Optimization*, arXiv, 2023.
H. T. Nguyen and P. L. Donti, *FSNet: Feasibility-Seeking Neural Network for Constrained Optimization with Guarantees*, arXiv, 2025.

Feasibility by Construction: Constraint Parameterizations

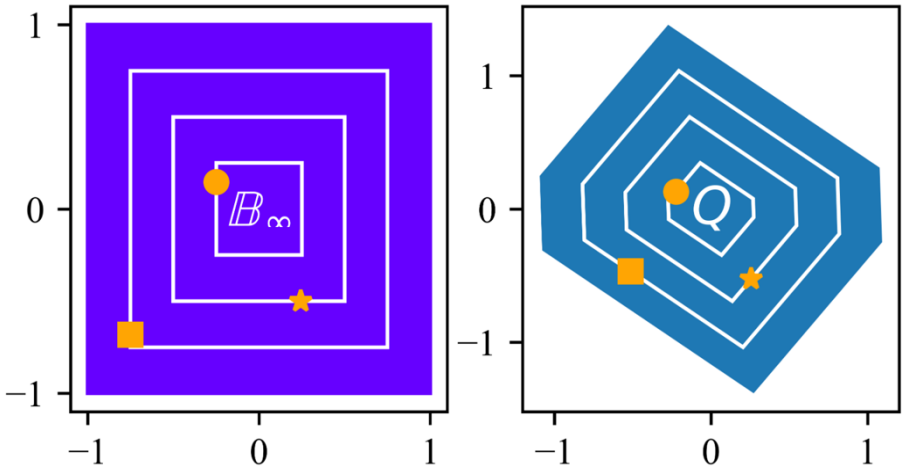
Constraint Parameterizations

Predict latent z and map x = Φ(z) to the feasible set

- Convex hull based, gauge function based, …

Fast and feasible when **constraint geometry** is exploitable, e.g, convex, star-convex, …

Gauge mapping between cube and polytope



- Constrained NN output in a cube via *Tanh/Sigmoid* activation function
- Then map any point in cube to a point in polytope via closed-form gauge mapping

L. Zheng, Y. Shi, L. J. Ratliff, and B. Zhang, *Safe Reinforcement Learning of Control-Affine Systems with Vertex Networks*, L4DC, 2021.
D. Tabas and B. Zhang, *Computationally Efficient Safe Reinforcement Learning for Power Systems*, IEEE ACC, 2022.
D. Tabas and B. Zhang, *Safe and Efficient Model Predictive Control Using Neural Networks: An Interior Point Approach*, IEEE CDC, 2022.

Survey on Machine Learning under Hard Constraints

Contents

Hard-Constrained Machine Learning

What is satisfied, useful, and affordable?

1 Constraint satisfaction

What supports feasibility?

- Guarantee type
- Violation residuals
- Tolerances

2 Output utility

Is the constrained output useful?

- Task accuracy
- Sample quality
- Downstream utility

3 Training and inference cost

What is paid for enforcement?

- Training overhead
- Inference latency
- Solver / sampler calls
- Memory use

4 Scope and assumptions

Where does the claim apply?

- Constraint class
- Input or sample domain
- Solver / oracle / map validity
- Discretization assumptions

Github page

E. Liang and M. Chen, *A Survey on Machine Learning with Hard Constraints*, in submission, 2026.

Takeaways: Enforce Neural Network Feasibility

1. Neural Network training

Training to change NN weights to enforce feasibility:

- Soft feasibility : Penalty, primal-dual, physics-informed, …
- Verified feasibility: Preventive learning, NN editing,...

2. Refine after Prediction

Repair NN prediction with a refinement step:

- Expensive: L_2/L_1 projection, solver-based, …
- Cheaper: Homeomorphic projection, bisection, …

3. Feasibility by Construction

Design NN layers so constraints hold by construction.

- Convex: convex hull, gauge-based methods, …
- General: Optimization layers (unrolled or implicit), …

- No single mechanism dominates, feasibility handling is a tradeoff among accuracy, guarantees, runtime, and training complexity.

- Hard-constrained NN/ML feasibility is not limited to OPF problems, more methods can be explored and applied.

Part II: Which Neural Structure Fits OPF Prediction?

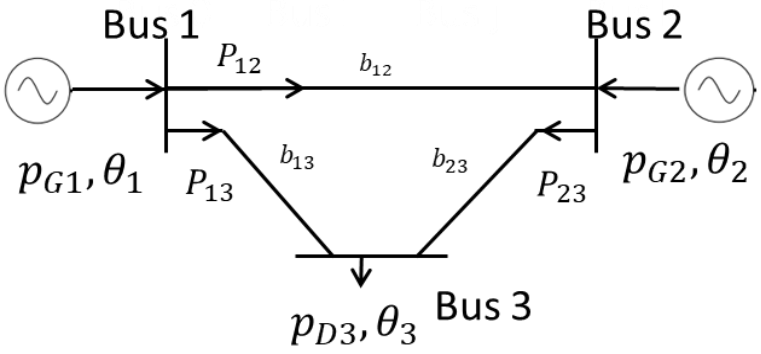
From flat MLPs to topology-aware models

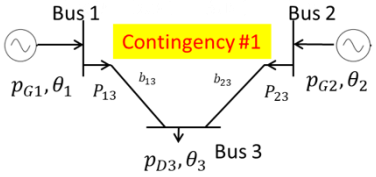
Flat DNNs Fail Under Topology Change

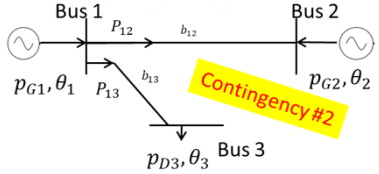
Grid operation changes the topology:

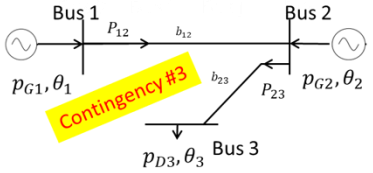
- Contingencies (lines, generators, or transformers)
- Different grid configurations lead different load-to-solution maps

A flat DNN learns a load-to-solution map for one fixed grid and fails to generalize to unseen grids.









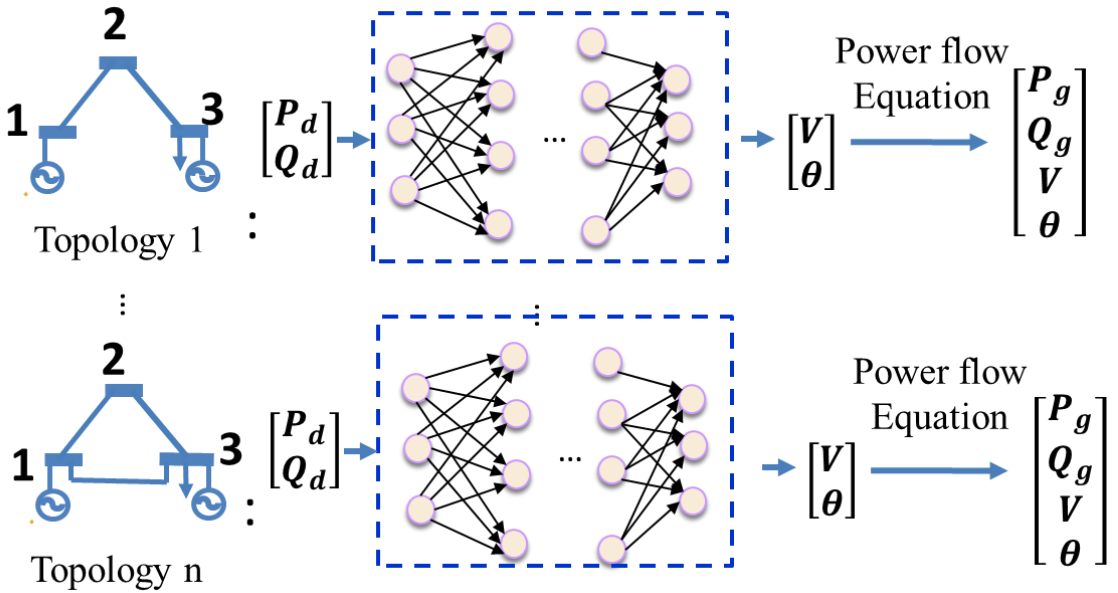
How do NN models adapt to different topologies

Training and Fine-tuning based methods

Training-based solution:

- Train one DNN per topology
- Re-train / fine-tune given a new topology

These fixes create **high** computation burden, storage cost, or operation delay.



How to Embed Topology in Neural Model?

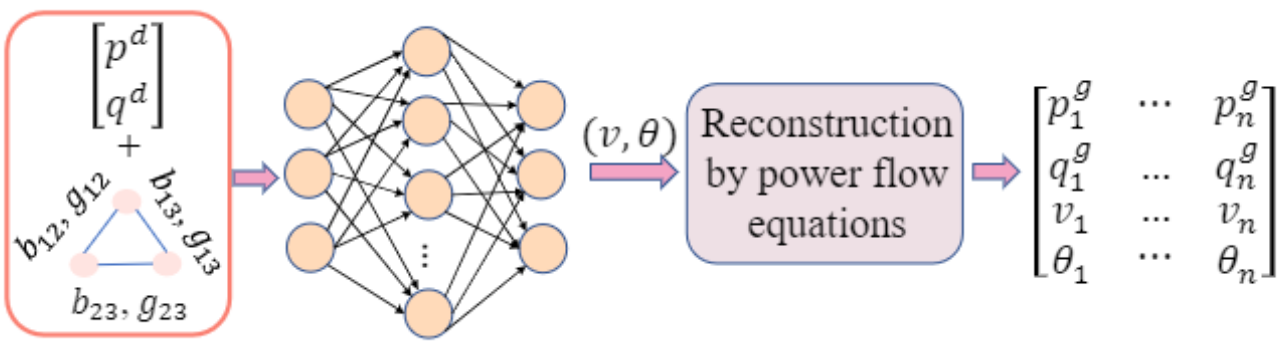
Y. Chen, S. Lakshminarayana, C. Maple, et al., *A Meta-Learning Approach to the Optimal Power Flow Problem Under Topology Reconfigurations*, IEEE Open Access J. Power Energy, 2022.

Admittance Encoding: Put Topology into the Input

Admittance encoding:

- Embed the discrete network representation into the continuous admittance space
- Use DNN to learn the mapping from (load, admittance) to bus voltages of an AC-OPF solution.

Limited **scalability** as high input dimension when grid size grows



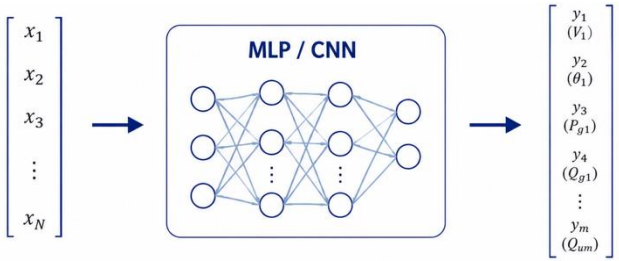
A DNN is used to learn the mapping from (p_d, q_d, b, g) to (v, θ). The remaining variables, i.e., (p_g , q_g), are computed by using the power flow equations.

M. Zhou, M. Chen, and S. H. Low, *DeepOPF-FT: One Deep Neural Network for Multiple AC-OPF Problems with Flexible Topology*, IEEE TPWRS, 2022.
Y. Jia, X. Bai, L. Zheng, Z. Weng, and Y. Li, *ConvOPF-DOP: A Data-Driven Method for Solving AC-OPF Based on CNN Considering Different Operation Patterns*, IEEE TPWRS, 2021

From Flat Predictors to Topology-Aware Neural Structures

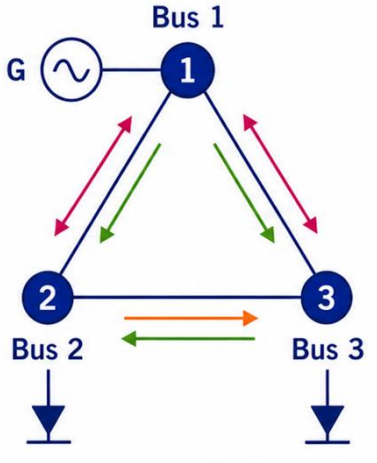
Flat MLP / CNN predictors:

+ Work well for fixed-size, fixed-topology OPF
+ *Universal approximation guarantees*
– fixed bus/line order and fixed input/output dimension
– Topology changes need retraining, fine-tuning, or extra encoding



Graph neural network (GNN)

- Power grid is a graph: buses and lines are graph entities
- Learning shared local update rules along grid connectivity
- Can be reused empirically, when topology or grid size changes



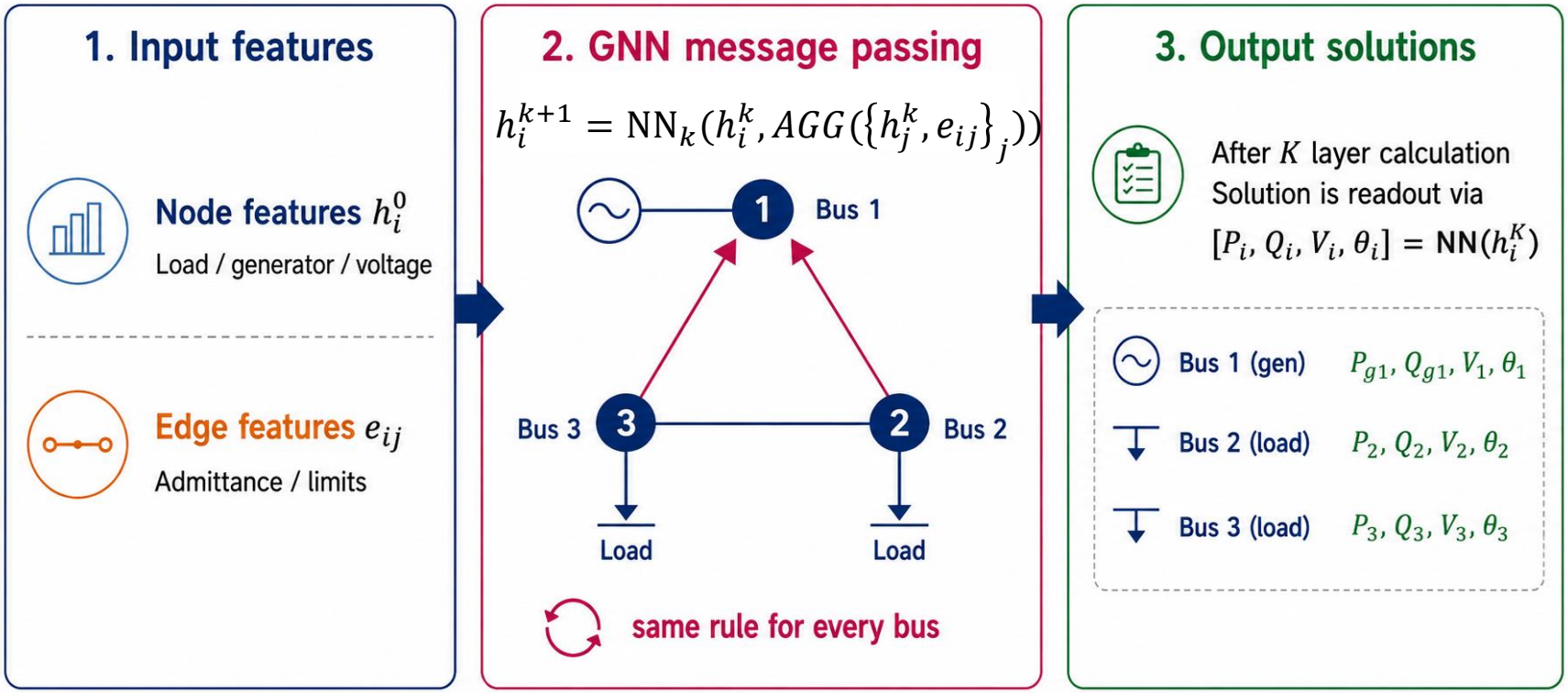
What is the design space of GNN for OPF?
When is it expressive enough for OPF?

Y. Jia, and X. Bai, *A CNN approach for optimal power flow problem for distribution network*. IEEE PSGEC. 2021.
K. Yang, W.Gao, and R. Fan, *Optimal power flow estimation using one-dimensional convolutional neural network*. IEEE NAPS. 2021
M. Zhou, M. Chen, and S. H. Low, *DeepOPF-FT: One Deep Neural Network for Multiple AC-OPF Problems with Flexible Topology*, IEEE TPWRS, 2022.
Y. Jia, X. Bai, L. Zheng, Z. Weng, and Y. Li, *ConvOPF-DOP: A Data-Driven Method for Solving AC-OPF Based on CNN Considering Different Operation Patterns*, IEEE TPWRS, 2022.
F. Diehl, *Warm-Starting AC Optimal Power Flow with Graph Neural Networks*, NeurIPS Workshop, 2019.
D. Owerko, F. Gama, and A. Ribeiro, *Optimal Power Flow Using Graph Neural Networks*, IEEE ICASSP, 2020.

Basic GNN: Message Passing between Buses

Graph neural network:

- Nodes: buses, generators, loads; Edges: lines, admittance, limits, …
- Learning local node/edge updating function given neighbor input



F. Diehl, *Warm-Starting AC Optimal Power Flow with Graph Neural Networks*, NeurIPS Workshop, 2019.
D. Owerko, F. Gama, and A. Ribeiro, *Optimal Power Flow Using Graph Neural Networks*, IEEE ICASSP, 2020.

GNN for OPF: Structure Design

Graph

- *Homogeneous*: buses as nodes, lines as edges, different devices as features
- *Heterogeneous*: different devices as different node/edge with type-dependent embedding

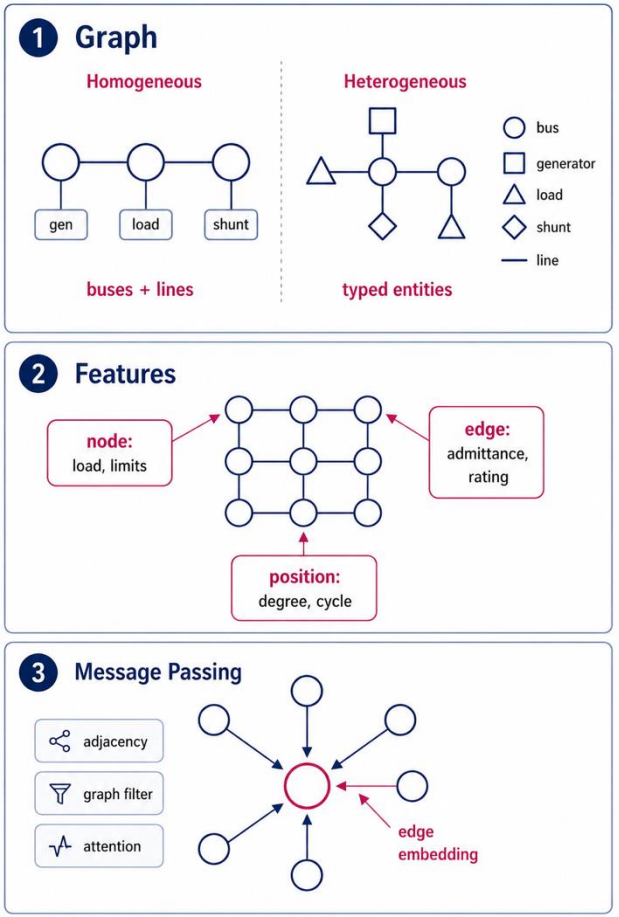
Features

- *Node*: load, limits, bus type,…
- *Edge*: admittance, rating, line status,…
- *Position*: degree, distance, cycle,…

Message Passing

- Neighbor aggregation: grid adjacency, graph filter, global attention
- Edge update: fixed or learned edge embedding

Note: GNN is a structure choice: it can be combined with different OPF losses, feasibility mechanisms, and task heads.



F. Diehl, *Warm-Starting AC Optimal Power Flow with Graph Neural Networks*, NeurIPS Workshop, 2019.
D. Owerko, F. Gama, and A. Ribeiro, *Optimal Power Flow Using Graph Neural Networks*, IEEE ICASSP, 2020.
S. Liu, C. Wu, and H. Zhu, *Topology-Aware Graph Neural Networks for Learning Feasible and Adaptive AC-OPF Solutions*, IEEE TPWRS., 2022.
T. Pham and X. Li, *Reduced Optimal Power Flow Using Graph Neural Network*, IEEE NAPS, 2022.
M. Gao, J. Yu, Z. Yang, and J. Zhao, *A Physics-Guided Graph Convolution Neural Network for Optimal Power Flow*, IEEE TPWRS., 2023.
T. Pham and X. Li, *N-1 Reduced Optimal Power Flow Using Augmented Hierarchical Graph Neural Network*, arXiv, 2024.
T. B. Lopez-Garcia and J. A. Dominguez-Navarro, *Optimal Power Flow with Physics-Informed Typed Graph Neural Networks*, IEEE TPWRS., 2024.
S. Ghamizi, A. Ma, J. Cao, and P. R. Cortes, *OPF-HGNN: Generalizable Heterogeneous Graph Neural Networks for AC Optimal Power Flow*, IEEE PESGM, 2024.
M. Yang, G. Qiu, J. Liu, et al., *Topology-Transferable Physics-Guided Graph Neural Network for Real-Time Optimal Power Flow*, IEEE TII., 2024.

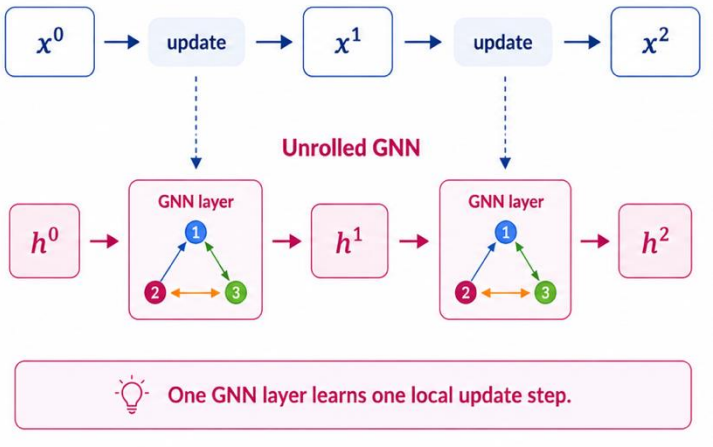
Note: Selected references are shown; many other GNN-OPF variants and applications are omitted for space.

GNN Expressive Power: Two theoretical views

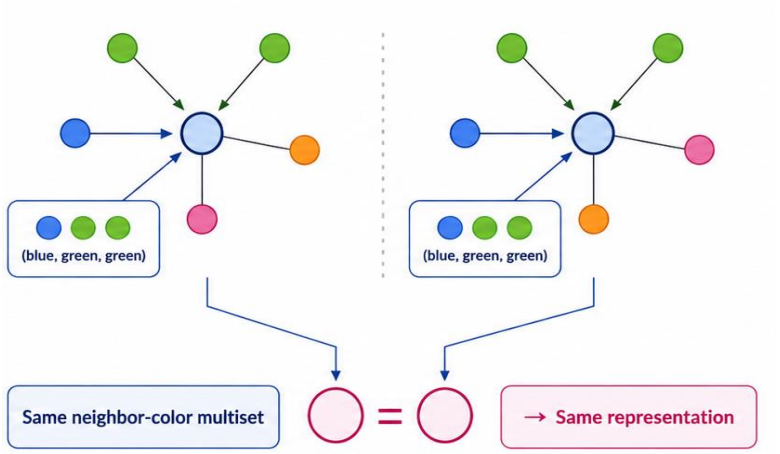
MLP/CNN can approximate general continuous functions.

How about GNN → Two views

- Algorithm unrolling: one layer can simulate one solver iteration step



- WL tests: GNNs distinguish graphs through neighborhood aggregation



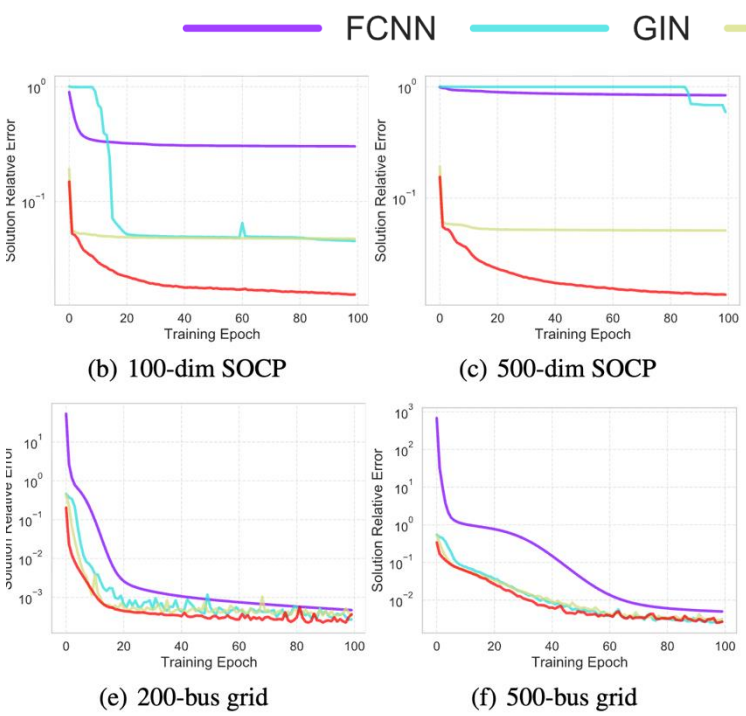
Known theory: GNN are universal approximator for solution mapping of LP[1,2], QP[3,4], SOCP[5], and SDP[6], but not hold for general non-convex QCQP[4].

1. Z. Chen, J. Liu, X. Wang, J. Lu, and W. Yin, *On Representing Linear Programs by Graph Neural Networks*, ICLR, 2023.
2. Q. Li, T. Ding, L. Yang, M. Ouyang, Q. Shi, and R. Sun, *On the Power of Small-Size Graph Neural Networks for Linear Programming*, NeurIPS, 2024.
3. Z. Chen, X. Chen, J. Liu, X. Wang, and W. Yin, *Expressive Power of Graph Neural Networks for (Mixed-Integer) Quadratic Programs*, ICML, 2025.
4. C. Wu, Q. Chen, A. Wang, et al., *On Representing Convex QCQPs via Graph Neural Networks*, TMLR, 2025.
5. R. Li, E. Liang, and M. Chen, *On the Universality and Complexity of GNN for Solving Second-Order Cone Programs*, ICLR, 2026.
6. C. Qian and C. Morris, *On the Expressive Power of GNNs to Solve Linear SDPs*, ICML, 2026.

GNN Expressive Power: Implications for OPF

Theoretical implications on GNN capability for OPF:

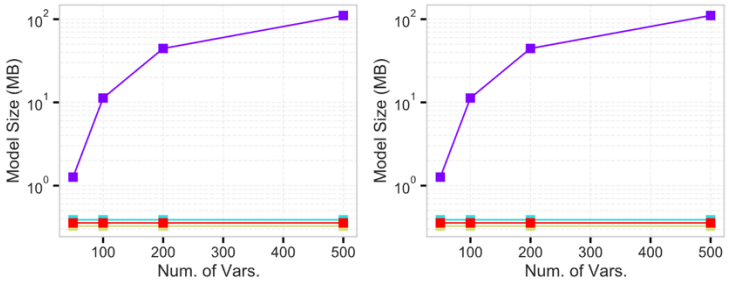
- One GNN can universally approximate the solution mapping of *DC-OPF, SOCP-OPF, SDP-OPF* in the fixed grid size under different topologies.



(b) 100-dim SOCP (c) 500-dim SOCP

(e) 200-bus grid (f) 500-bus grid

GNN achieves better approximation accuracy with 3 orders of magnitude fewer parameters than FCNN (MLP) for solving SOCP.



(c) Model size in SOCP (d) Model size in SoC-OPF

Open issues: small-to-large size generation?, non-convex AC-OPF?, …

R. Li, E. Liang, and M. Chen, *On the Universality and Complexity of GNN for Solving Second-Order Cone Programs*, ICLR, 2026.

Takeaways: Neural Structures for OPF

1. MLP/CNN can be strong model on one fixed grid

- fixed bus / line order / fixed input / output size
- Need retrain / fine-tune given topology change

2. Topology can be encoded in neural models

- **Input encoding**: flatten admittance as vector features (limited scalability)
- **Architecture encoding**: grid graph defines neural models (parameter efficiency)

GNN is a structure choice: graph/feature/message passing/…

- It can be combined with different OPF losses, training methods, feasibility mechanisms, and task heads, …

Expressivity theory gives guidance, not a full answer:

- GNNs can approximate solution maps for convex OPF relaxations, but non-convex AC-OPF and small-to-large grid transfer remain open.

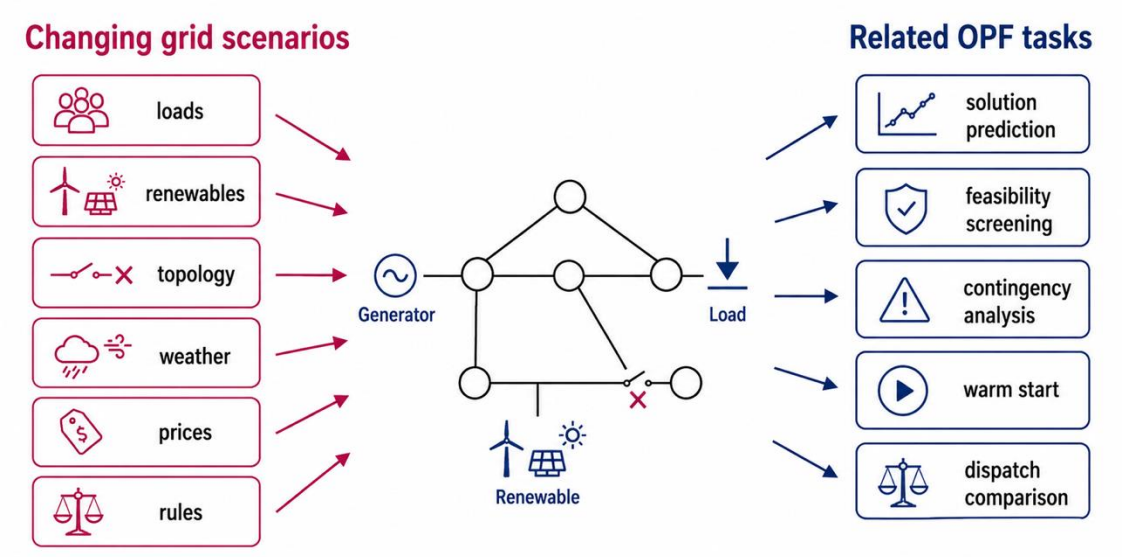
Part III: Foundation Models and LLM Workflows for OPF

From Reusable Grid Models to Automatic Workflows

From GNN to Grid Foundation Model

GNNs encode grid topology for one OPF task and can adapt to topology changes.

- But grid operation creates many related OPF tasks (*solution prediction, feasibility screening, contingency analysis, …*) under changing scenarios (*loads, renewables, topology, weather, prices, and sizes, …*)

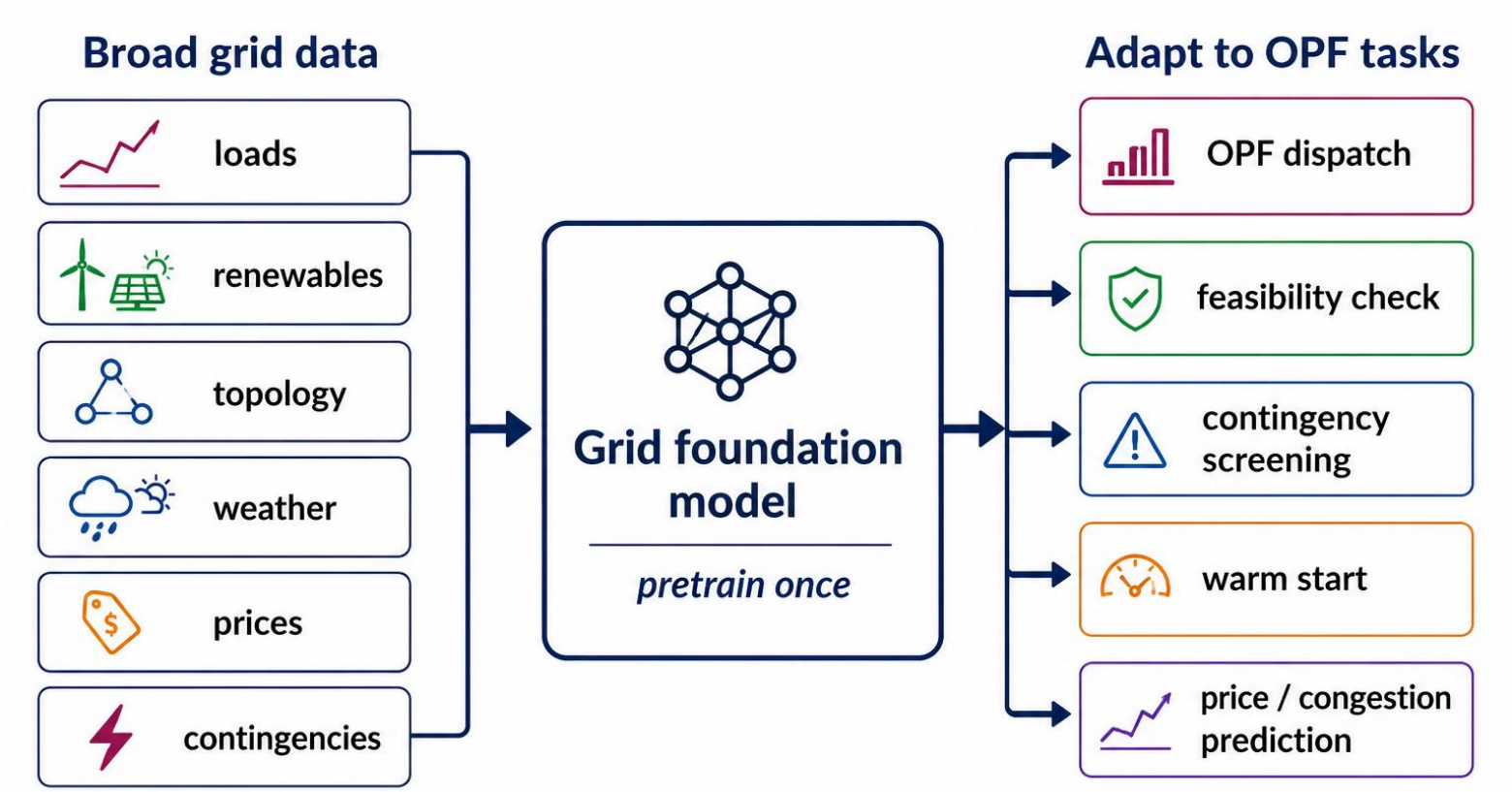


Can one model learn reusable grid representations across scenarios and adapt to many downstream OPF tasks? → ***Grid foundation model***

What is Foundation Models for Grid Operation?

Foundation model: a large model trained once on broad data, then adapted to many downstream tasks.

- Adaptation can use *fine-tuning*, *prompting*, *adapters*, ...



H. F. Hamann, T. Brunschwiler, B. Gjorgiev, L. S. A. Martins, A. Puech, A. Varbella, J. Weiss, J. Bernabe-Moreno, A. B. Massé, S. Choi, I. Foster, B.-M. Hodge, R. Jain, K. Kim, V. Mai, F. Mirallès, M. De Montigny, O. Ramos-Leaños, H. Suprême, L. Xie, E.-N. S. Youssef, A. Zinflou, A. J. Belyi, R. J. Bessa, B. P. Bhattarai, J. Schmude, and S. Sobolevsky, *Foundation Models for the Electric Power Grid*, Joule, 2024.

Grid Foundation Models
From Concept to OPF Applications

2024

CANOS

Task: AC-OPF under N-1 topology perturbations
Arch: Heterogeneous GNN
Train: supervised OPF labels + constraint loss
Scale: 10k buses; 0.3M scenarios

2026.05

LUMINA

Task: ACOPF surrogate transfer and benchmark
Arch: Homo./Heter. GNN backbones
Train: Multi-topology + MSE/AL/VBL
Scale: up to 2k buses

2026.05

HydraGNN OPF-GFM

Task: transferable OPF surrogate
Arch: HeteroSAGE / HeteroHEAT
Train: supervised MSE loss
Scale: ~1.7M params; up to 13k+ buses; 3M scenarios.

2024 - 2025

GridFM Benchmark

Task: reusable grid representation + down stream OPF tasks
Arch: TransformerConv / GPSConv encoder
Train: masked reconstruction + constraint loss
Scale: up to 2k buses

2026.05

GridSFM

Task: AC-OPF state, feasibility, warm start
Arch: DEC-aware heter. Graph Transformer
Train: Multi-topology + MSE + constraint loss
Scale: 15M params, up to 4k buses (Open version); 100M params, up to 13K+ buses (Premier version); >0.5M scenarios;

- GNN-OPF precursors provide the graph backbone for topology-aware OPF learning, but most are task-specific and limited in broad pretraining.
- Grid FMs extend this backbone with large-scale cross-scenario pretraining, reusable representations, and downstream adaptation.

L. Piloto, S. Liguori, S. Madjiheurem, et al., *CANOS: A Fast and Scalable Neural AC-OPF Solver Robust to N-1 Perturbations*, arXiv, 2024.
H. F. Hamann, T. Brunschwiler, B. Gjorgiev, et al., *Foundation Models for the Electric Power Grid*, Joule, 2024.
Y. Li, Z. Memon, H. Jin, S. Fenu, K. Song, S. B. Sharma, P. Gasana, H. Kim, L. Zhao, and K. Kim, *LUMINA: Foundation Models for Topology Transferable ACOPF,* arXiv, 2026.
W. Yang, A. B. M. Lima, T. V. Spina, S. Fowers, B. Zhang, and C. White, *GridSFM: A Foundation Model for AC Optimal Power Flow,* Microsoft Research, 2026.
M. Lupo Pasini, Y. Li, K. Kim, and T. Kuruganti, *Scalable Heterogeneous Graph Foundation Models for Data-Driven Optimal Power Flow in Smart Grids*, arXiv, 2026.

GridSFM: GNN Designs and Structures

Power grids as hetero. graphs

- Buses, generators, loads, shunts as typed nodes
- lines & transformers as typed edges

Learn a hierarchy of representations

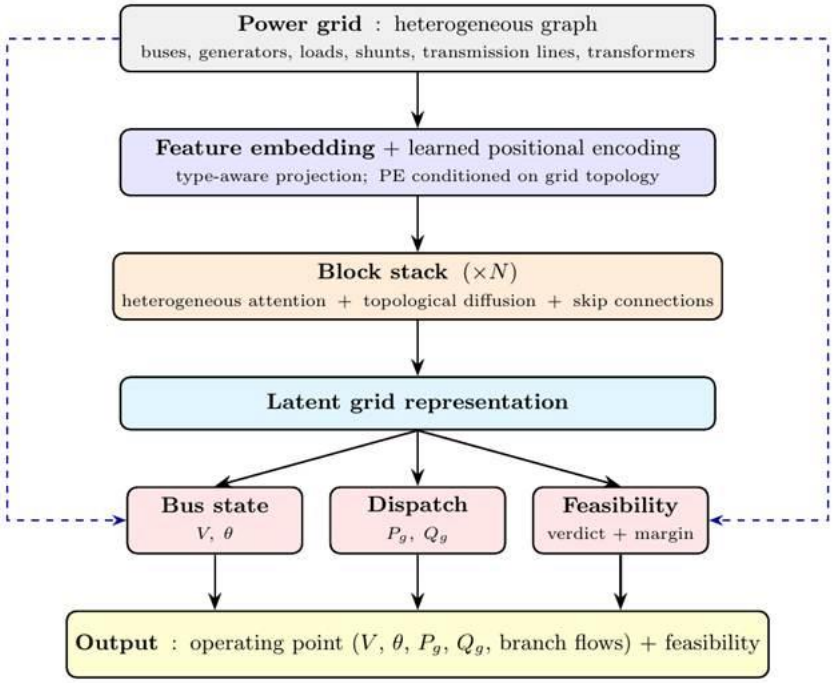
- Local bus-level physics → zonal patterns (positional feature) → system-wide context (via attention)

Scale-invariant design:

- relative features, node-budget batching
- Ratios in [0,1] regardless of grid size;
- constant GPU memory per batch

Training with MSE + Constraint penalty

- Auto-weighted multi-loss across supervised regression and physics-based objectives



One model trains on grids from 100 ~10,000 buses with diverse grid topologies simultaneously

Technical and experimental details are based on slides form Weiwei Yang | Microsoft Research | GridFM Workshop @ Harvard 2026

W. Yang, A. B. M. Lima, T. V. Spina, S. Fowers, B. Zhang, and C. White, *GridSFM: A Foundation Model for AC Optimal Power Flow*, Microsoft Research, 2026.

GridSFM: Prediction Accuracy

Dev:	4× A6000	Training:	8× B200	Distributed:	DDP	Pre-train:	~48 hrs

Single model evaluated across 5 pglib-opf topologies

Metric	case500	case2000	case6470	case10000	case13659
Voltage Error V%	9.04	8.42	8.21	7.47	4.13
Cost Gap (mean %)	1.71	1.95	1.82	0.49	0.82
P Dispatch (P%cap)	10.3	5.4	16.8	13.0	1.7
KCL P Balance (%)	0.33	0.22	0.06	0.17	0.03
Thermal Violation (max %)	0.0	0.0	29.5	0.0	0.0
Δθ Edge Error (mean °)	0.55	0.47	1.02	0.51	0.43
Feasibility Accuracy	100%	100%	95.9%	100%	100%

99.2% feasibility accuracy, <2% cost gap,
near-zero thermal violations*

Technical and experimental details are based on slides form Weiwei Yang | Microsoft Research | GridFM Workshop @ Harvard 2026

W. Yang, A. B. M. Lima, T. V. Spina, S. Fowers, B. Zhang, and C. White, *GridSFM: A Foundation Model for AC Optimal Power Flow*, Microsoft Research, 2026.

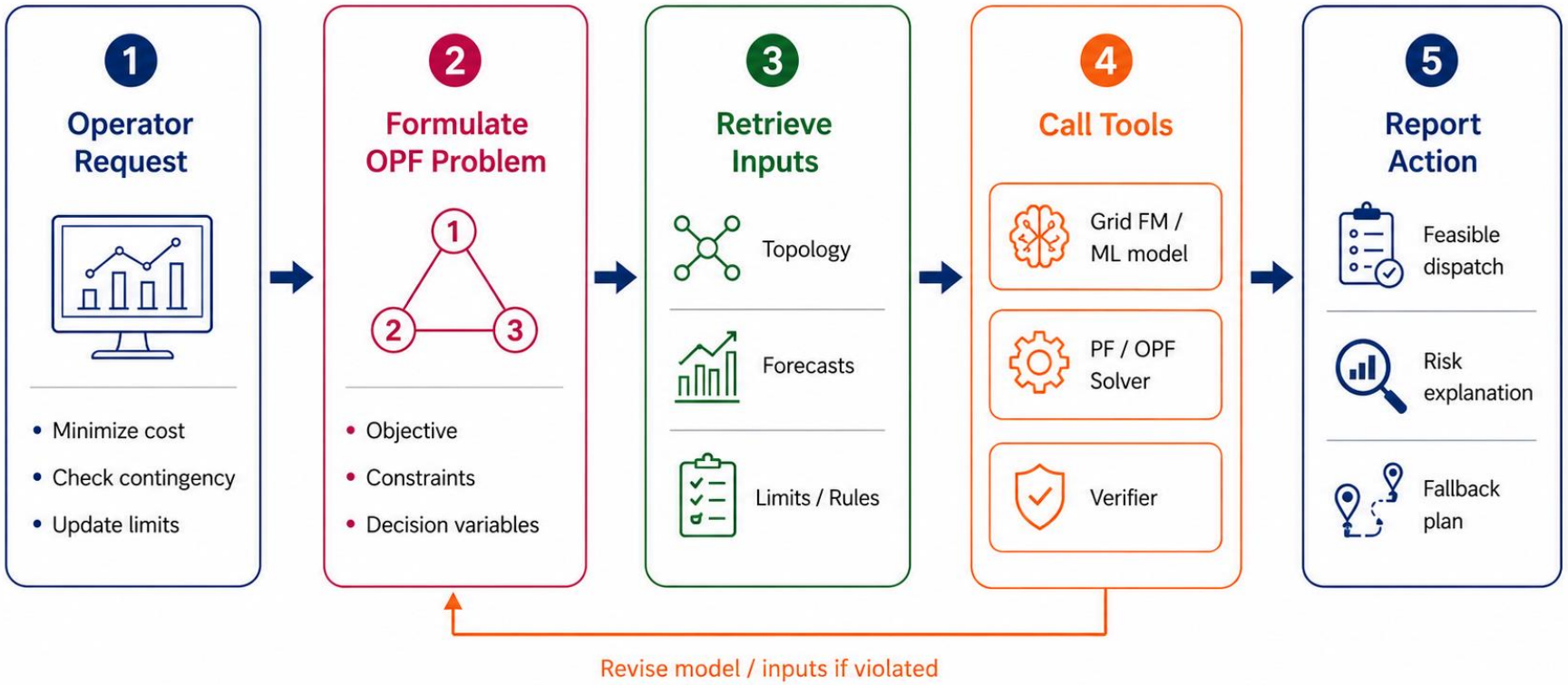
From Specialized Tasks to Complete Workflow

Grid FMs help with multiple tasks across scenarios:

- Predict grid states, screen feasibility, and warm-start solutions

But grid operation needs a complete workflow:

- Formulate problems, retrieve data, call tools, verify results, and report actions

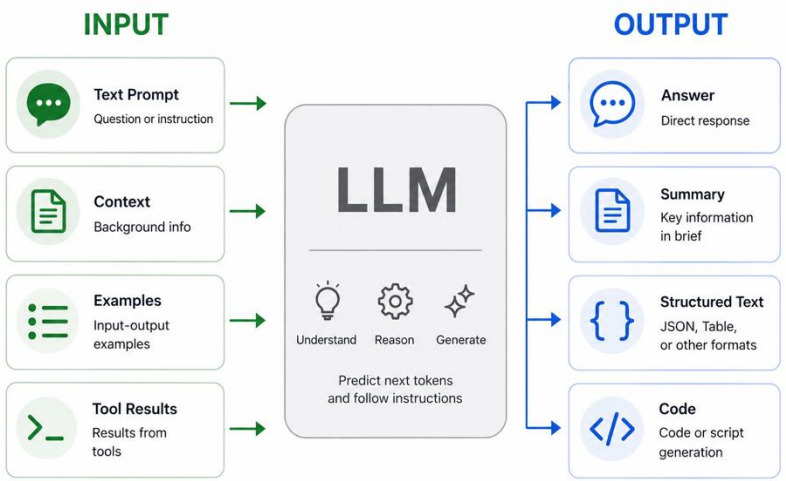


C. Huang, S. Li, R. Liu, H. Wang, and Y. Chen, *Large Foundation Models for Power Systems*, IEEE PESGM, 2024.
Q. Zhang and L. Xie, *PowerAgent: A Road Map Toward Agentic Intelligence in Power Systems*, IEEE Power Energy Mag., 2025.

Why LLMs Can Help Coordinate Workflows?

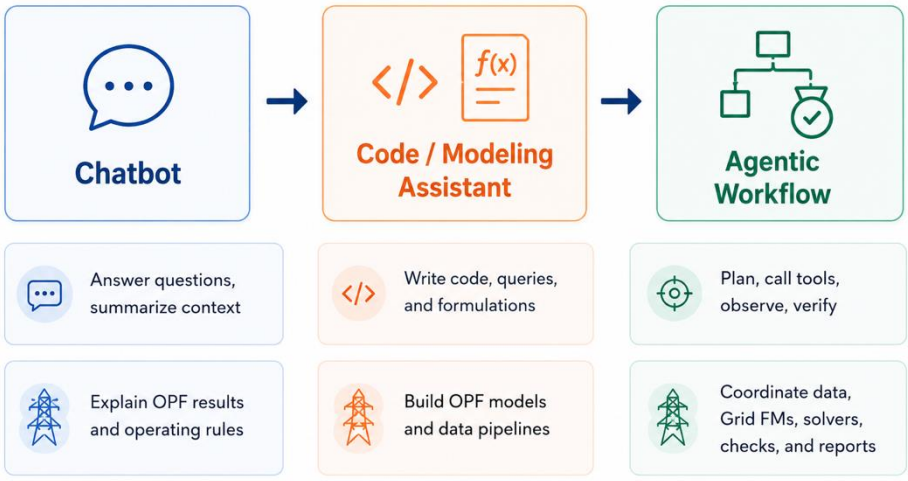
Large Language Model

LLMs map text and context into answers, structured outputs, code, and tool calls.



From Chatbot to Agent

Recent systems evolve from chatbots to agents that can plan, act, observe, and verify.



Given these capabilities, LLMs can enter OPF research at different levels: as a solver, a modeling assistant, or a workflow agent.

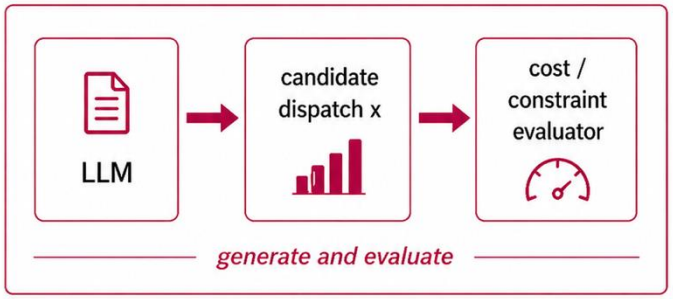
C. Huang, S. Li, R. Liu, H. Wang, and Y. Chen, *Large Foundation Models for Power Systems*, IEEE PESGM, 2024.
Q. Zhang and L. Xie, *PowerAgent: A Road Map Toward Agentic Intelligence in Power Systems*, IEEE Power Energy Mag., 2025.

LLM for OPF: Recent Applications

1. LLM as solution generator

Generate candidate dispatch and evaluate cost / violations.

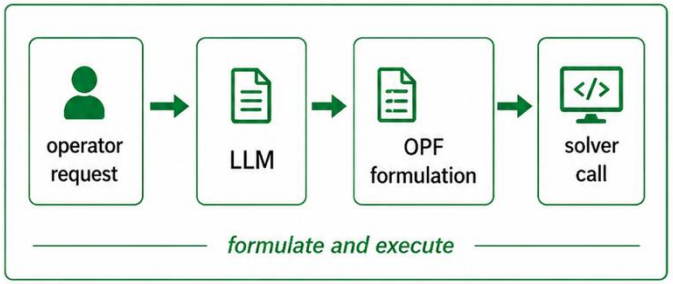
+ Flexible prompting and small-scale exploration
− Hard to scale, certify, and trust for full AC-OPF



2. LLM as modeling assistant

Translate operator requirements into OPF formulation, code, data queries, and solver calls.

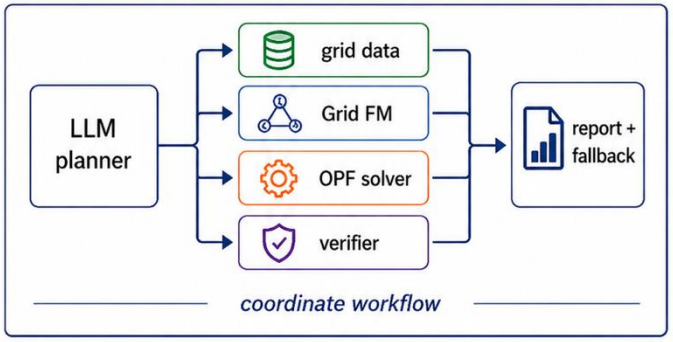
+ Reduces modeling and programming burden
− Needs validation, repair, and expert review



3. LLM as workflow agent

Plan workflow, call Grid FMs / PF / OPF solvers / verifiers, explain results, and trigger fallback.

+ Closest to real grid-operation workflow
− Requires reliable tool grounding and audit trails



C. Huang, S. Li, R. Liu, H. Wang, and Y. Chen, *Large Foundation Models for Power Systems*, IEEE PESGM, 2024.
Q. Zhang and L. Xie, *PowerAgent: A Road Map Toward Agentic Intelligence in Power Systems*, IEEE Power Energy Mag., 2025.
F. Bernier, J. Cao, M. Cordy, and S. Ghamizi, *PowerGraph-LLM: Novel Power Grid Graph Embedding and Optimization with Large Language Models*, IEEE TPWS., 2025.
X. Yang, C. Lin, Y. Yang, Q. Wang, H. Liu, H. Hua, and W. Wu, *LLM Powered Automated Modeling and Optimization of Active Distribution Network Dispatch Problems*, arXiv, 2025.
X. Chen, *X-GridAgent: An LLM-Powered Agentic AI System for Assisting Power Grid Analysis*, arXiv, 2025.
M. Shamseldein, *Grid-Mind: An LLM-Orchestrated Multi-Fidelity Agent for Automated Connection Impact Assessment*, arXiv, 2026.
B. Liu, J. Dong, and J. Lian, *Grid-Orch: An LLM-Powered Orchestrator for Distribution Grid Simulation and Analytics*, arXiv, 2026.

Note: Selected references are shown; many other LLM-OPF variants and applications are omitted for space.

LLM as Solution Generator for OPF

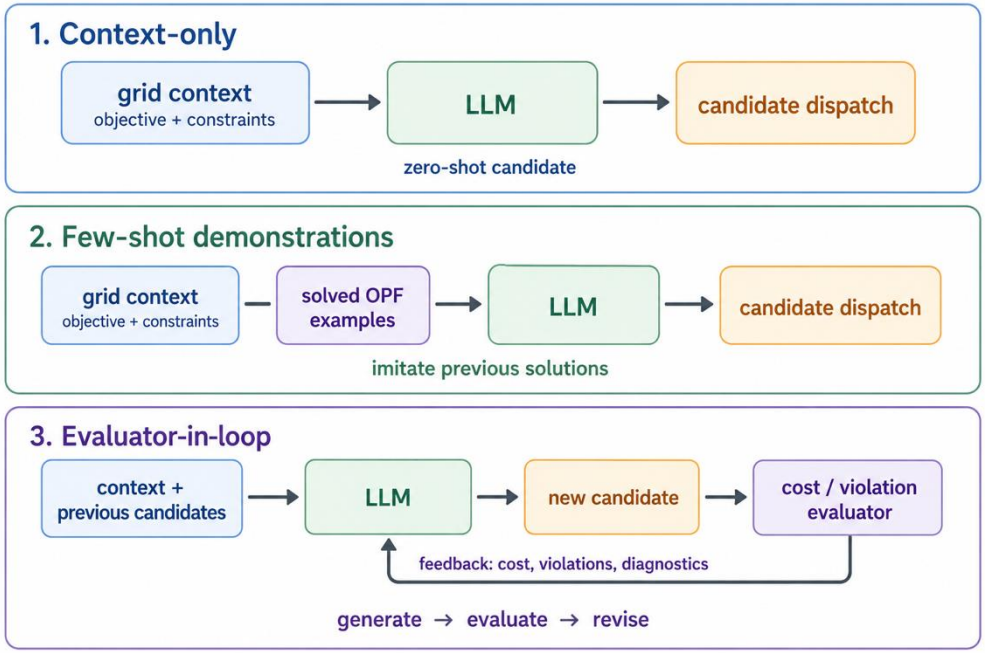
Exploratory use of LLMs for optimization

- Can an LLM use problem context to propose reasonable dispatch?
- Useful as a benchmark for LLM numerical and constraint reasoning

1. Context-only:
   → *prompt engineering*

2. Few-shot:
   → *in context learning*

3. Evaluator-in-loop:
   → *black-box/evolutionary opt.*



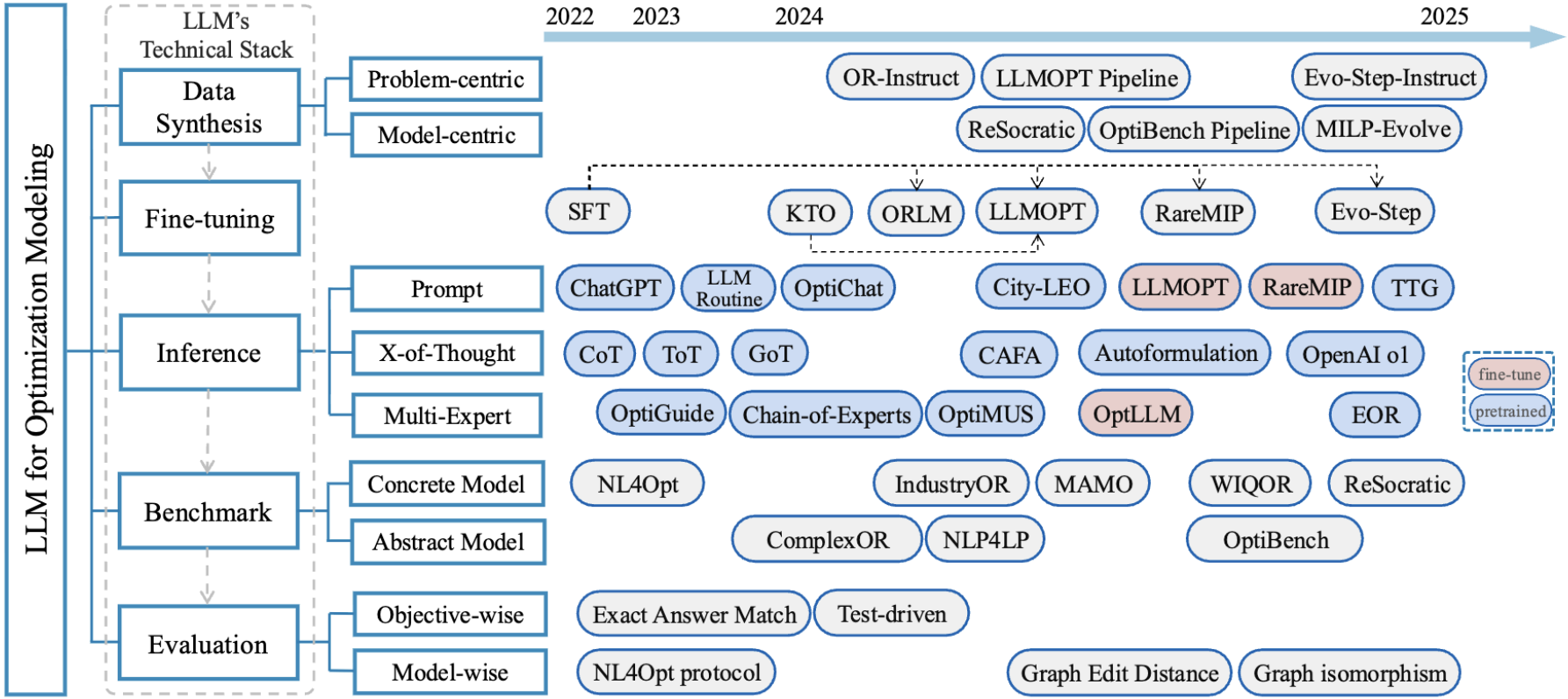
Current studies explore whether LLMs can act as optimizers; for OPF, the generated candidates should be evaluated, refined, or rejected.

C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, *Large Language Models as Optimizers*, ICLR, 2024.
C. Huang, S. Li, R. Liu, H. Wang, and Y. Chen, *Large Foundation Models for Power Systems*, IEEE PESGM, 2024.
F. Bernier, J. Cao, M. Cordy, and S. Ghamizi, *PowerGraph-LLM: Novel Power Grid Graph Embedding and Optimization with Large Language Models*, IEEE TPWS., 2025.
F. Bernier, S. Ghamizi, P. Dogoulis, and M. Cordy, *Can Large Language Models Reason and Optimize Under Constraints?*, arXiv, 2026.

LLM as Problem Modeling Assistant

Problem modeling is often the bottleneck

- Many optimization tasks start from natural-language requirements
- Manual model construction is slow and expertise-intensive
- natural language → optimization models is an active area in general **LLM-OR**

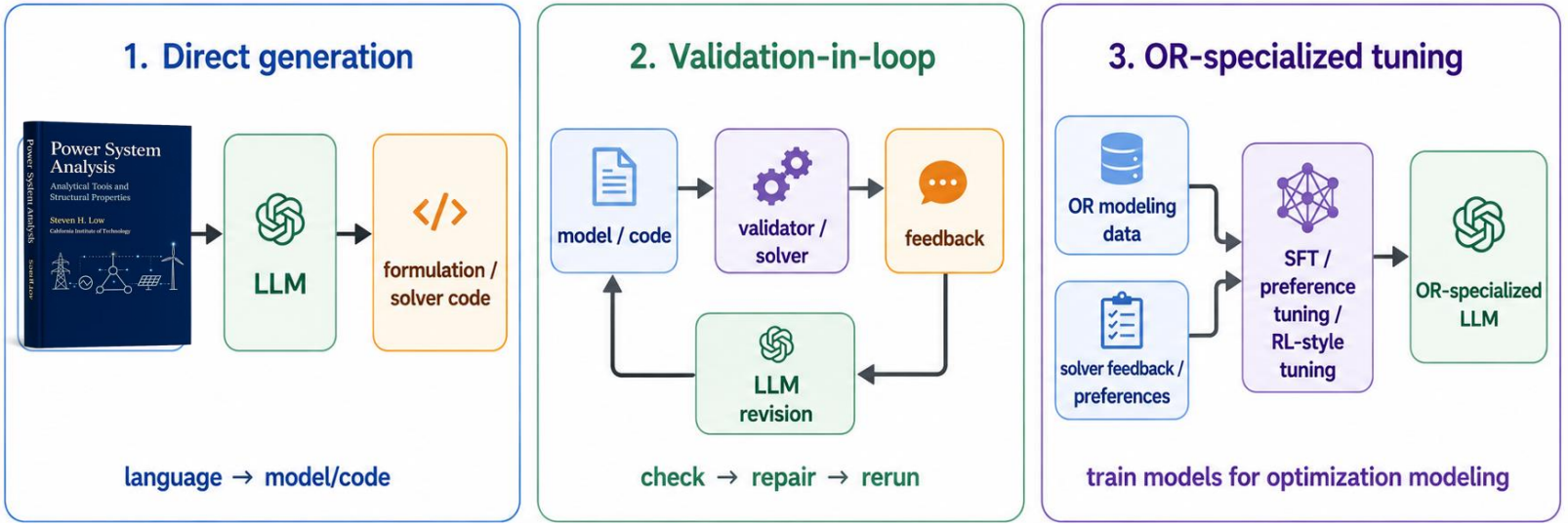


Z. Xiao, J. Xie, L. Xu, S. Guan, J. Zhu, X. Han, X. Fu, W. Yu, H. Wu, W. Shi, Q. Kang, J. Duan, T. Zhong, M. Yuan, J. Zeng, Y. Wang, G. Chen, and D. Zhang, *A Survey of Optimization Modeling Meets LLMs: Progress and Future Directions*, arXiv, 2025.

LLM as Problem Modeling Assistant

Problem modeling is often the bottleneck

- Many optimization tasks start from natural-language requirements
- Manual model construction is slow and expertise-intensive



- For OPF, modeling assistants must be grounded in **grid specific context**
- **Challenges**: semantic correctness and operational feasibility

LLMs are promising modeling assistants, but expert review remain essential.

Z. Xiao, J. Xie, L. Xu, S. Guan, J. Zhu, X. Han, X. Fu, W. Yu, H. Wu, W. Shi, Q. Kang, J. Duan, T. Zhong, M. Yuan, J. Zeng, Y. Wang, G. Chen, and D. Zhang, *A Survey of Optimization Modeling Meets LLMs: Progress and Future Directions*, arXiv, 2025.

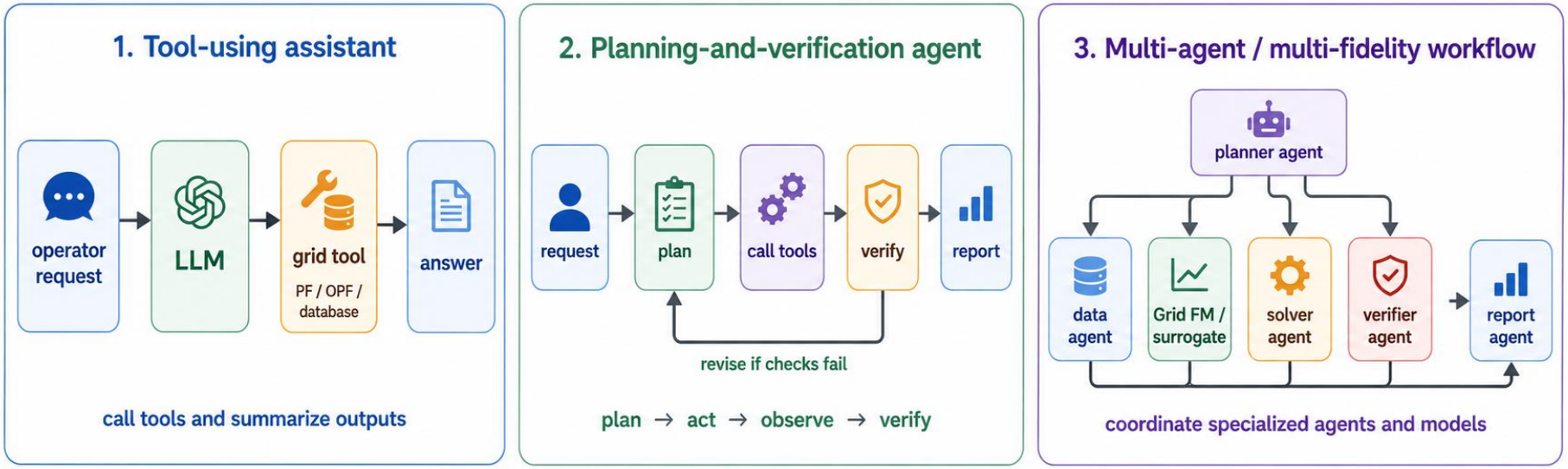
X. Yang, C. Lin, Y. Yang, Q. Wang, H. Liu, H. Hua, and W. Wu, *Large Language Model Powered Automated Modeling and Optimization of Active Distribution Network Dispatch Problems*, arXiv, 2025.

C. Shen, Z. Guo, X. Wan, Z. Yang, Y. Zhang, W. Huang, J. Song, Z. Zhang, and M. Sun, *ProOPF: Benchmarking and Improving LLMs for Professional-Grade Power Systems Optimization Modeling*, ICML, 2026.

LLM as Workflow Agent for Grid Operation

Grid operation is a multi-step workflow

- Including data retrieval, model formulation, tool calls, verification, and reporting
- LLM agents can help coordinate these steps through planning, tool use, and feedback



- **Challenges**: tool grounding, provenance, audit trail, and operational safety

LLM agents are most useful when grounded by grid data, trusted solvers, verifiers, and audit trails.

Q. Zhang and L. Xie, *PowerAgent: A Road Map Toward Agentic Intelligence in Power Systems*, IEEE Power Energy Mag., 2025.
H. Jin, K. Kim, and J. Kwon, *GridMind: LLMs-Powered Agents for Power System Analysis and Operations*, arXiv, 2025.
Y. Wen and X. Chen, *X-GridAgent: An LLM-Powered Agentic AI System for Assisting Power Grid Analysis*, arXiv, 2025.
M. Shamseldein, *Grid-Mind: An LLM-Orchestrated Multi-Fidelity Agent for Automated Connection Impact Assessment*, arXiv, 2026.

Takeaways: GridFM and LLM Agent for OPF

1. Grid foundation models

- Learn reusable grid representations across scenarios.
- Useful for state prediction, feasibility screening, warm start

2. LLMs for OPF

- Candidate generation needs external cost / constraint checks
- Modeling assistance needs validation and expert review

3. Workflow agents

- Closest to real grid-operation workflow
- Require tool grounding, provenance, audit trails, and human oversight

Practical OPF intelligence will likely combine fast grid FMs, trusted solvers, verifiers, and LLM-based workflow coordination under human oversight.

AI-OPF Code Tutorial

From Concepts to Codes Implementations

AI for Optimal Power Flow

This tutorial introduces **AI for AC Optimal Power Flow (AC-OPF)** using a small 5-bus power network and PyTorch-based neural networks. The goal is to learn the full workflow: formulate a basic AC-OPF problem, generate or load solver labels, train ML approximators, evaluate them with OPF-aware metrics, and understand the limits of using ML for grid operation.

Version released in 2023

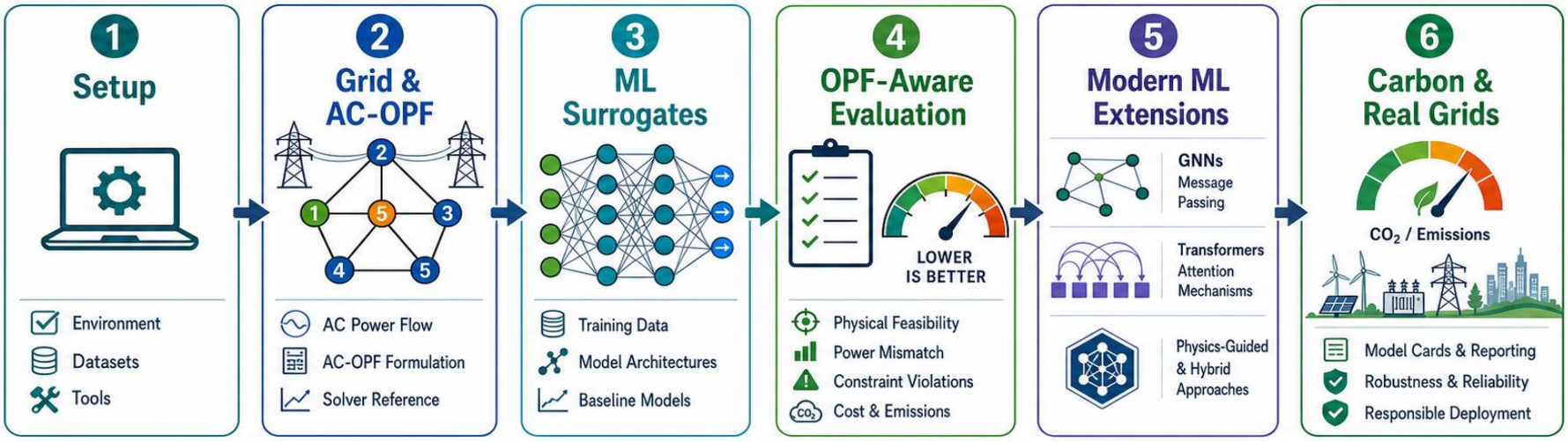
- Jorge Montalvo, CCAI, jorge@climatechange.ai
- Utkarsha Agwan, uagwan@berkeley.edu
- Panos Moutis, panay1ot1s@climatechange.ai

Version revised and released in 2024

- Enming Liang, enming.cityu@gmail.com
- Advising comments from Priya L. Donti (MIT) and Minghua Chen (CityU, HK)

Version revised and released in 2026

- Enming Liang, enming.cityu@gmail.com
- Advising comments from the CCAI Virtual Summer School Organizing Committee and Minghua Chen (CUHK-SZ)



Outline

- Grid operations and OPF formulations
- Relevant approaches and recent advances

- Machine learning (ML) for constrained optimization
- End-to-end ML for standard AC-OPF problems (SL, UL)
- Solving AC-OPF with multiple load-solution mappings
- Machine learning for solving 2-stage OPF problems

- Ensuring DNN feasibility for constrained optimization
- DNN/GNN for OPF problems over flexible topology
- Large language models for solving OPF problems

- Concluding remarks and open challenges

OPF is Critical for Power System Operation

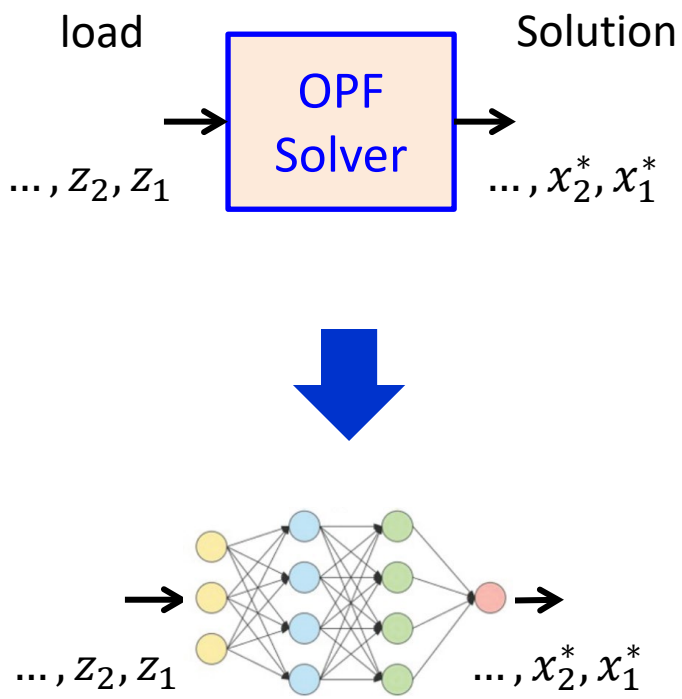
- OPF is to minimize the cost of serving load subject to physical and operational constraints
  - KCL and KVL physical constraints
  - Voltage, generation, and branch flow limits
  - Other operational constraints

- OPF underpins various important power system applications
  - Demand response
  - Economic dispatch
  - Unit commitment
  - Electricity market clearing
  - Security and reliability assessment

Solving OPF is Important but Challenging

- Penetration of renewable and grid modernization require solving OPF fast
  - Early termination of algorithms gives suboptimal solution
  - 5% saving amounts to 36 billion USD/year globally
- AC-OPF problem is non-convex and NP-hard
  - Practical OPF can involve more than 1M variables
  - ARPA-E GO: 30,000 buses, 3,526 generators, 32,020 lines
- General Newton-like iterative algorithms — Not scalable
- Linearization to solve OPF approximately — Inaccurate
- Convexification to solve OPF optimality — Only applicable to special case

ML for Solving OPF is Working



- <0.2% optimality loss AC-OPF simulations over IEEE cases, real-world topology, and loads by various NN schemes
  - Come with theoretical justification
  - 15,000x speedup over a 2000-bus network; 10% load variation
  - Evaluated over actual RTE networks with 9,241 buses, actual Korea-4492 system
  - Can also be used to generate a preliminary evaluation quickly
- Generalizable approaches
  - PR2 to ensure equality feasibility
  - Unsupervised learning to reduce training data complexity
  - Decomposition to improve scalability
  - Permutation-invariant NN design for 2-stage stochastic optimization problems

Summary of Challenges and Approaches

- NN solutions may be infeasible
  - Predict-and-reconstruct (PR2) / equation completion
  - Primal-dual training (utilizing KKT conditions)
  - Differentiable NN layers
- High training/data complexity for large-scale OPF problems
  - Un-/self- supervised learning, GNN, grid decomposition, compact learning, approximate data labels
- The multi-valued mapping issue for non-convex (AC-OPF) problems
  - Augmented learning, data preparation/selection
  - Generative learning (learning the input-dependent solution distributions)
- 2-stage stochastic ACOPF problems incurs dimensionality explosion (NN input dimension linear in K)
  - Permutation-invariant design to address dimensionality explosion with desirable theoretical guarantee
  - Input dimension independent of K (#scenarios), laten dimension $O(\log K)$

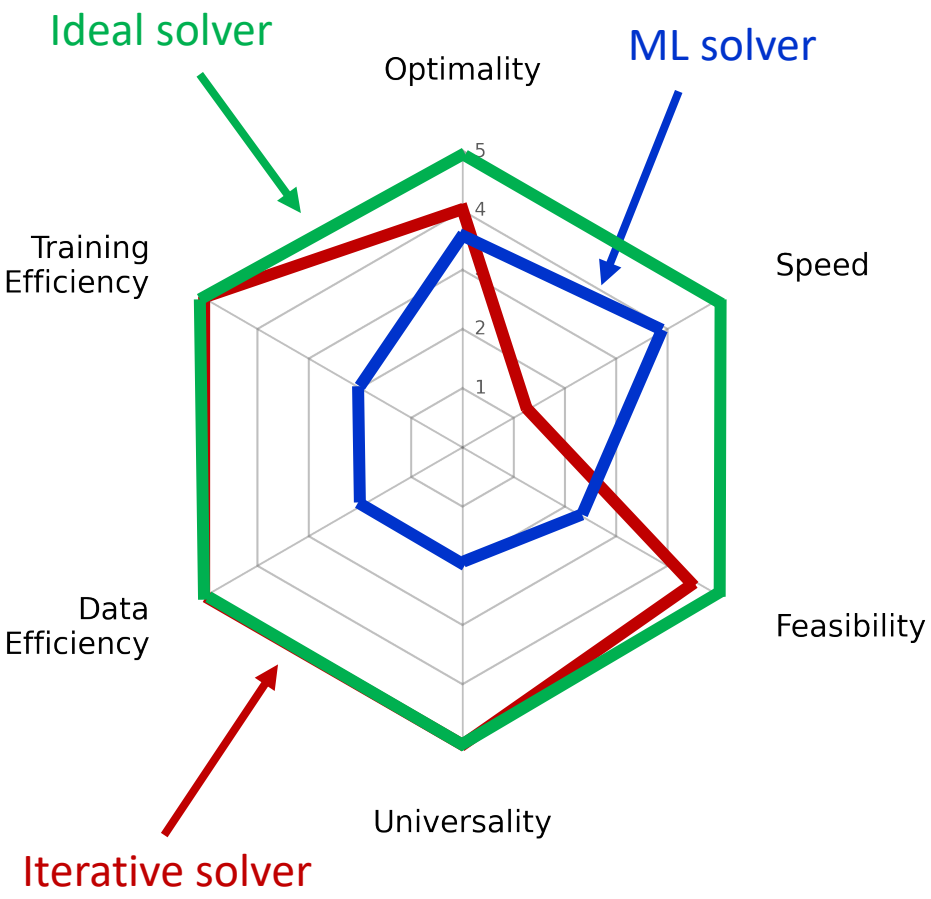
Some Open Issues

- Feasibility guarantees without killing speedup
- Generalization across topology, admittance, outages, and grid size
- Multi-valued AC-OPF mappings (become important for discrete optimization like unit commitment
- Data generation cost and label quality
- Scaling to very large grids
- Security-constrained and stochastic OPF, real-time OPF
- Human/operator trust (even a fast model with 99.9% feasibility may be unacceptable if the failure mode is opaque)
- The power of Agentic AI in identifying new structures and solutions
- Wiki: https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki

A Bigger Picture: ML is Amortized Solver for Constrained Optimization

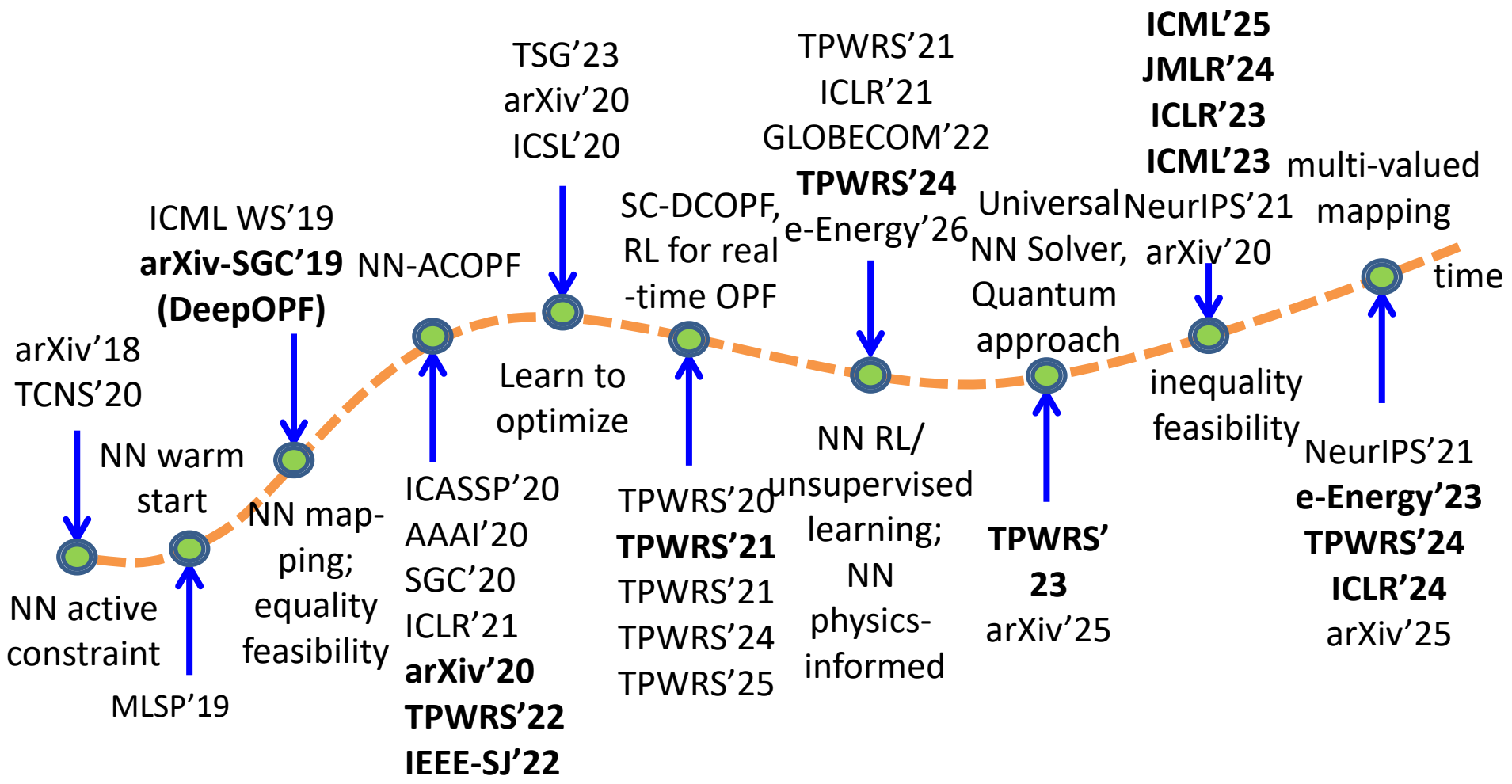
- ML for OPF sets a benchmark and is a frontier playground for ML for constrained optimization
  - Physics-respect hard constraints are not optional
  - Solutions must be near-optimal, feasible, and robust at constraint boundaries
  - Architecture, data, and loss design determine whether speed becomes useful
  - Deployment needs verification, fallback, auditability, and human trust

Six Dimensions for Comparing Solvers



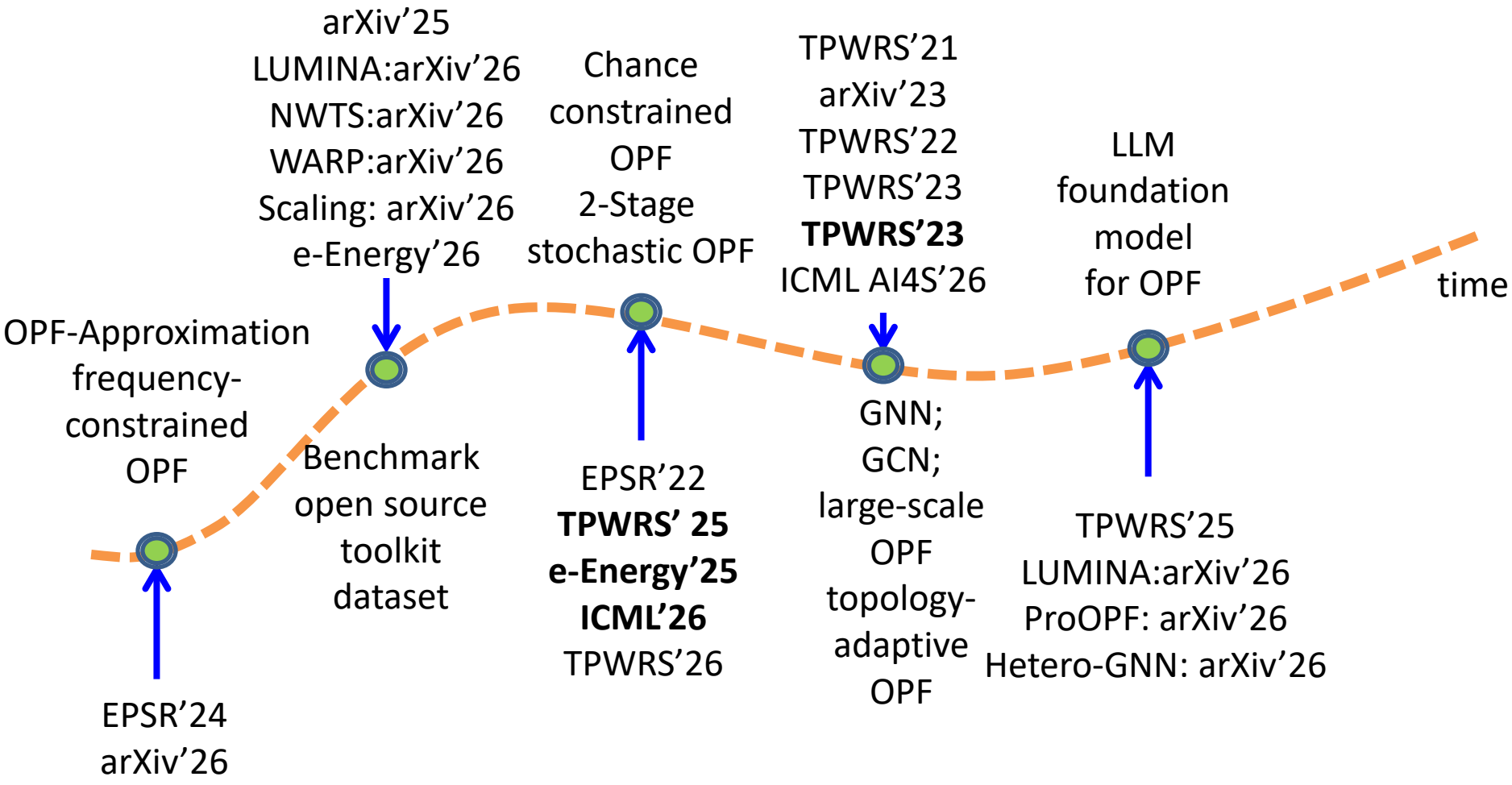
- **Feasibility**: solution feasibility
- **Optimality**: solution optimality
- **Speed**: run-time solution speed
- **Universality**: one solver for all problems
- **Data efficiency**: minimum data-preparation effort
- **Training efficiency**: minimum training effort

The Path Behind was Refreshing



- Our works in bold font
- [Wiki: https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki](https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki)

The Path Ahead is Still Unfolding

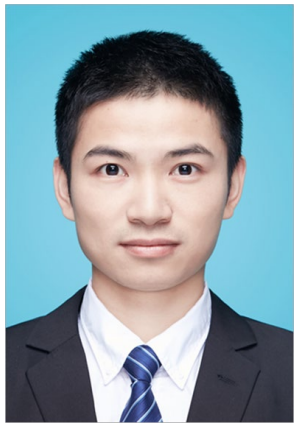


- Our works in bold font
- Wiki: https://energy.hosting.acm.org/wiki/index.php/ML_OPF_wiki

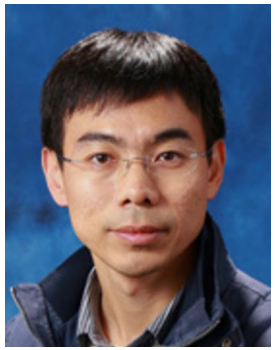
Acknowledgements

Xiang Pan
(CUHK; Tencent)

Tianyu Zhao
(CUHK; Lenovo Research)



Min Zhou
(CityU)



Shengyu Zhang
(Tencent)

Wanjun Huang
(CityU; Beihang Univ)

Thank you!

Enming Liang, Minghua Chen, and Steven Low

香港城市大學 City University of Hong Kong
香港中文大學(深圳) The Chinese University of Hong Kong, Shenzhen
Caltech