AI-guided Surrogate Modeling for Real-world Optimization

Yuandong Tian
Research Scientist and Manager

Meta AI (FAIR)
Reinforcement Learning

Go

Chess

Shogi

Poker

Big Success in Games

DoTA 2

StarCraft II
ELF OpenGo

Vs top professional players

<table>
<thead>
<tr>
<th>Name</th>
<th>ELO (world rank)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim Ji-seok</td>
<td>3590 (#3)</td>
<td>5-0</td>
</tr>
<tr>
<td>Shin Jin-seo</td>
<td>3570 (#5)</td>
<td>5-0</td>
</tr>
<tr>
<td>Park Yeonghun</td>
<td>3481 (#23)</td>
<td>5-0</td>
</tr>
<tr>
<td>Choi Cheolhan</td>
<td>3466 (#30)</td>
<td>5-0</td>
</tr>
</tbody>
</table>

Single GPU, 80k rollouts, 50 seconds
Offer unlimited thinking time for the players

[ELF OpenGo: An Analysis and Open Reimplementation of AlphaZero, Y. Tian et al, ICML 2019]
What’s Beyond Games?
Chip Design (Google)

Several weeks with **human experts** in the loop

→

**Fully automatic design** in 6 hours

Optimization Problems

Travel Salesman Problem

Job Scheduling

Vehicle Routing

Bin Packing

Protein Folding

Model-Search

\[ x^* = \arg \max_{x \in \Omega} f(x) \]
Wait...What?

• Many problems are NP-hard problems.
  • No good algorithm unless $P = NP$

• These guarantees are worst-case ones.
  • To prove a lower-bound, construct an adversarial example to fail the algorithm

• For specific distribution, there might be better heuristics.
  • Human heuristics are good but may not be suitable for everything
Efficient Search for Games

Go

Chess

Human Knowledge
Machine learned models
Efficient Search for Optimization

Exhaustive search to get a good solution

Human Knowledge
More Efficient Search for Optimization

Can we use Machine Learning?

Exhaustive search to get a good solution
Components of Search

- Search Heuristics
- State Representation
- Design of State/Action Space
- Design Surrogate Models
Components of Search

Search Heuristics

Initial solution $\rightarrow$ Improved solution 1 $\rightarrow$

Improved solution 2 $\rightarrow$ ...

[X. Chen and Y. Tian, Learning to Perform Local Rewriting for Combinatorial Optimization, NeurIPS'19]
[H. Shi et al, Deep Symbolic Superoptimization Without Human Knowledge, ICLR’20]
[H. Zhu et al, Network planning with deep reinforcement learning, SIGCOMM’21]
[T. Huang et al, Local Branching Relaxation Heuristics for Integer Linear Programs, CPAIOR’23]
[T. Huang et al, Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning, arXiv’23]
Components of Search

Design of State/Action Space

If useful actions only happen after 50 binary moves, then we will waste our efforts in this $2^{50}$ possibilities.

[L. Wang, et al, Sample-Efficient Neural Architecture Search by Learning Action Space, T-PAMI’21]
[Y. Zhao, et al, Multi-objective Optimization by Learning Space Partitions, ICLR’22]
Components of Search

- **State Representation**

- **GOAL**: Letting in as much sunlight as possible

[**X. Chen et al,** *Latent Execution for Neural Program Synthesis Beyond Domain-Specific Languages,* NeurIPS’21]

[**T. Wang et al,** *Denoised MDPs: Learning World Models Better Than the World Itself,* ICML’22]

[**Z. Jiang et al,** *Efficient Planning in a Compact Latent Action Space,* ICLR’23]
Components of Search

Design Surrogate Models

[Y. Zhao, et al, Few-shot neural architecture search, ICML’21]
**SurCo: Learning Linear Surrogates for Combinatorial Nonlinear Optimization**

Aaron Ferber¹, Taoan Huang¹, Daochen Zha², Martin Schubert³, Benoit Steiner⁴, Bistra Dilkina¹, Yuandong Tian⁴

¹University of Southern California, ²Rice University, ³Reality Lab Display, ⁴Meta AI (FAIR)

Optimizing Nonlinear Functions over Combinatorial Regions

- Nonlinear + differentiable objective
- Combinatorial feasible region
- Real-world domains:
  - Computer system planning
  - Designing photonic devices
  - Throughput optimization
  - Antenna design
  - Energy grid
Example: Embedding Table Placement

Given:
- $k$ tables
- $n$ identical devices
- Table $i$ has memory requirement $m_i$
- Device $j$ has memory capacity $M_j$

Find
- Allocation of tables to devices observing device memory limits
- Minimize latency which is estimated by a neural network (capturing nonlinear interactions)
Example: Embedding Table Placement

Given:
- \( k \) tables
- \( n \) identical devices
- Table \( i \) has memory requirement \( m_i \)
- Device \( j \) has memory capacity \( M_j \)

Formulation

\[
\min_x L(\{x_{ij}\}) \quad \text{s.t.} \quad \sum_i x_{ij} m_i \leq M_j, \quad \sum_j x_{ij} = 1, \quad x_{ij} \in \{0,1\}
\]

\( L \) is nonlinear due to system issues (e.g., batching, communication, etc)
Nonlinear Optimization is Hard

• Specific domains have specialized solvers

• General solvers are often slow (without very careful modeling)

• Genetic algorithms or gradient-based methods may not find feasible solutions
Linear Optimization is *Easy*(ish)

- MILP solvers (CPLEX, Gurobi, SCIP) easily handle industry-scale problems
- Plus other solvers for linear settings
  - Greedy
  - LP + total unimodularity
Idea: Find a Linear Surrogate

- Learn a MILP objective whose optimal solution $x^*$ solves the nonlinear problem

\[
\begin{align*}
\min_{x} & \quad f(x; y) \\
\text{s.t} & \quad x \in \Omega = \text{combinatorial constraints}
\end{align*}
\]

Originally

Nonlinear optimization with combinatorial constraints

Now

Surrogate optimization

\[
\begin{align*}
x^*(y) &= \arg\min_{x} \ c(y)^T x \\
\text{s.t} & \quad x \in \Omega
\end{align*}
\]

Predict surrogate cost $c = c(y)$

$x^*(y)$ optimizes $f(x; y)$ as much as possible
Idea: Find a Linear Surrogate

• Learn a MILP objective whose optimal solution $x^*$ solves the nonlinear problem

\[
\min_x f(x; y) \\
\text{s.t. } x \in \Omega
\]

Nonlinear optimization with combinatorial constraints

Originally

\[
x^*(y) \text{ optimizes } f(x; y) \text{ as much as possible}
\]

Surrogate optimization

\[
x^*(y) = \arg\min_x c(y)^T x \\
\text{s.t. } x \in \Omega
\]

\[
\text{solved by existing combinatorial solvers}
\]

Challenge: how to find the right objective?
Idea: Find a Linear Surrogate

- Learn a MILP objective whose optimal solution $x^*$ solves the nonlinear problem

Proposal: gradient-based optimization
Proposal: surrogate learning

- Use surrogate MILP to solve original problem
- Find linear coefficients $c$ such that $\arg\min_{x \in \Omega} f(x) \approx \arg\min_{x \in \Omega} c^T x$
SurCo-zero: gradient-based optimization

- **Iterative** solver based on linear surrogate guided by **gradient updates**
- Update linear coefficients $c$ such that $x^*(c)$ improves objective $f(x^*(c))$
SurCo-prior: distributional learning

- One pass solver based on model **learned offline**
- Use neural model based on **problem features** to predict linear coefficients

**Diagram**

- **Problem features** $y$
- **Surrogate Coefficients** $c$
- **Solution** $x^*(c)$
- **Loss** $f(x^*)$

**Flowchart**

- Neural Network $c = NN(y; \theta)$
- Solver $x^*(c) = \arg\min_{x \in \Omega} c^T x$
- Objective $f(x^*)$

**Calculations**

- $\nabla_y NN(y; \theta)$
- $\nabla_c x^*(c)$
- $\nabla_x f(x)$

**Differentiation**

- **Standard NN autograd**
- **Recent work on differentiable optimization**
- **Assumed differentiable**

**Related work**

- Pytorch, Tensorflow, JAX etc.
- Differentiation of blackbox optimizers
- CVXPYLayers
- MIpaaL
- ... more in related work
SurCo-prior: distributional learning

- Update neural network parameters from training dataset

\[ c_i = NN(y_i; \theta) \]

Surrogate Coefficients
\[ c_{test} = NN(y_{test}; \theta) \]

Solver
\[ x^*(c) = \arg\min_{x \in \Omega} c^T x \]
SurCo-hybrid: fine-tuning from trained model

Update neural network parameters from training dataset

\[ c_i = NN(y_i; \theta) \]

Train Model parameters \( \theta \)

Initial Surrogate Coefficients

\[ c_0 = NN(y_{\text{test}}; \theta) \]

Fine-tune surrogate on-the-fly

Solver

\[ x^*(c) = \arg\min_{x \in \Omega} c^T x \]

Objective

\[ f(x^*) \]

Loss \( f(x^*) \)
SurCo-zero

No offline training data, just solve a single problem instance on-the-fly
SurCo-prior

$c_i = NN(y_i; \theta)$

Train Model parameters $\theta$

Surrogate Coefficients $c_{\text{test}} = NN(y_{\text{test}}; \theta)$

Solution $x^*(c_{\text{test}})$

$x^*(c) = \arg\min_{x \in \Omega} c^T x$

Uses offline training data to quickly solve problems at test time with just one solver call
SurCo-hybrid

\[ c_i = NN(y_i; \theta) \]

Initial Surrogate Coefficients
\[ c_0 = NN(y_{test}; \theta) \]

Train Model parameters \( \theta \)

Offline train + on-the-fly fine-tuning the surrogate

Objective \( f(x^*) \)

Solver
\[ x^*(c) = \arg\min_{x \in \Omega} c^T x \]

Solution
\[ x^*(c) \]

Loss
\[ f(x^*) \]
Related Work

**Differentiable optimization: backprop through solvers**

- **Amos et al.** OptNet: Differentiable optimization as a layer in neural networks. ICML 2017
- **Agrawal et al.** Differentiable Convex Optimization Layers. NeurIPS 2019
- **Berthet et al.** Learning with Differentiable Perturbed Optimizers. NeurIPS 2020
- **Demirović et al.** Predict+Optimise with Ranking Objectives: Exhaustively Learning Linear Functions. IJCAI 2019
- **Demirović et al.** Dynamic Programming for Predict + Optimise. AAAI 2020
- **Djolonga et al.** Differentiable Learning of Submodular Models. NeurIPS 2017
- **Donti et al.** Task-Based End-to-End Model Learning in Stochastic Optimization. NeurIPS 2017
- **Elmachtoub et al.** Smart “Predict, then Optimize”. Management Science 2022
- **Ferber et al.** MIPaaL: Mixed Integer Program as a Layer. AAAI 2020
- **Lee et al.** Meta-Learning with Differentiable Convex Optimization. CVPR 2019
- **Mandi et al.** Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems. AAAI 2020
- **Niepert et al.** Implicit MLE: Backpropagating Through Discrete Exponential Family Distributions. NeurIPS 2021
- **Valstelica et al.** Differentiation of Blackbox Combinatorial Solvers. ICLR 2019
- **Rolnýnek et al.** Optimizing Rank-Based Metrics with Blackbox Differentiation. CVPR 2020
- **Wang et al.** Automatically Learning Compact Quality-Aware Surrogates for Optimization Problems. NeurIPS 2020
- **Wang et al.** SATNet: Bridging Deep Learning and Logical Reasoning Using a Differentiable Satisfiability Solver. ICML 2019
- **Wilder et al.** Melding the Data-Decisions Pipeline: Decision-focused Learning for Combinatorial Optimization. AAAI 2019
- **Wilder et al.** End to End Learning and Optimization on Graphs. NeurIPS 2019
How SurCo is different from Predict+Optimise?

---

**Predict+Optimize**

- Suitable for linear optimization problems.
- Requires a ground truth linear coefficients $c_i$ of the objective.

**SurCo**

- Suitable for **nonlinear** objective $f(x; y)$.
- Unlike existing nonlinear solvers, **NO** analytic form needed.
- Does **NOT** require a ground truth linear coefficients $c_i$. Learned surrogate coefficients by itself.

---

Both requires contextual information (i.e., problem description $y_i$)
Related Work

**Mixed Integer Nonlinear Optimization: general-purpose solvers**

- **Belotti et al.** Mixed Integer Nonlinear Optimization. Acta Numerica 2013

**General-purpose heuristic optimizers: combinatorial constraints are hard**

- **Gad et al.** Pygad: An Intuitive Genetic Algorithm Python Library. 2021
- **Rapin et al.** Nevergrad – A Gradient-Free Optimization Platform. 2018
- **Wang et al.** Learning Search Space Partition for Black-Box Optimization Using Monte Carlo Tree Search. NeurIPS 2020
- **Wang et al.** Sample Efficient Neural Architecture Search by Learning Actions for Monte Carlo Tree Search. PAMI 2021

**RL for combinatorial optimization: combinatorial constraints are hard**

- **Kool et al.** Attention, Learn to Solve Routing Problems! ICLR 2018
- **Mazyavkina et al.** Reinforcement Learning for Combinatorial Optimization: A Survey. COR 2021
- **Nazari et al.** Reinforcement Learning for Solving the Vehicle Routing Problem. NeurIPS 2018
- **Zhang et al.** A Reinforcement Learning Approach to Job-Shop Scheduling. IJCAI 1995
Embedding Table Sharding

Used in large-scale deep learning systems: recommendation systems, knowledge graph

Place N “tables” (with known memory need $m_i$) on K devices ($x_{ij} = 1$: table $i$ assigned to device $j$)

$$\text{Min}_x L\left(\{x_{ij}\}\right) \quad \text{s.t.} \quad \sum_i x_{ij} m_i \leq M_j, \quad \sum_j x_{ij} = 1, \quad x_{ii} \in \{0,1\}$$

$L$ : Runtime bottleneck $f(x)$ estimated by NN (longest-running device)

$L$ is nonlinear due to system issues (e.g., batching, communication, etc.)

c($y; \theta$) gives surrogate ”per-table cost” $c_{ij}$

(and $\sum_{ij} c_{ij} x_{ij}$ is the surrogate latency objective)
Embedding Table Sharding

- Public Deep Learning Recommendation Model (DLRM dataset) placing between 10 to 60 tables on 4 GPUs

- Baseline: Greedy
- SoTA: RL approach Dreamshard\(^1\)
- SurCo: Surrogate NN model learned via CVXPYLayers (differentiable LP Solver)

\(^1\) Zha et al. NeurIPS 2022

Dataset: [https://github.com/facebookresearch/dlrm_datasets](https://github.com/facebookresearch/dlrm_datasets)
Results – Table Sharding

Table Sharding Solution Loss (Latency)

Table Sharding Deployment Runtime (s)
Inverse Photonic Design

• Design physically-viable devices that take light waves and routes different wavelengths to correct locations

\[ \mathcal{L}(S) = \left( \left\| \text{softplus}\left( g \frac{|S|^2 - |S_{\text{cutoff}}|^2}{\min(w_{\text{valid}})} \right) \right\|_2 \right)^2 \]

• Device design misspecification loss \( f(x) \) computed by differentiable electromagnetic simulator

• Feasible solution: the design must be the union of brush pattern
  • \( x = \text{binary\_opening}(x, \text{brush}) \)
  • \( x = \sim\text{binary\_opening}(\sim x, \text{brush}) \)
Inverse Photonic Design

- Dataset: Ceviche Challenges\(^1\)
- Most baselines don’t work here due to combinatorial constraints
- SoTA: Brush-based algorithm \(^1\)

- SurCo: Surrogate learned via blackbox differentiation\(^2\) of brush solver

\(^1\)Schubert et al. ACS Photonics 2022
\(^2\)Vlastelica et al. ICLR 2019
Dataset: [https://github.com/google/ceviche-challenges](https://github.com/google/ceviche-challenges)
Results – Inverse Photonics

Inverse Photonics Solution Loss (% Invalid)

- Solution Loss (% Invalid)
- Solution Loss (% Invalid)
- Solution Loss (% Invalid)
- Solution Loss (% Invalid)

Inverse Photonics Deployment Runtime (s)

- Deployment Runtime (s)
- Deployment Runtime (s)
- Deployment Runtime (s)
- Deployment Runtime (s)
Inverse photonics Convergence comparison + Solution example

Takeaways:
- SurCo-Zero finds loss-0 solutions quickly
- SurCo-Hybrid uses offline training data to get a head start

Inverse Photonics Loss Convergence

Method
- Pass-Through
- SurCo-zero
- SurCo-hybrid

Device Design

$E_2$ magnitude first wavelength

$E_2$ magnitude second wavelength

Wavelength division multiplexer
Conclusion

• Handle industrial applications with differentiable optimization

• High-quality solutions to combinatorial nonlinear optimization by finding linear surrogates
  • Sometimes we can find “easier” surrogate problems that solve much more difficult instances

• SurCo works in several data settings
  • Zero-shot vs Offline training
  • One step inference vs fine-tuning
Sample-efficient Surrogate Model for Frequency Response of Linear PDEs using Self-Attentive Complex Polynomials

Andrew Cohen¹*, Weiping Dou², Jiang Zhu², Slawomir Koziel³, Peter Renner², Jan-Ove Mattson², Xiaomeng Yang¹, Beidi Chen¹,⁴, Kevin Stone¹, Yuandong Tian¹*

¹Meta AI (FAIR), ²Reality Lab Antenna (Meta), ³Reykjavik University, ⁴Carnegie Mellon University
* = Equal technical contribution

https://arxiv.org/abs/2301.02747
Background

Solving the linear PDE

\[ \frac{\partial^n \psi}{\partial t^n} = F(\psi, \nabla_x \psi, \ldots; h) \]

\( \psi(x, t) \) is the spatial-temporal signal under time evolutions. 
\( F \) is a linear function with respect to \( \psi \) and its derivatives. 
\( h \) is design choice.
Examples

\[
\frac{\partial \psi}{\partial t} = \nabla^2 \psi
\]

Heat equation

\[
\frac{\partial^2 \psi}{\partial t^2} = c^2 \nabla^2 \psi
\]

Wave equation

\[
i\hbar \frac{\partial \psi}{\partial t} = \left[ -\frac{\hbar^2}{2m} \nabla^2 + V \right] \psi
\]

Schrodinger’s Equation

\[
\nabla \cdot E = \frac{\rho}{\epsilon_0}, \nabla \cdot B = 0 \\
\n\nabla \times E = -\frac{\partial B}{\partial t}, \nabla \times B = \mu_0 j + \frac{1}{c^2} \frac{\partial E}{\partial t}
\]

Maxwell’s equation

Tricky to simulate accurately and efficiently → Can we do better?
Antenna Design problem

Goal:
find the right design to achieve the right frequency response

\[ h = \]

\[ S_{11}(\psi) = \]

![Diagram of 2D antenna design](Image)
Antenna Design problem

Goal:
find the right design to achieve the right frequency response

\[ h = \frac{S_{CC}}{\frac{\pi}{2}} \]

\[ S_{11}(\psi) = \]

Target frequency response

Strong absorption at specific frequency
Discretization of linear PDE systems

\[
\frac{\partial^n \psi}{\partial t^n} = F(\psi, \nabla_x \psi, \ldots; h)
\]

\[
\frac{\partial \phi}{\partial t} = A(h) \phi
\]

The matrix \( A \) encodes the information of \( F \)

One example:

\[
\frac{\partial^2 \psi}{\partial t^2} = c^2 \nabla^2 \psi
\]

\[
\phi = \left[ \psi(x_1), \ldots \psi(x_N), \frac{\partial \psi}{\partial t}(x_1), \ldots, \frac{\partial \psi}{\partial t}(x_N) \right]
\]

Discretized onto \( N \) vertices

\[
A = \begin{bmatrix} 0 & I \\ c^2B & 0 \end{bmatrix}
\]
Frequency Domain

Signal in the temporal domain: $\phi(x, t)$
Vector form $\phi(t) = [\phi(x_1, t), \ldots, \phi(x_N, t)]$

Signal in the frequency domain: $\hat{\phi}(x, \omega) = \int \phi(x, t)e^{-i\omega t} dt$
Vector form $\hat{\phi}(t) = [\hat{\phi}(x_1, \omega), \ldots, \hat{\phi}(x_N, \omega)]$
Parametric formula for Linear PDEs

**Theorem:** For any linear coefficients $b_1$ and $b_2$:

$$\frac{b_1^T \hat{\phi}(\omega)}{b_2^T \hat{\phi}(\omega)} = c_0(h) \prod_{k=1}^{K_1} (\omega - z_k(h)) \prod_{k=1}^{K_2} (\omega - p_k(h))^{-1}$$

where the constant $c_0(h)$, zeros $z_k(h)$ and poles $p_k(h)$ are complex functions of the design choice $h$.

**Proof idea:** Linear ODE theory gives us the analytic form of the solution $\phi(t) = e^{At} \phi(0)$. Fourier Transform yields $\hat{\phi}(\omega)$ as a rational function of complex polynomials w.r.t. frequency $\omega$. 
For Antenna Optimization

The *Scattering Coefficients* $S_{11}(\omega)$:

$$S_{11}(\omega) = \frac{Z_{\text{in}}(\omega) - Z_0}{Z_{\text{in}}(\omega) + Z_0}$$

$Z_{\text{in}}(\omega)$: Input *Impedance*. Impedance $Z(\omega) := V(\omega)/I(\omega)$

Both are linear function w.r.t. signal $\Phi(\omega)$
Parameter form of $\log|S_{11}(\omega)|$

\[
\log|S_{11}(\omega)| = \log|c_0(h)| + \sum_{k=1}^{K} \log \frac{|\omega - z_k(h)|}{|\omega - p_k(h)|}
\]

where the constant $c_0(h)$, zeros $z_k(h)$ and poles $p_k(h)$ are complex functions of the design choice $h$. 
CZP model architecture

Predict the constant, zeros and poles from an image representation of an antenna.
Data Collection

Dataset is collected from commercial simulators (e.g., CST)
Numerical simulation of Electromagnetic wave dynamics

It takes minutes (or even hours) to get one simulation data point.
Dataset size = 48k
Model Evaluation

• Static Evaluation
  • On a held-out test set, compute the loss
  • Loss = gap between surrogate models and ground truth (commercial software)

• Dynamic Evaluation
  • Use the surrogate model to search a good design
  • Evaluate the design in ground truth (commercial software)
Static Evaluation: Surrogate Model Test Loss

- **MLP + Coord**
- **CNN + Image**
- **Transformer + Binary Image**
- **FNO + Image**
- **Transformer + Image**

- **No CZP structure**
- **CZP models with more zeros/poles**

- **Proposed approach**
  - 50% error reduction

Loss on Static test set

- **Raw**
- **CZP (K=8)**
- **CZP (K=12)**
- **CZP (K=16)**
- **CZP (K=20)**
Static Evaluation: Surrogate Model Test Loss

Image-based Representation of design choice $\mathbf{h}$

Loss on Static test set

Legend:
- Raw
- CZP (K=8)
- CZP (K=12)
- CZP (K=16)
- CZP (K=20)
Static Evaluation: Surrogate Model Test Loss

Loss on Static test set

Image-based Representation of design choice $h$

Coordinate-based Representation of design choice $h$
Visualization

Our CZP model captures the smooth structure of scattering coefficients $S_{11}(\omega)$
Dynamic Evaluation: CZP model with Search

We use Soft Actor-Critic as the specific search technique.

**Goal:** to find a solution to satisfy the frequency constraints (verified with CST)

3 models trained x 3 search attempts using different random seed

For each attempt, check top-3 solutions
Future Work

• The formulation applies to general linear PDEs
  • Maxwell’s Equations
  • Schrodinger’s Equations
  • Many more ...

• Test on more complicated scenarios.
  • 3D antenna
Thanks!