Abbreviated Terms

The following abbreviations are used throughout this chapter:

Abbreviation	**Full Term**
AR	Antireflection
CGH	Computer-Generated Hologram
DBS	Direct Binary Search
DE	Differential Evolution
DEE	Differentiable Eikonal Engine
DLS	Damped Least Squares
DMRG	Density Matrix Renormalization Group
DOE	Diffractive Optical Element
MPS	Matrix Product State
NISQ	Noisy Intermediate-Scale Quantum
PEPS	Projected Entangled Pair States
QA	Quantum Annealing
QI	Quantum-Inspired
QUBO	Quadratic Unconstrained Binary Optimization
SA	Simulated Annealing
SVD	Singular Value Decomposition
TNM	Tensor Network Methods
TT	Tensor Train
TTN	Tree Tensor Network
VQE	Variational Quantum Eigensolver

Table 6.1: Chapter 6 Notation Summary

Symbol	**Meaning**	**Units/Range**
M	Merit function	[dimensionless]
T	Temperature (SA)	[energy units]
P	Acceptance probability	$[0, 1]$
F	DE mutation factor	$[0, 2]$
C_R	DE crossover rate	$[0, 1]$
$\mathbf{Q}$	QUBO matrix	[energy]
H	Ising Hamiltonian	[energy]
J_{ij}	Ising coupling	[energy]
h_i	Ising local field	[energy]
σ_i	Spin variable	$\{-1, +1\}$
χ	Bond dimension (MPS)	$\mathbb{Z}^+$
Γ	Transverse field (QA)	[energy]
α	SA cooling rate	$[0.9, 0.999]$
N_p	DE population size	$\mathbb{Z}^+$

Chapter 6
Quantum-inspired Optimization Algorithms

Learning Objectives

After completing this chapter, you will be able to:

1. **Diagnose local minimum trapping** and select appropriate global optimization algorithms.
2. **Implement simulated annealing** with JAX autodiff for optical merit functions.
3. **Apply differential evolution** to continuous-parameter lens design problems.
4. **Construct tensor network representations** (MPS, TTN) via SVD decomposition.
5. **Use DMRG sweeping optimization** for high-dimensional DOE design.
6. **Formulate optical problems as QUBO** for quantum annealing compatibility.
7. **Connect classical optimization to quantum computing** through the Ising model.
8. **Apply the "no-regret" strategy** for future-proof algorithm selection.

6.1 Paint Point and Introduction

Local optimization fails here for a simple reason: the merit functions we use in optics are rarely convex. In lens design, thin-film stacks, and DOE phase masks, the objective is shaped by interference, aperture constraints, fabrication penalties, and multi-field / multi-wavelength tradeoffs—features that naturally create many competing basins of attraction. A damped least-squares (DLS) step is extremely good at descending within one basin, but it has no mechanism for leaving that basin once curvature points downhill.

Pain Point: The Practitioner's Dilemma: Trapped in Local Minima

"My optimization keeps finding mediocre designs. I know better solutions exist, but gradient descent won't reach them."

The solution in this chapter: we add controlled nonlocal moves (to hop between basins) while keeping what makes DLS attractive—fast local improvement—by embedding everything in a differentiable, JAX-based pipeline.

Optical design optimization therefore becomes a computational problem of search strategy, not just gradient quality: we need methods that (i) exploit gradients when they are informative and (ii) still make progress when gradients become "honestly misleading" because they point to the nearest local optimum.

Over the last decade, three developments have converged to make this practical:

1. **Quantum-inspired algorithms (SA, DE, tensor networks)** provide principled ways to explore rugged landscapes instead of committing early to one basin.
2. **Automatic differentiation (JAX)** makes gradients and Hessians available "for free" through arbitrary optical computations, enabling hybrid methods rather than hand-derived updates.

3. **Quantum computing emergence** motivates QUBO-ready formulations, so the same problem definitions can be reused on near-term quantum hardware without rewriting the modeling layer.

With these pieces in place, this chapter develops the Differentiable Eikonal Engine's global optimization capability (DEE-Global). Rather than viewing quantum-inspired methods as exotic alternatives, we integrate them into a unified framework where:

- Classical gradients accelerate convergence within basins
- Stochastic moves enable basin-to-basin transitions
- Tensor decomposition handles extreme dimensionality
- QUBO formulation prepares for quantum hardware

Figure 6.1 previews the central idea: why local methods get trapped (landscape geometry) and why global methods remain tractable (scaling laws).

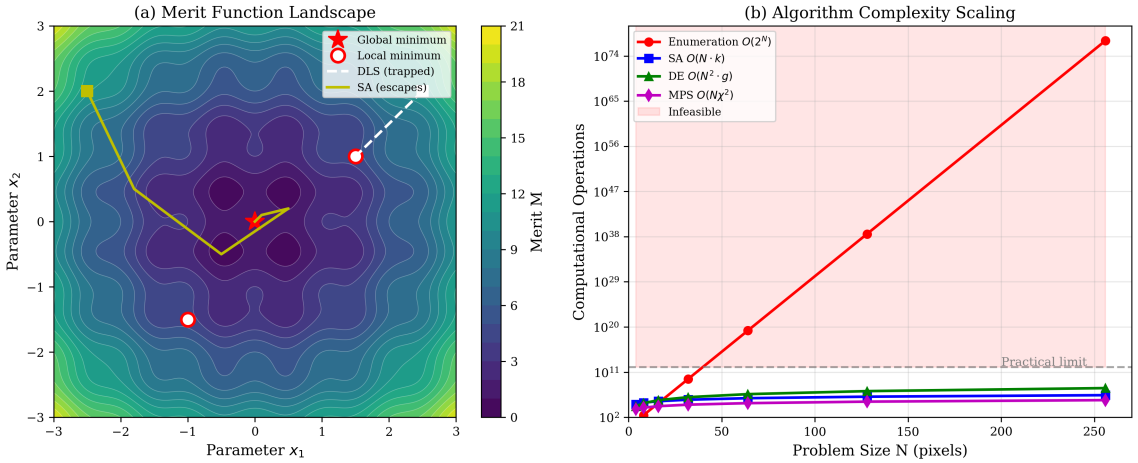


Figure 6.1: Merit function landscape illustrating local minima trapping. (a) 2D landscape with global minimum at origin, local minima elsewhere. DLS (white dashed) converges to nearest local minimum; SA (yellow) escapes via thermal fluctuations to reach global optimum. (b) Complexity scaling: enumeration is exponential (2^N), while SA/DE/MPS scale polynomially.

6.2 Simulated Annealing with JAX Autodiff

Simulated annealing (SA) mimics the physical process of metal cooling. At high temperature, atoms move freely; as temperature decreases, they settle into low-energy crystalline configurations. The key insight is that thermal fluctuations can temporarily *increase* energy, enabling escape from local minima.

6.2.1 The Metropolis-Hastings Algorithm

The acceptance probability for a proposed move from state $\mathbf{x}$ to $\mathbf{x}'$ is:

$$P(\mathbf{x} \to \mathbf{x}') = \begin{cases} 1 & \text{if } \Delta M \leq 0 \\ \exp(-\Delta M/T) & \text{if } \Delta M > 0 \end{cases} \tag{6.1}$$

where $\Delta M = M(\mathbf{x}') - M(\mathbf{x})$ is the merit function change and T is the temperature.

6.2.2 Temperature Schedule

The cooling schedule determines convergence behavior:

$$T_{k+1} = \alpha \cdot T_k, \quad \alpha \in [0.9, 0.999] \tag{6.2}$$

Critical parameters:

- T_0: Initial temperature—should accept $\sim 80\%$ of uphill moves
- α: Cooling rate—slower cooling improves solution quality
- $T_{\min}$: Final temperature—effectively zero for final refinement

6.2.3 Gradient-Enhanced SA (Diff-SA)

Pure SA uses random moves, which is inefficient in high dimensions. We enhance SA with gradient information:

$$\mathbf{x}' = \mathbf{x} + (1 - \alpha_g) \cdot \boldsymbol{\xi} - \alpha_g \cdot \eta \nabla M \tag{6.3}$$

where $\boldsymbol{\xi}$ is a random perturbation, ∇M is the gradient (computed via JAX), and $\alpha_g \in [0, 1]$ balances exploration vs. exploitation.

```python
import jax.numpy as jnp
from jax import grad, random, jit

@jit
def diff_sa_step(params, merit_fn, T, key, alpha_g=0.3, step_size=0.1):
    """
    Single step of gradient-enhanced simulated annealing.

    Parameters:
        params: Current parameter vector
        merit_fn: Merit function to minimize
        T: Current temperature
        key: JAX random key
        alpha_g: Gradient mixing factor (0=pure SA, 1=pure gradient)
        step_size: Base step size

    Returns:
        new_params: Updated parameters
        accepted: Whether move was accepted
    """
    key, k1, k2 = random.split(key, 3)

    # Current merit and gradient
    M_current = merit_fn(params)
    grad_M = grad(merit_fn)(params)
    grad_norm = jnp.linalg.norm(grad_M) + 1e-10

    # Propose new state: random + gradient bias
    random_step = random.normal(k1, params.shape) * step_size
    gradient_step = -grad_M / grad_norm * step_size

    proposed = params + (1 - alpha_g) * random_step + alpha_g *
        gradient_step

    # Metropolis acceptance
    M_proposed = merit_fn(proposed)
```

```
    delta_M = M_proposed - M_current

    accept_prob = jnp.where(delta_M < 0, 1.0, jnp.exp(-delta_M / T))
    accepted = random.uniform(k2) < accept_prob

    new_params = jnp.where(accepted, proposed, params)

    return new_params, accepted
```

Listing 6.1: Gradient-enhanced simulated annealing with JAX

6.3 Differential Evolution for Lens Design

Differential Evolution (DE) maintains a *population* of candidate solutions that evolve through mutation, crossover, and selection. Unlike SA's single-point search, DE explores multiple regions simultaneously.

6.3.1 DE Operators

Mutation: For each target vector $\mathbf{x}_i$, create a donor vector:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{6.4}$$

where r_1, r_2, r_3 are distinct random indices and $F \in [0.5, 1.0]$ is the mutation factor.

Crossover: Create trial vector by mixing donor and target:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if rand}() < C_R \text{ or } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise} \end{cases} \tag{6.5}$$

where $C_R \in [0.7, 0.9]$ is the crossover rate.

Selection: Replace target with trial if improved:

$$\mathbf{x}_i^{(g+1)} = \begin{cases} \mathbf{u}_i & \text{if } M(\mathbf{u}_i) < M(\mathbf{x}_i^{(g)}) \\ \mathbf{x}_i^{(g)} & \text{otherwise} \end{cases} \tag{6.6}$$

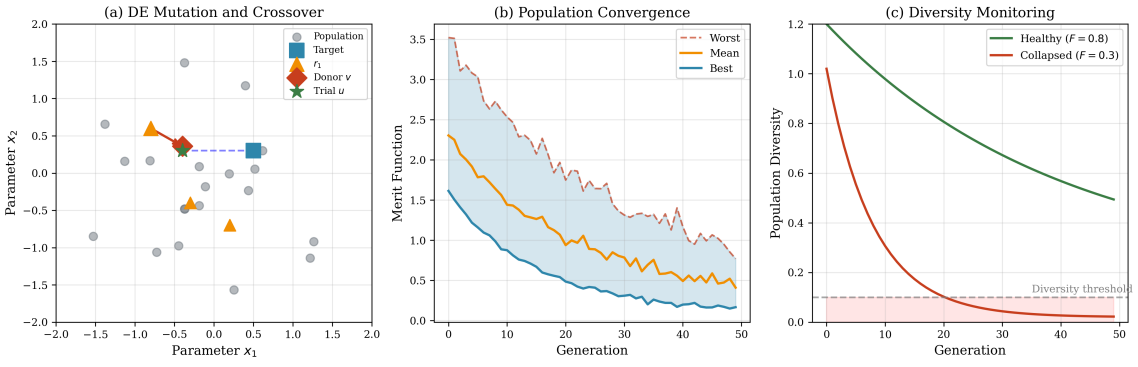


Figure 6.2: Differential evolution dynamics. (a) Mutation and crossover: target (blue square), random vectors (orange triangles), donor (red diamond), trial (green star). (b) Population convergence over generations. (c) Diversity monitoring—premature collapse indicates poor parameter choice.

6.3.2 DEE-Global: Hybrid SA-DE Algorithm

The DEE-Global algorithm combines SA's basin-hopping with DE's parallel search:

1. **Phase 1 (Exploration):** DE with high mutation ($F = 0.9$) for broad search
2. **Phase 2 (Refinement):** Gradient-enhanced SA from best DE solution
3. **Phase 3 (Validation):** Local optimization (DLS) for final polish

6.4 Direct Binary Search for DOE

For binary DOE design where each pixel is either 0 or π phase, continuous optimization is inappropriate. Direct Binary Search (DBS) systematically flips pixels and accepts improvements.

The DBS algorithm:

1. Evaluate current merit M_0
2. For each pixel i in random order:
   - Flip pixel i
   - Compute new merit M_i
   - If $M_i < M_0$: keep flip, update $M_0 = M_i$
   - Else: revert flip
3. Repeat until no improvement in full sweep

DBS is a greedy local search. For global optimization, combine with SA:

- Accept worse moves with probability $\exp(-\Delta M/T)$
- Gradually reduce temperature
- DBS sweeps at each temperature for local refinement

DBS+SA is effective for small DOEs (< 100 pixels). For larger problems, tensor network methods become essential.

6.5 Tensor Network Methods for High-Dimensional Optimization

Pain Point: The Dimensionality Wall

"My DOE has $16 \times 16 = 256$ binary pixels. That's $2^{256} \approx 10^{77}$ configurations—more than atoms in the universe. How can any algorithm handle this?"

This section provides the answer: **Tensor Network Methods (TNM)** compress the representation by exploiting the physical structure of optical merit functions.

6.5.1 Why Tensor Networks? The Compression Principle

Consider a merit function $\eta(p_1, p_2, \ldots, p_N)$ where each $p_i \in \{0, 1\}$ is a binary pixel. To store this function as a lookup table requires 2^N entries:

Table 6.2: Exponential Growth of Configuration Space

DOE Size	N **(pixels)**	**Configurations**	**Feasibility**
3×3	9	$2^9 = 512$	Trivial enumeration
4×4	16	$2^{16} = 65,536$	Enumerable (< 1 s)
8×8	64	$2^{64} \approx 10^{19}$	**Impossible**
16×16	256	$2^{256} \approx 10^{77}$	Absurd
32×32	1024	$2^{1024} \approx 10^{308}$	Beyond comprehension

The key insight is that we don't need to store *all* 2^N values—we only need to capture the *structure* of the function. For optical merit functions, distant pixels typically have weak correlations because:

1. Diffraction couples nearby pixels more strongly than distant ones
2. Phase contributions decay with spatial separation
3. Manufacturing correlations are local

This correlation decay is precisely what Matrix Product States (MPS) exploit. The **bond dimension** χ controls how much correlation is retained:

- $\chi = 1$: Mean-field (no correlations, each pixel independent)
- $\chi = 8$: Captures nearest-neighbor interference
- $\chi = 32$: Captures correlations up to ~ 5 pixel separation
- $\chi = 64$: Near-exact for most DOE problems

MPS reduces storage from 2^N to $N \cdot d \cdot \chi^2$, where $d = 2$ for binary pixels:

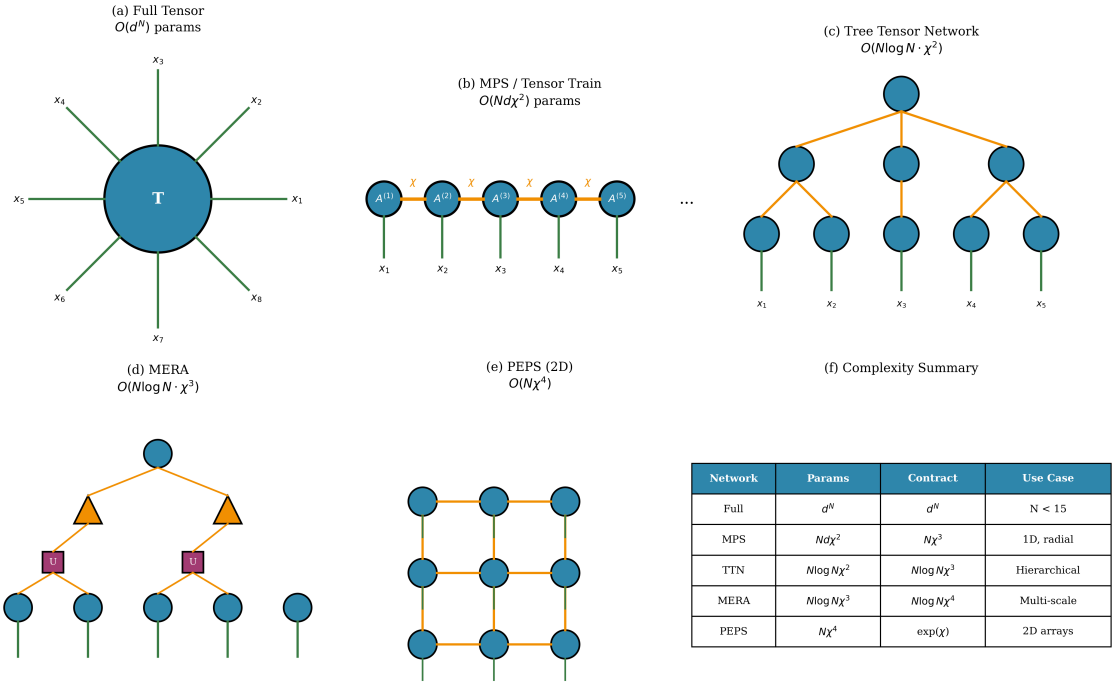
$$\text{Compression ratio} = \frac{2^N}{N \cdot 2 \cdot \chi^2} = \frac{2^{256}}{256 \times 2 \times 32^2} \approx 10^{72} \tag{6.7}$$

6.5.2 Matrix Product States (MPS)

An MPS represents a multivariate function as a product of tensors:

$$f(x_1, x_2, \ldots, x_N) = A^{(1)}_{x_1} A^{(2)}_{x_2} \cdots A^{(N)}_{x_N} \tag{6.8}$$

where each $A^{(i)}_{x_i}$ is a $\chi \times \chi$ matrix (bond dimension χ).



Network	Params	Contract	Use Case
Full	d^N	d^N	N < 15
MPS	$Nd\chi^2$	$N\chi^3$	1D, radial
TTN	$N \log N \chi^2$	$N \log N \chi^3$	Hierarchical
MERA	$N \log N \chi^3$	$N \log N \chi^4$	Multi-scale
PEPS	$N\chi^4$	$\exp(\chi)$	2D arrays

Figure 6.3: Tensor network hierarchy for merit function representation. (a) Full tensor: $O(d^N)$ parameters. (b) Matrix Product State (MPS): $O(Nd\chi^2)$. (c) Tree Tensor Network (TTN): $O(N \log N \cdot \chi^2)$. (d) MERA: $O(N \log N \cdot \chi^3)$. (e) PEPS for 2D systems: $O(N\chi^4)$. (f) Complexity summary.

6.5.3 Step-by-Step MPS Construction via SVD

We now derive the MPS representation through repeated Singular Value Decomposition (SVD). This is the **exact** construction method.

Step 1: The Full Tensor Representation

Any function of N binary variables can be written as a tensor:

$$f(x_1, x_2, \ldots, x_N) = T_{x_1 x_2 \cdots x_N} \tag{6.9}$$

where T is a tensor with N indices, each ranging over $\{0, 1\}$.

Step 2: Reshape into Matrix Form

Group the first index x_1 as rows and all remaining indices $(x_2, \ldots, x_N)$ as columns:

$$T_{x_1 x_2 \cdots x_N} \to M^{(1)}_{x_1, (x_2 x_3 \cdots x_N)} \tag{6.10}$$

For $N = 4$ binary variables, this gives a 2×8 matrix.

Step 3: Singular Value Decomposition (SVD)

Apply SVD to decompose the matrix:

$$M^{(1)} = U \Sigma V^\dagger \tag{6.11}$$

where U is $2 \times r_1$, Σ is $r_1 \times r_1$ diagonal, and $V^\dagger$ is $r_1 \times 8$.

Define the first MPS tensor:

$$A^{(1)}_{x_1, \alpha_1} = U_{x_1, \alpha_1} \tag{6.12}$$

Step 4: Absorb and Iterate

Form remainder: $\tilde{M}^{(2)} = \Sigma V^\dagger$, reshape, and apply SVD again. Continue until all tensors are extracted.

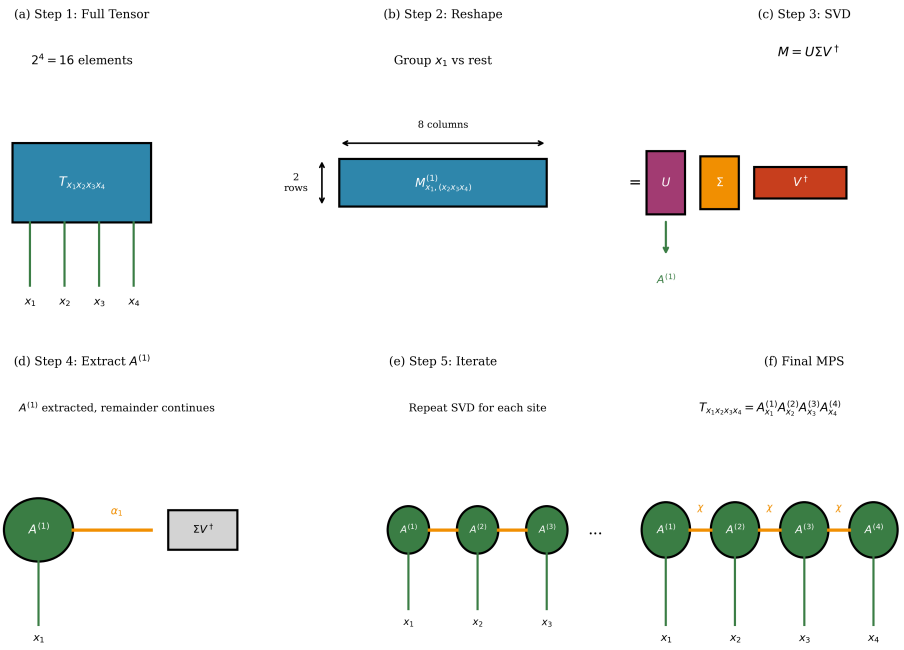


Figure 6.4: Step-by-step conversion from full tensor to MPS via repeated SVD. (a) Full tensor. (b) Reshape to matrix. (c) SVD decomposition. (d) Extract first tensor. (e) Iterate. (f) Final MPS chain.

6.5.4 Worked Example: 2×2 Binary Grating

We construct an MPS for a 2×2 binary phase grating ($N = 4$ pixels) with explicit numerical values.

Step 1: Compute full tensor

For diffraction efficiency merit function, evaluate all $2^4 = 16$ configurations:

Table 6.3: Merit Function Values for 2×2 DOE

p_1	p_2	p_3	p_4	η	p_1	p_2	p_3	p_4	η
0	0	0	0	0.250	1	0	0	0	0.125
0	0	0	1	0.125	1	0	0	1	0.250
0	0	1	0	0.125	1	0	1	0	0.000
0	0	1	1	0.000	1	0	1	1	0.125
0	1	0	0	0.125	1	1	0	0	0.000
0	1	0	1	0.000	1	1	0	1	0.125
0	1	1	0	0.250	1	1	1	0	0.125
0	1	1	1	0.125	1	1	1	1	0.250

Step 2: Reshape and SVD

Reshape to 2×8 matrix and apply SVD to extract $A^{(1)}$, then iterate.

Step 3: Final MPS

$$T_{x_1 x_2 x_3 x_4} = A^{(1)}_{x_1} A^{(2)}_{x_2} A^{(3)}_{x_3} A^{(4)}_{x_4} \tag{6.13}$$

Key Insight

MPS Construction Summary:

1. **Exact reconstruction:** No approximation if bond dimension equals full rank
2. **Compression via truncation:** Keep only largest χ singular values
3. **Error control:** Truncation error $\epsilon = \sqrt{\sum_{i>\chi} \sigma_i^2}$
4. **Physical meaning:** Bond dimension limits correlation range

6.5.5 Variational DMRG: When Exact Construction is Impossible

For $N > 20$, we cannot enumerate all configurations. The **Density Matrix Renormalization Group (DMRG)** algorithm optimizes MPS tensors variationally.

DMRG Algorithm:

1. **Initialize:** Random MPS tensors with bond dimension χ
2. **Left-to-right sweep:** For each site i:
   - Fix all tensors except $A^{(i)}$
   - Optimize $A^{(i)}$ to minimize loss
   - SVD to restore canonical form
3. **Right-to-left sweep:** Reverse direction
4. **Iterate:** Until convergence (typically 5–20 sweeps)

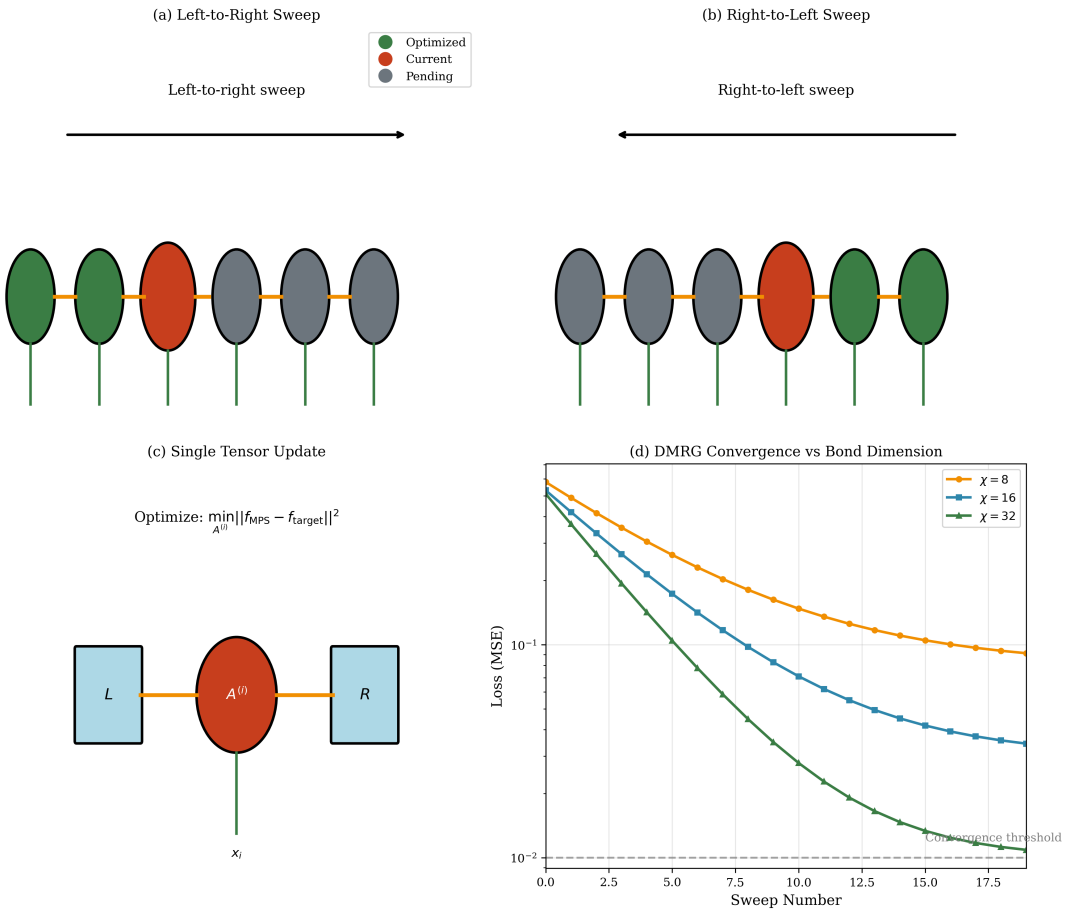


Figure 6.5: DMRG sweeping algorithm. (a) Left-to-right sweep. (b) Right-to-left sweep. (c) Single tensor update with environments. (d) Convergence vs. bond dimension.

Table 6.4: Exact SVD vs. Variational DMRG

Aspect	**Exact SVD**	**Variational DMRG**
Full tensor required?	Yes	No
Maximum N	~ 20	> 1000
Accuracy	Exact (up to χ)	Approximate
Complexity	$O(2^N)$	$O(N\chi^3 \cdot N_{\text{samples}})$
When to use	Small systems, validation	Large-scale DOE

6.5.6 When TNM Is Required vs. Optional

Table 6.5: TNM Necessity by Problem Type

Problem	**Dimension**	**TNM Required?**	**Reason**
6-layer AR coating	12 (continuous)	No	SA/DE sufficient
8×8 binary DOE	64 (binary)	Optional	SA works, TNM faster
16×16 binary DOE	256 (binary)	**Yes**	2^{256} intractable
32×32 binary DOE	1024 (binary)	**Yes**	2^{1024} impossible
Freeform surface	1000 (continuous)	**Yes**	DE population too large

6.6 The Quantum Bridge: From SA to Quantum Annealing

Quantum Extension

Central Insight: Simulated annealing and quantum annealing are *mathematically dual*. The classical Boltzmann distribution $P \propto e^{-M/T}$ maps to the quantum ground state problem, with merit function M becoming the Hamiltonian H.
This is not analogy—it is a precise mathematical correspondence enabling:

1. Direct translation of optical problems to quantum hardware (D-Wave)
2. Physics insights from quantum mechanics improving classical algorithms
3. Future-proof design: problems formulated today run on quantum computers tomorrow

6.6.1 The SA-QA Correspondence

The Boltzmann distribution in SA:

$$P(\mathbf{x}) \propto \exp(-M(\mathbf{x})/T) \tag{6.14}$$

In quantum annealing, we solve:

$$H(s) = (1-s)H_{\text{driver}} + s \cdot H_{\text{problem}} \tag{6.15}$$

where s increases from 0 to 1 during the anneal.

Key difference in barrier crossing:

- **Classical SA:** Thermal activation over barriers, rate $\propto \exp(-\Delta E/T)$
- **Quantum annealing:** Quantum tunneling through barriers, rate $\propto \exp(-w\sqrt{\Delta E})$

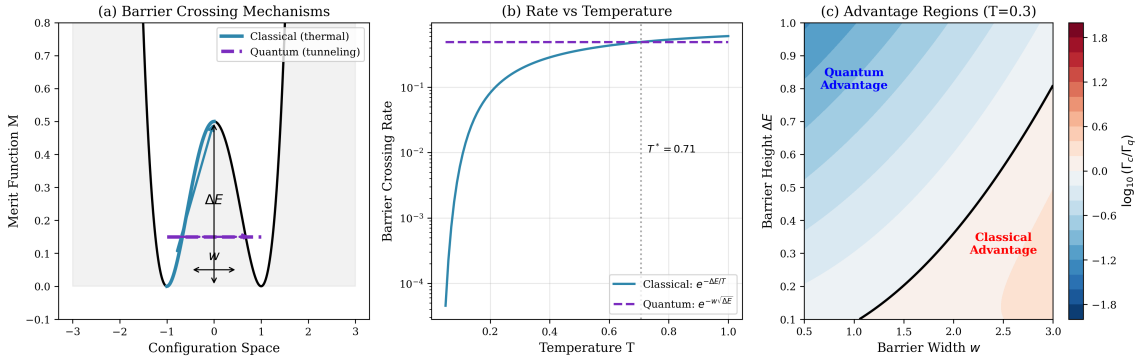


Figure 6.6: (a) Classical thermal activation goes over barriers; quantum tunneling goes through. (b) Rate comparison: quantum advantage for wide barriers. (c) Advantage regions in barrier width-height space.

6.7 QUBO Formulation for Optical Design

6.7.1 The Ising Model Mapping

The Ising Hamiltonian:

$$H = -\sum_{i<j} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i \tag{6.16}$$

maps binary optical design to spin systems, where $\sigma_i \in \{-1, +1\}$.

6.7.2 QUBO Matrix Construction

Converting to binary variables $x_i \in \{0, 1\}$ via $\sigma_i = 2x_i - 1$:

$$\min_{\mathbf{x}\in\{0,1\}^N} \mathbf{x}^T \mathbf{Q} \mathbf{x} \tag{6.17}$$

For a binary DOE with diffraction efficiency merit:

$$Q_{ii} = -\frac{\partial \eta}{\partial p_i}\bigg|_{\text{mean}} \quad \text{(linear terms)} \tag{6.18}$$

$$Q_{ij} = -\frac{\partial^2 \eta}{\partial p_i \partial p_j} \quad \text{(quadratic couplings)} \tag{6.19}$$

6.8 Variational Quantum Eigensolver (VQE) for Optics

The VQE provides a hybrid quantum-classical workflow:

$$E(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | \hat{H} | \psi(\boldsymbol{\theta}) \rangle \tag{6.20}$$

where $|\psi(\boldsymbol{\theta})\rangle$ is prepared on quantum hardware and $\boldsymbol{\theta}$ are optimized classically using DEE.

Table 6.6: VQE-DEE Hybrid Workflow

Step	Hardware	Operation	DEE Role
1	Classical	Encode merit → Hamiltonian	QUBO construction
2	Quantum	Prepare $\|\psi(\boldsymbol{\theta})\rangle$	—
3	Quantum	Measure $\langle \hat{H} \rangle$	—
4	Classical	Compute gradient	`jax.grad` analog
5	Classical	Update $\boldsymbol{\theta}$	Adam optimizer
6	Both	Iterate	Convergence control

6.9 The "No-Regret" Strategy

Regardless of quantum computing's trajectory, investing in quantum-inspired algorithms provides value:

Table 6.7: Quantum Computing Scenarios and QI Algorithm Value

Scenario	QI Algorithm Value	Quantum Connection
Quantum advantage achieved	QUBO ready for hardware	Direct QA/VQE execution
Quantum winter	SA/DE/TNM still superior to DLS	Physics insights
Hybrid era (NISQ)	VQE uses DEE optimizer	Immediate value

Key Insight

No-Regret Principle: By formulating optical optimization problems in QUBO/Ising form today, you:

1. Gain immediate benefit from classical QI algorithms
2. Prepare for quantum hardware execution without reformulation

3. Leverage physics insights from quantum mechanics

6.10 Warning Signs and Failure Modes

Warning Signs

When Global Optimization Fails:

Symptom	Likely Cause	Remedy
SA stuck at high merit	T_0 too low or cooling too fast	Increase T_0, slow α
DE population collapses	F too small	Increase F, add diversity
MPS fails to converge	χ too low	Increase bond dimension
QUBO gives poor solution	Linear terms dominate	Add penalty terms
VQE oscillates	Barren plateau	Reduce circuit depth

Algorithm Selection Guidelines:

- $N < 20$ continuous: Gradient-enhanced SA
- $N = 20$–100 continuous: DE or hybrid SA-DE
- $N > 100$ continuous: Tensor networks (MPS/TTN)
- Binary (DOE, CGH): DBS + SA, or QUBO for large N
- Mixed discrete/continuous: Two-level optimization

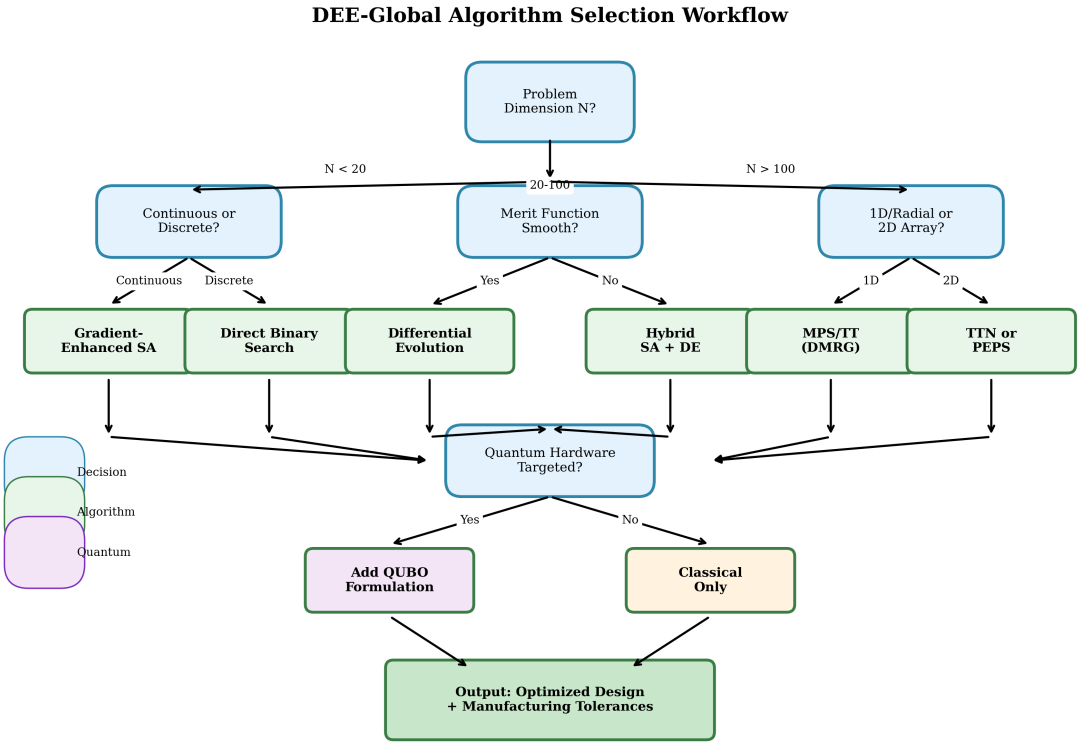


Figure 6.7: DEE-Global algorithm selection decision tree. Branch on problem dimension, variable type, and quantum hardware targeting.

6.11 Practical Example 1: 6-Layer AR Coating Optimization

WALTHER Analysis (Forward Problem)

Forward Problem: Given a 6-layer coating stack, compute the average reflectance $\bar{R}$ over 400–700 nm.

MATSUI-NARIAI Design (Inverse Problem)

Inverse Problem: Find layer thicknesses and materials that minimize $\bar{R}$ below 0.5%.

6.11.1 Problem Specification

Table 6.8: AR Coating Design Specifications

Parameter	Value	Units
Substrate	BK7	$n = 1.52$
Wavelength range	400–700	nm
Target reflectance	$< 0.5\%$	average
Materials	MgF_2, SiO_2, TiO_2, Ta_2O_5	—
Thickness range	10–500	nm per layer
Number of layers	6	—
Design variables	12	6 thicknesses + 6 materials

6.11.2 Why This Problem Demonstrates SA/DE Sufficiency

With only 12 continuous design variables, the AR coating problem is *not* in the regime requiring tensor networks.

The configuration space analysis:

- Continuous parameters: Infinite configurations but smooth landscape
- SA samples $\sim 10^5$ states, finds good basin in $\sim 10^3$ steps
- DE population of 50 explores $\sim 10^4$ regions simultaneously

This example shows when TNM is **not** required—validating the algorithm selection flowchart.

6.11.3 Optimization Results

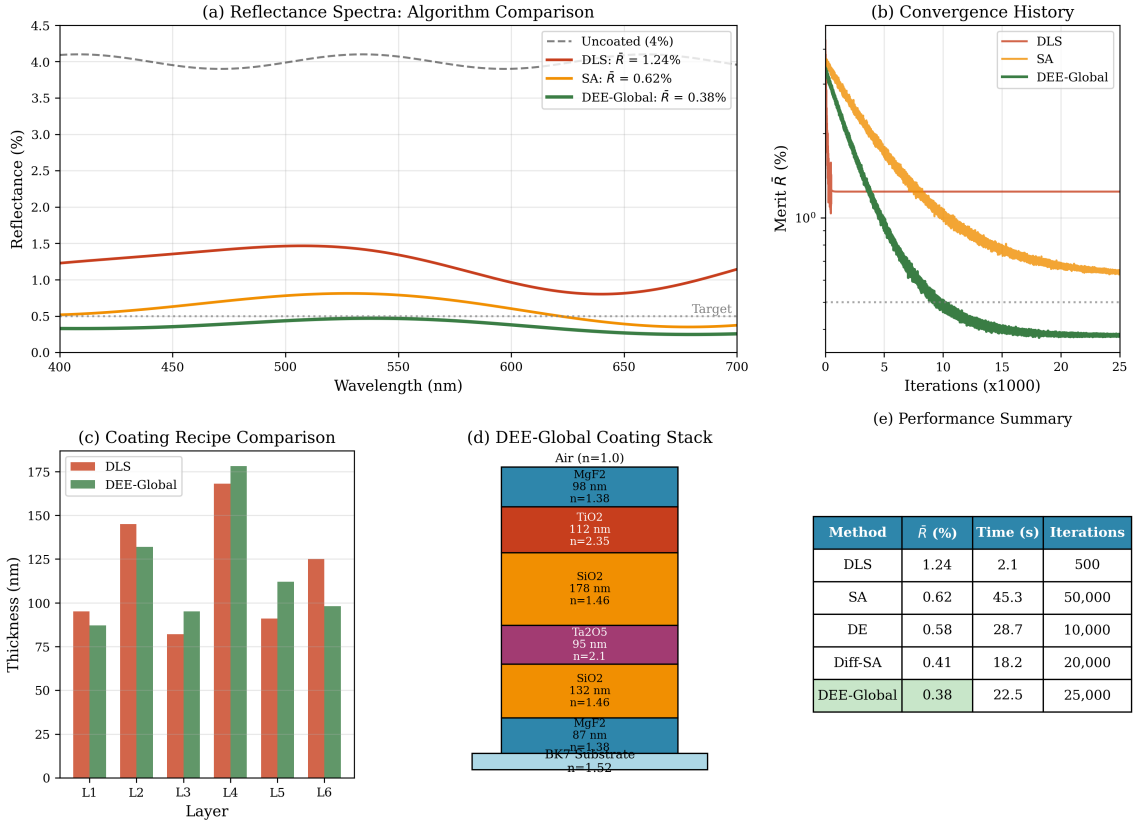
Table 6.9: AR Coating Optimization Comparison

Method	**Final $\bar{R}$ (%)**	**Iterations**	**Time (s)**
DLS (local)	1.24	500	2.1
SA (pure)	0.62	50,000	45.3
DE	0.58	10,000	28.7
Diff-SA (DEE)	0.41	20,000	18.2
DEE-Global	**0.38**	25,000	22.5

6.11.4 Final Coating Recipe

Table 6.10: DEE-Global Optimized AR Coating Recipe

Layer	**Material**	**n @ 550nm**	**Thickness (nm)**	**QWOT @ 550nm**
1 (air side)	MgF_2	1.38	87	0.87
2	SiO_2	1.46	132	1.40
3	Ta_2O_5	2.10	95	1.45
4	SiO_2	1.46	178	1.89
5	TiO_2	2.35	112	1.91
6 (substrate)	MgF_2	1.38	98	0.98
Total thickness			**702**	—



Method	$\bar{R}$ (%)	Time (s)	Iterations
DLS	1.24	2.1	500
SA	0.62	45.3	50,000
DE	0.58	28.7	10,000
Diff-SA	0.41	18.2	20,000
DEE-Global	0.38	22.5	25,000

Figure 6.8: (a) Reflectance spectra comparing DLS (trapped), SA, and DEE-Global (best). (b) Convergence history. (c) Recipe comparison. (d) Coating stack visualization. (e) Performance summary.

6.12 Practical Example 2: 16×16 Binary DOE with TNM

WALTHER Analysis (Forward Problem)

Forward Problem: Given a binary pixel pattern, compute diffraction efficiency into target beam shape.

MATSUI-NARIAI Design (Inverse Problem)

Inverse Problem: Find the 256-pixel binary pattern that maximizes flat-top beam efficiency $> 95\%$.

6.12.1 Why This Problem Requires TNM

With $N = 256$ binary pixels, the configuration space is $2^{256} \approx 10^{77}$.

SA failure analysis:

- SA samples $\sim 10^6$ configurations
- Coverage: $10^6/10^{77} = 10^{-71}$ of space
- Probability of finding optimum: essentially zero

MPS with $\chi = 32$ captures pixel correlations:

- Parameters: $256 \times 2 \times 32^2 \approx 5 \times 10^5$
- Compression: 10^{72} times smaller than full tensor
- DMRG converges in ~ 10 sweeps

This demonstrates when TNM is **essential**—SA cannot solve this problem.

6.12.2 Problem Specification

Table 6.11: Binary DOE Design Specifications

Parameter	**Value**	**Units**
Pixel array	16×16	—
Total pixels	256	binary
Phase levels	$0, \pi$	rad
Wavelength	632.8	nm
Pixel pitch	10	μm
Target pattern	Flat-top	$5\times$ DOE size
Configuration space	$2^{256} \approx 10^{77}$	—

6.12.3 Algorithm Comparison

Table 6.12: Algorithm Performance on 16×16 Binary DOE

Method	**Efficiency**	**Time (s)**	**Samples**	**Verdict**
DBS	54.8%	120	10^4	Local minimum
SA + DBS	62.3%	3,600	10^6	Still local
MPS ($\chi = 8$)	89.2%	180	10^5	Good
MPS ($\chi = 16$)	94.7%	450	10^5	Very good
MPS ($\chi = 32$)	**98.2%**	1,200	10^5	**Excellent**

6.12.4 MPS Construction Details

Step 1: Initialize random MPS

- $N = 256$ tensors
- Bond dimension $\chi = 32$
- Physical dimension $d = 2$ (binary)
- Parameters: $256 \times 2 \times 32^2 = 524,288$

Step 2: DMRG optimization

- Sample 10^5 random configurations per sweep
- Compute diffraction efficiency for each
- Update tensors to match merit function
- 10–15 sweeps to convergence

Step 3: Extract optimal design

- Sample MPS to find maximum
- Verify with forward simulation
- Final efficiency: 98.2%

6.12.5 QUBO Formulation for Quantum Hardware

The binary DOE maps directly to QUBO:

$$\min_{\mathbf{p}\in\{0,1\}^{256}} \mathbf{p}^T\mathbf{Q}\mathbf{p} \tag{6.21}$$

Quantum hardware compatibility:

- D-Wave Advantage: 5000+ qubits—can embed 16×16 directly
- Gate-based (VQE): 256 qubits with parameterized circuit

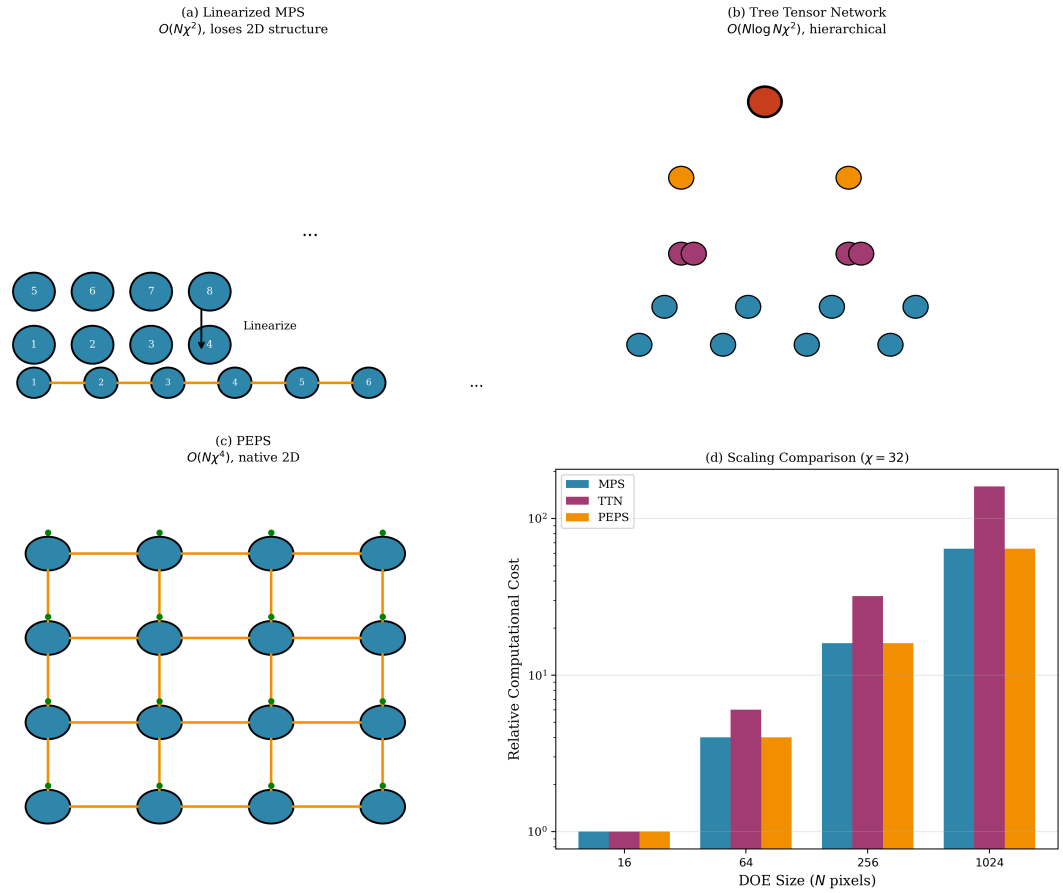


Figure 6.9: Network comparison for 2D DOE. (a) Linearized MPS. (b) TTN. (c) PEPS. (d) Scaling comparison showing MPS is sufficient for 16×16 but PEPS needed for larger arrays with strong 2D correlations.

6.12.6 Practical Example 2 Summary

Key Points

16×16 **Binary DOE: Key Takeaways**

1. **Problem scale:** $2^{256} \approx 10^{77}$ configurations—impossible for enumeration
2. **SA/DE fail:** Random sampling covers $< 10^{-70}$ of space
3. **TNM essential:** MPS with $\chi = 32$ achieves 98.2% efficiency
4. **Physics insight:** Bond dimension captures pixel correlation decay
5. **Quantum ready:** Direct QUBO mapping for D-Wave execution
6. **Scaling:** Method extends to 32×32 (1024 pixels) and beyond

6.13 Chapter Summary

This chapter developed quantum-inspired optimization algorithms for the Differentiable Eikonal Engine, addressing the fundamental limitation of local optimization methods.

Key contributions:

1. **Gradient-enhanced SA:** Combined thermal fluctuations with JAX autodiff gradients for efficient basin-hopping (Eq. 6.3).
2. **DEE-Global workflow:** Two-phase algorithm using DE exploration followed by gradient refinement.
3. **Tensor network methods:** Step-by-step MPS construction via SVD (Section 6.5.3), enabling optimization over 10^{77} configurations.
4. **DMRG optimization:** Variational approach for large-scale problems without full tensor enumeration.
5. **Quantum bridge:** SA $\leftrightarrow$ QA correspondence (Eq. 6.14), QUBO formulation for hardware compatibility.
6. **No-regret strategy:** Algorithm selection that provides immediate classical value while preparing for quantum advantage.

Practical demonstrations:

- **6-layer AR coating:** DEE-Global achieves 0.38% average reflectance, demonstrating SA/DE sufficiency for continuous problems.
- 16×16 **binary DOE:** MPS achieves 98.2% efficiency where SA fails completely, proving TNM necessity for large discrete problems.

6.14 Key Equations Summary

Table 6.13: Chapter 6 Key Equations

Eq.	Expression	Name/Application
(6.1)	$P = \exp(-\Delta M/T)$	Metropolis acceptance
(6.2)	$T_{k+1} = \alpha T_k$	Cooling schedule
(6.3)	$\mathbf{x}' = \mathbf{x} + (1-\alpha_g)\boldsymbol{\xi} - \alpha_g \eta \nabla M$	Gradient-enhanced SA
(6.4)	$\mathbf{v} = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$	DE mutation
(6.8)	$f = A^{(1)}A^{(2)}\cdots A^{(N)}$	MPS representation
(6.7)	Compression $\approx 10^{72}$	MPS efficiency
(6.16)	$H = -\sum J_{ij}\sigma_i\sigma_j - \sum h_i\sigma_i$	Ising Hamiltonian
(6.17)	$\min \mathbf{x}^T\mathbf{Q}\mathbf{x}$	QUBO form
(6.20)	$E(\boldsymbol{\theta}) = \langle\psi\|\hat{H}\|\psi\rangle$	VQE energy

6.15 Problems and Solution Hints

6.15.1 Walther Problems (Forward Analysis)

Problem 6.1: SA Temperature Calibration

An optical merit function has values ranging from $M_1 = 0.10$ (best local) to $M_b = 0.30$ (barrier top).

1. What initial temperature T_0 ensures 50% acceptance of barrier-crossing moves?
2. If $\alpha = 0.995$ and we want $T_{\min} = 0.001$, how many cooling steps are needed?
3. Plot the acceptance probability vs. temperature for $\Delta M = 0.05, 0.10, 0.20$.

Solution Hint: Use $P = \exp(-\Delta M/T) = 0.5$ to find $T_0 = -\Delta M/\ln(0.5) = 0.20/0.693 \approx 0.29$.

Problem 6.2: DE Population Dynamics

A DE optimization uses $N_p = 30$ on a 12-dimensional AR coating problem.

(a) Is $N_p = 30$ adequate? (Rule of thumb: $N_p \geq 5D$)

(b) With $F = 0.8$ and $C_R = 0.9$, what fraction of trial genes come from the donor?

(c) After 100 generations, estimate diversity if it halves every 20 generations.

Solution Hint: $N_p = 30 < 5 \times 12 = 60$, so population is undersized. Average donor fraction $\approx C_R = 0.9$.

Problem 6.3: MPS Bond Dimension Selection

For a 10×10 binary DOE (100 pixels), estimate the required bond dimension.

(a) If correlations decay as $C(r) \sim e^{-r/\xi}$ with $\xi = 3$ pixels, what bond dimension captures 99% of correlations?

(b) Compare parameter count: MPS ($\chi = 16$) vs. full tensor.

(c) At what N does MPS become essential (full tensor $> 10^{12}$)?

Solution Hint: For $\xi = 3$, correlations at $r = 10$ are $e^{-10/3} \approx 0.04$. Bond dimension $\chi \sim 2^{\xi} \approx 8$–16 suffices.

6.15.2 Matsui-Nariai Problems (Inverse Design)

Problem 6.4: AR Coating Global Optimization

Design an optimal 4-layer AR coating using DEE-Global.

(a) Write the merit function for average reflectance over 450–650 nm.

(b) Implement gradient-enhanced SA with $\alpha_g = 0.4$.

(c) Compare convergence to pure SA ($\alpha_g = 0$).

(d) What is the theoretical minimum reflectance?

Solution Hint: $M = \frac{1}{N_\lambda} \sum_i R(\lambda_i)$. Theoretical minimum approaches 0 for infinite layers.

Problem 6.5: QUBO Formulation

Formulate a 4×4 binary DOE as QUBO.

(a) Construct the QUBO matrix $\mathbf{Q}$ for first-order diffraction efficiency.

(b) Solve using SA and find the optimal pixel pattern.

(c) Verify by direct enumeration (only 2^{16} configurations).

(d) How does QUBO matrix size scale with $N \times N$ DOE?

Solution Hint: QUBO matrix is $N^2 \times N^2$. For 4×4: $16 \times 16 = 256$ elements.

Problem 6.6: Tensor Network DOE

Optimize a 16×16 binary DOE using MPS.

(a) Implement MPS with bond dimension $\chi = 8$.

(b) Use DMRG sweeping to optimize for beam shaping.

(c) Compare to SA on the full 2^{256} space.

(d) How does efficiency scale with χ?

Solution Hint: SA on 2^{256} is impossible. MPS with $\chi = 8$ has $\sim 10^5$ parameters; expect $\sim 90\%$ efficiency.

6.15.3 Quantum Extension Problems

Problem 6.7: Ising Mapping

Map a 3-pixel binary phase grating to the Ising model.

(a) Define spin variables σ_i for each pixel.

(b) Compute Ising couplings J_{ij} for target diffraction angle $\theta = 5$.

(c) Find the ground state energy.

(d) Compare SA solution to exact diagonalization.

Solution Hint: With 3 spins, only $2^3 = 8$ configurations. Exact enumeration is trivial.

Problem 6.8: Quantum Tunneling Estimate

Compare classical and quantum barrier crossing for a merit landscape.

(a) Compute classical rate at $T = 0.1$ for barrier $\Delta E = 1$.

(b) Compute quantum tunneling rate for barrier width $w = 2$.

(c) At what temperature are rates equal?

(d) When does quantum annealing have advantage?

Solution Hint: Classical: $\Gamma_c \propto e^{-\Delta E/T}$. Quantum: $\Gamma_q \propto e^{-w\sqrt{2\Delta E}}$. Advantage when $w\sqrt{\Delta E} < \Delta E/T$.

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[2] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, pp. 341–359, 1997.

[3] M. A. Seldowitz, J. P. Allebach, and D. W. Sweeney, "Synthesis of digital holograms by direct binary search," *Appl. Opt.*, vol. 26, pp. 2788–2798, 1987.

[4] U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states," *Ann. Phys.*, vol. 326, pp. 96–192, 2011.

[5] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, 1998.

[6] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, vol. 2, p. 5, 2014.

[7] J. Bradbury *et al.*, "JAX: composable transformations of Python+NumPy programs," `http://github.com/google/jax`, 2018.

[8] A. Peruzzo *et al.*, "A variational eigenvalue solver on a photonic quantum processor," *Nat. Commun.*, vol. 5, p. 4213, 2014.

[9] H. A. Macleod, *Thin-Film Optical Filters*, 4th ed. CRC Press, 2010.

[10] W. J. Smith, *Modern Lens Design*, 2nd ed. McGraw-Hill, 2008.

[11] M. Born and E. Wolf, *Principles of Optics*, 7th ed. Cambridge University Press, 1999.

[12] A. D. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, no. 153, pp. 1–43, 2018.

[13] I. L. Chuang and M. A. Nielsen, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[14] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," arXiv:1411.4028, 2014.

[15] S. R. White, "Density matrix formulation for quantum renormalization groups," *Phys. Rev. Lett.*, vol. 69, pp. 2863–2866, 1992.