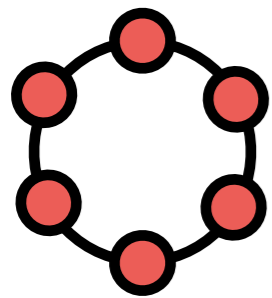


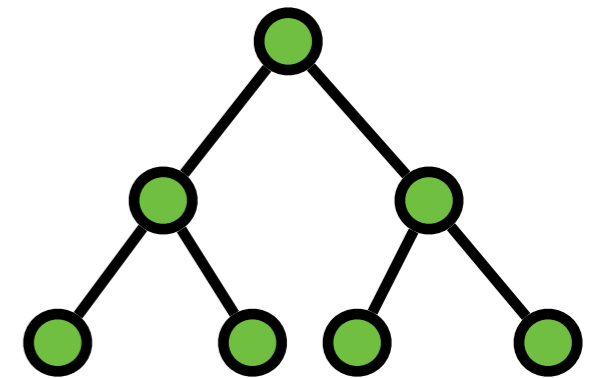
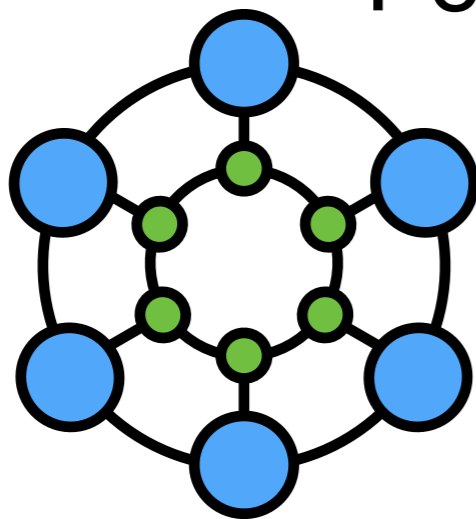
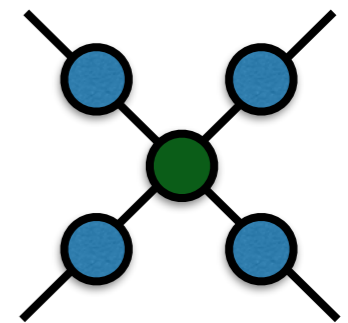
# Evolutionary Structure Learning for Tensor Network Decomposition



Chao Li

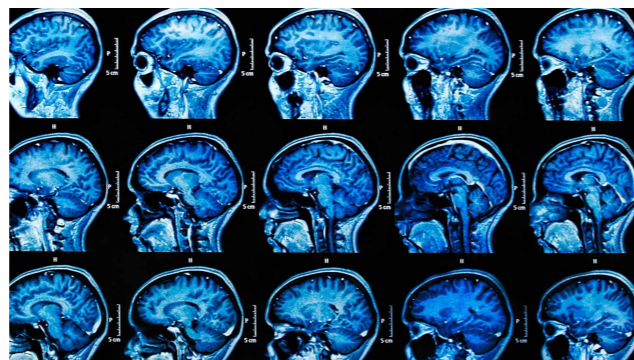
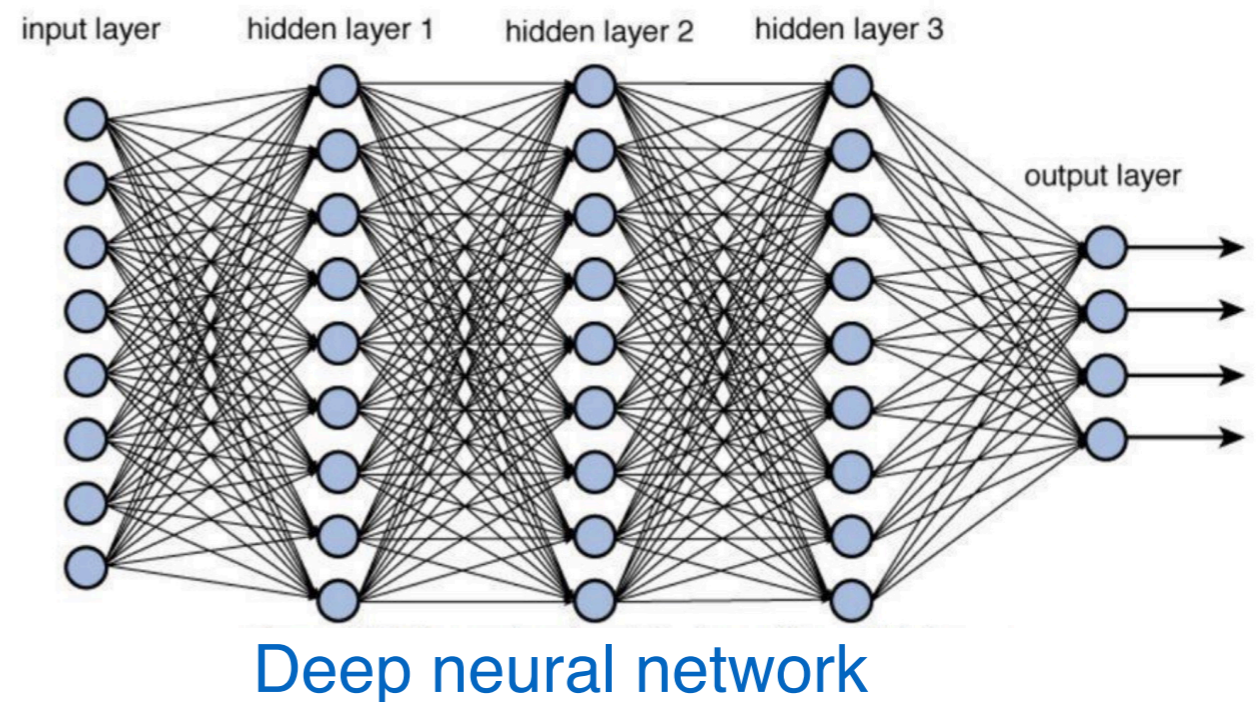
Postdoc researcher at TLT  
RIKEN-AIP

[chao.li@riken.jp](mailto:chao.li@riken.jp)



# Learning with Bigger Data

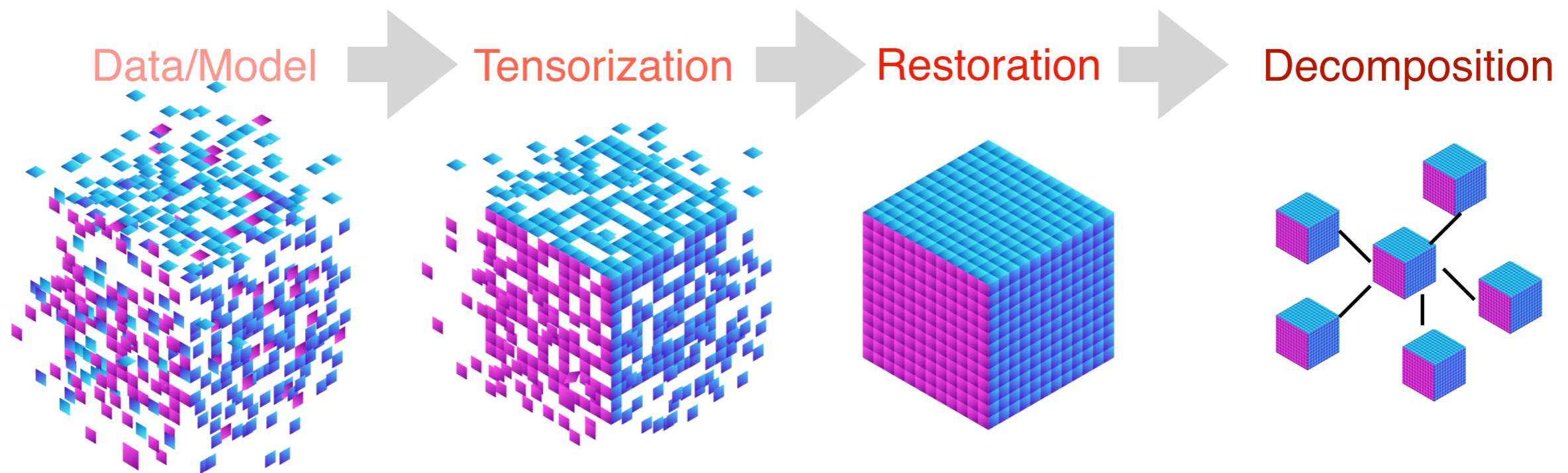
- ▶ Bigger data always imply we can learn more knowledge.
  - ▶ High-dimensional, multi-modal, and incomplete
  - ▶ Big data need bigger models: **Over-parameterization**



How to efficiently represent a HD problem with few parameters?

# Our Solution: Tensor Network (TN)

- ▶ TN is not a learning model but an efficient model representation.
  - ▶ TNs efficiently exploit the low-rank structures of the model.
  - ▶ There are lots of theoretical results to analyze the performance.



Tensor network is a simple yet promising framework.

# From Topology Search to Structure Learning

---

## Evolutionary Topology Search for Tensor Network Decomposition: an Extended Abstract

---

Chao Li<sup>1</sup>, Zhun Sun<sup>1</sup>, Junhua Zeng<sup>1,2</sup>, and Qibin Zhao<sup>1\*</sup>

<sup>1</sup>RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan

<sup>2</sup>Guangdong University of Technology  
{chao.li, qibin.zhao}@riken.jp

### Abstract

Tensor network (TN) decomposition is a promising framework to represent extremely high-dimensional problems with few parameters. However, it remains challenging to search the (near-)optimal topological structures for TN decomposition, since the number of candidate solutions exponentially grows with increasing the order of a tensor. In this work, we claim that the issue can be *practically* tackled by genetic algorithms (GAs) in an affordable manner, and the key is to encode the complex topological structures into fixed-length binary strings, *a.k.a.*, chromosomes in the context of GA. The experimental results on natural images demonstrate that, in the decomposition task, GA is able to discover more efficient topologies than the well-known TN models within a small number of generations. Our code is available at <https://github.com/minogame/icml2020-TNGA>.

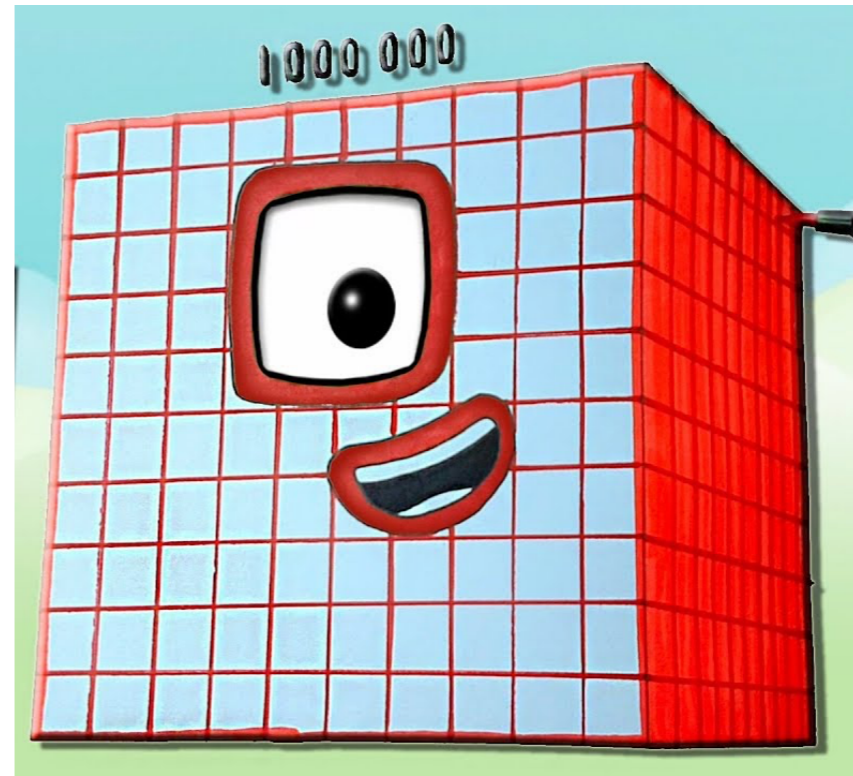
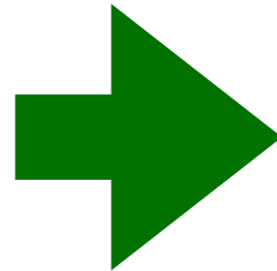
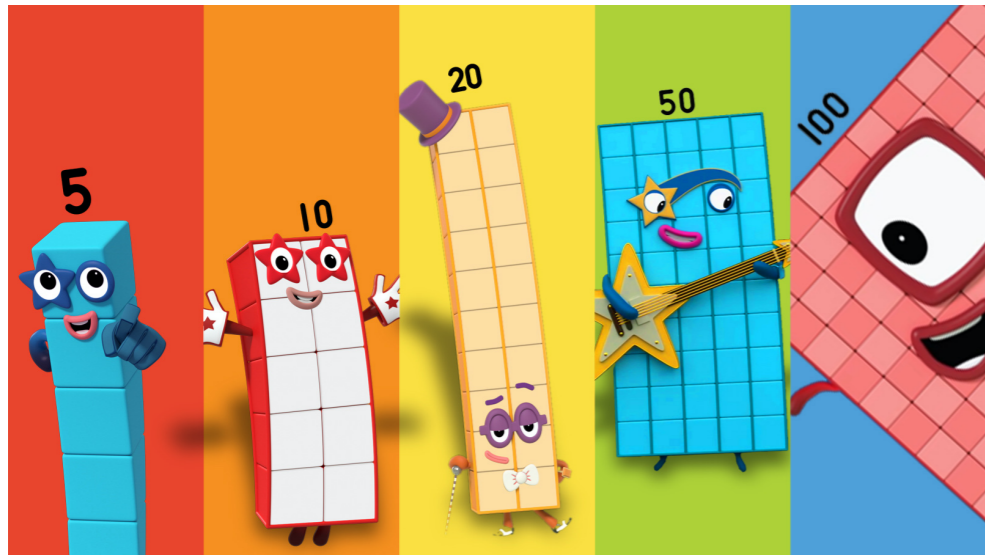
# Outline

- ▶ Preliminaries: Tensor network and diagram notation
- ▶ Structure Learning of Tensor Network Decomposition
- ▶ Tensor Network Decomposition Meets Genetic Algorithm

# Part 1: Tensor and Tensor Network

# Tensor is Ubiquitous

Intuitively, **TENSOR** is number-block.



Vectors and matrices

Tensor cubic

Scalar:  $a, b \in \mathbb{R}$

Vector:  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^I$

Matrix:  $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{I_1 \times I_2}$

Tensor:  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_p}$



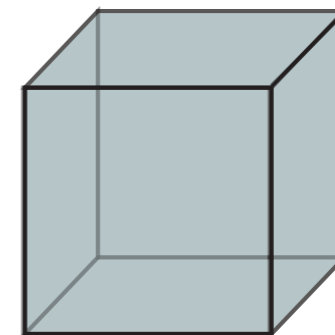
Order-0  
Tensor  
scalar



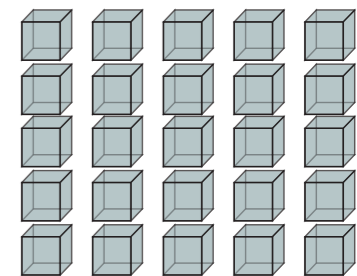
Order-1  
Tensor  
vector



Order-2  
Tensor  
matrix



Order-3  
Tensor



Order-5  
Tensor

# Diagram Notation



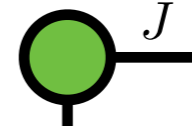
$(1 \times 1)$

Scalar



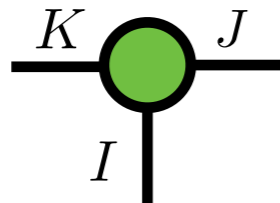
$(I \times 1)$

Vector



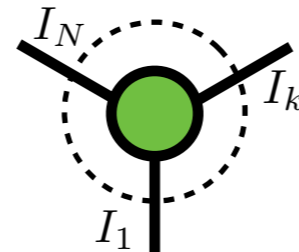
$(I \times J)$

Matrix



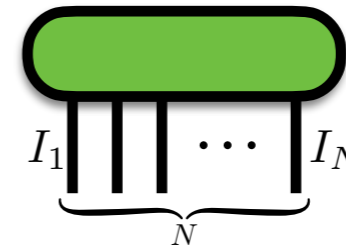
$(I \times J \times K)$

order-3 Tensor



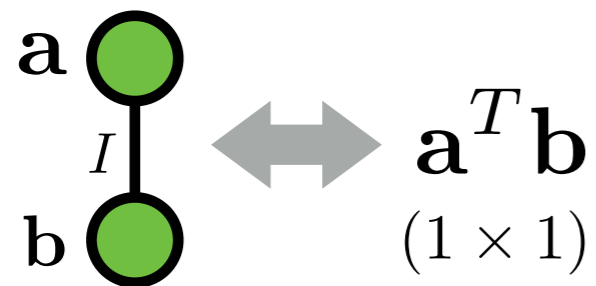
$(I_1 \times I_2 \times \dots \times I_N)$

order- $N$  Tensor

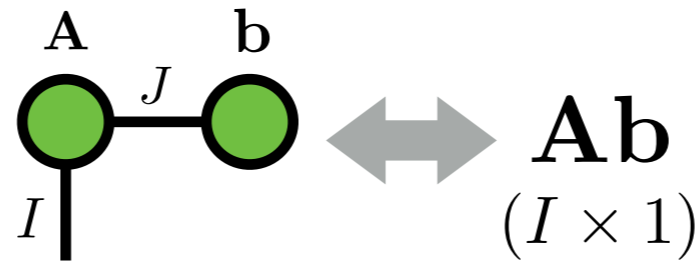




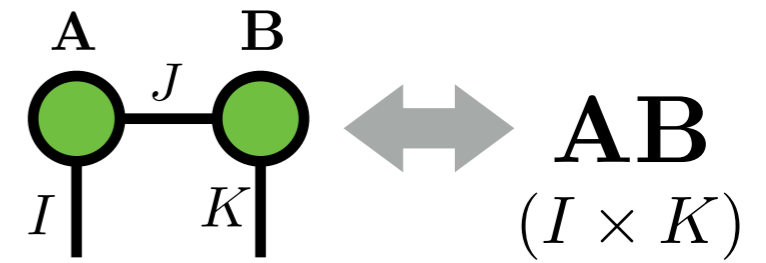
# Tensor Contraction



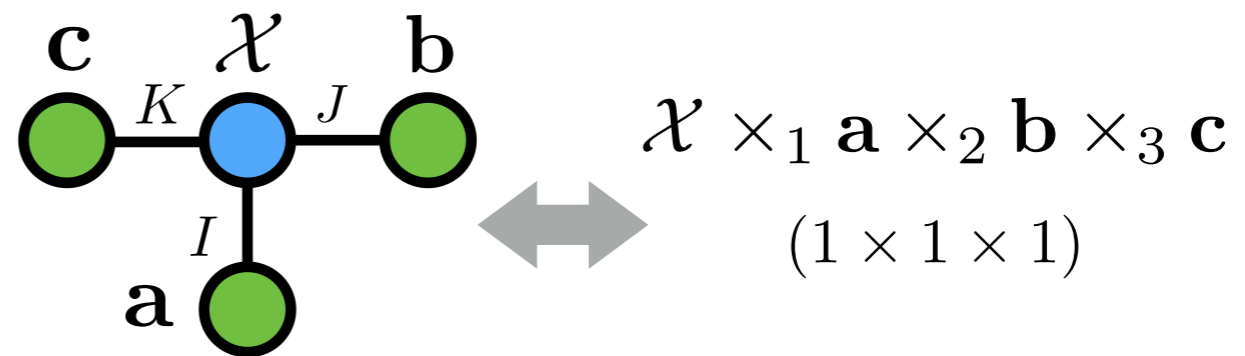
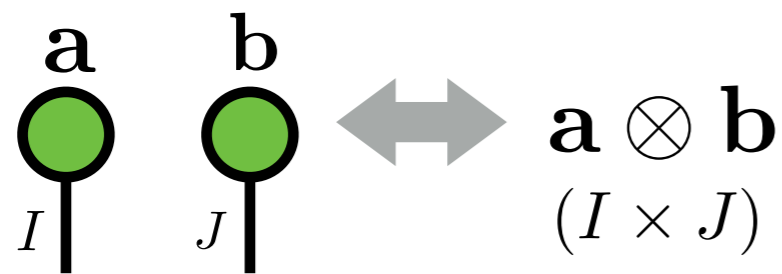
Inner product



Linear transform



Matrix multiplication

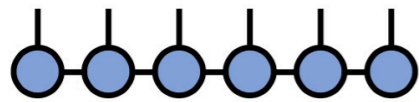


Node:=Tensor Edge:=Contraction Graph:=Tensor Network

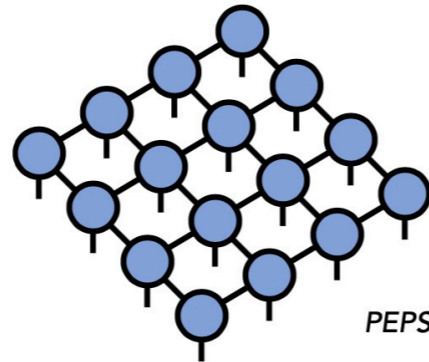
- ▶ The essence of the tensor network is multivariate polynomial.
- ▶ Tensor network decomposition = *Polynomial Approximation*

# Various Tensor Network Models

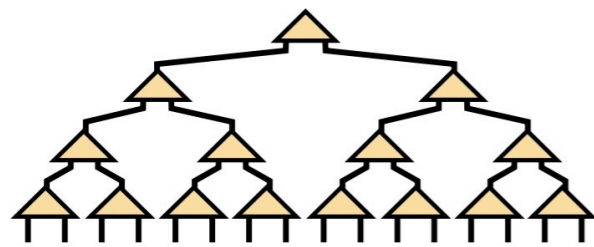
[Oseledets, 2011]



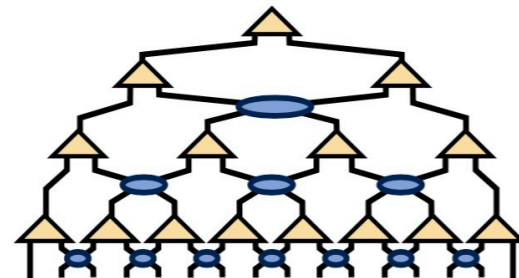
matrix product state /  
tensor train



PEPS network

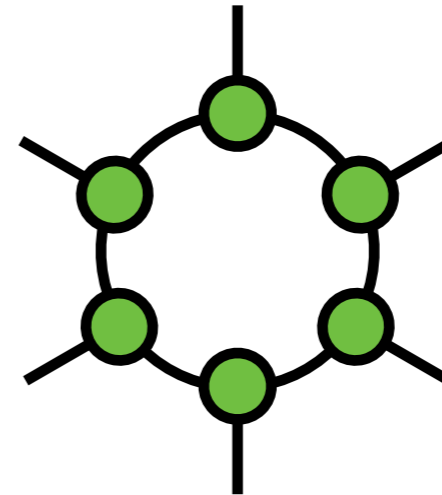


tree tensor network /  
hierarchical Tucker

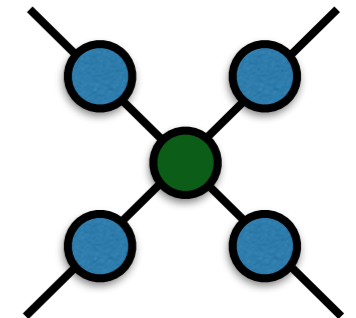


MERA network

R. Orus, Ann. of Phys. 349, 117–158 (2014)



[Zhao et al, 2016]



[Tucker, 1966]

- ▶ Given a task, which model is the best one?
  - ① How to select the best model in an affordable manner?
  - ② Is it possible to get models *going beyond* existing ones?



# Part 2: Structure Learning of TND

# Reforge the Diagram Notation

To build a bijection from an arbitrary simple graph to the TN structure.

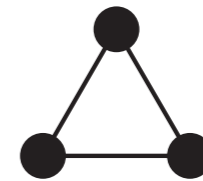
<i>Graph</i>	<i>Tensor network</i>
vertices	cores/core tensors/factors
edges	contracted indices
degree of vertex	number of indices in each core
weight of edge	upper limit of summation
number of edges	number of indices contracted



(a) matrix product



(b) tensor train (TT)



(c) tensor ring (TR)

Correspondence between Graphs and TN structures [Ye & Lim, 2019].

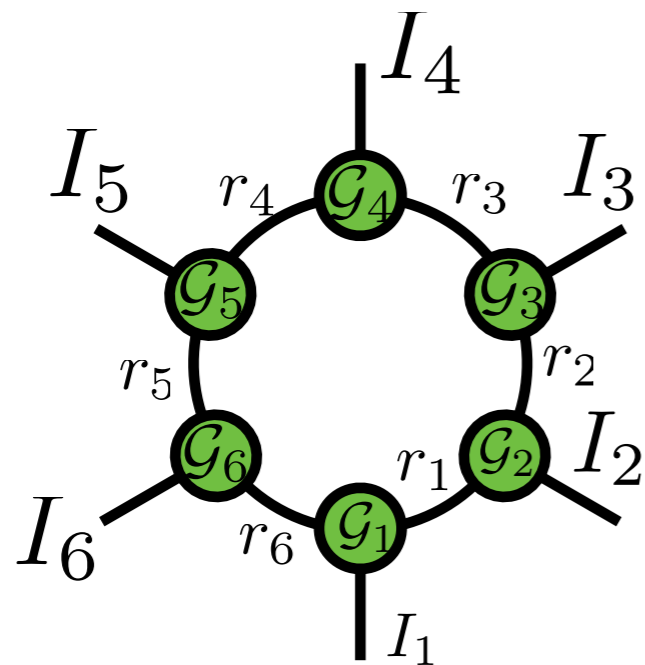
## Additional rules

- Weight-one edges are not allowed.
- The isolated sub-graphs are “connected” by tensor outer product.
- The vertices can be internal.

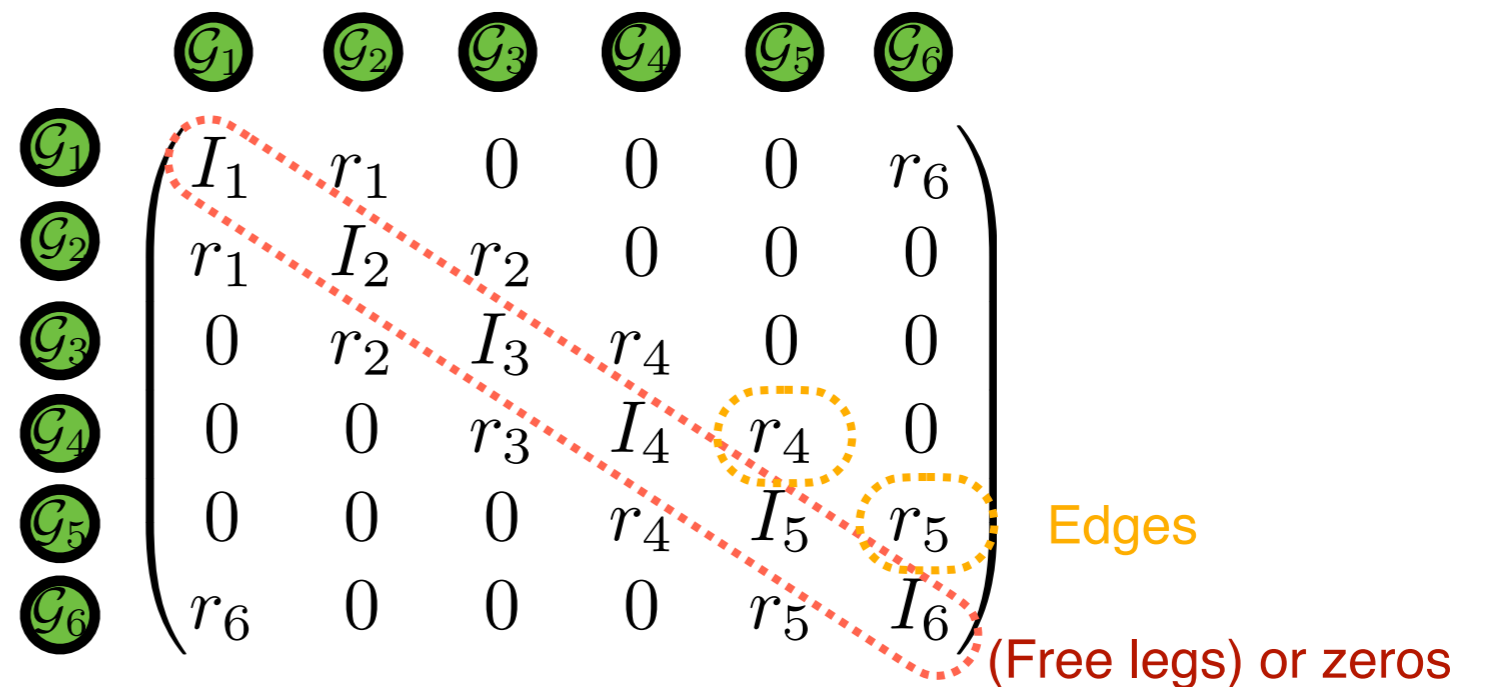
The TN structures can be fully described by its **adjacency matrices**.

# Unified Model Formulation

- ▶ (Augmented) *adjacency matrix*: a simple way to model tensor networks



order-6 Tensor Ring



(Augmented) Adjacency matrix

- ▶ Tensor networks (TN) are fully modeled by adjacency matrices.
- ▶ The matrix contains the both topological and algebraic information.

# Unified Model Formulation

**A general model** of TN representation with arbitrary graphical structure.

$$\mathcal{X} = TN(\mathbb{V}; \mathbf{A}), \quad (1)$$

where  $TN(\cdot; \mathbf{A})$  denotes the index contraction operations under the adjacency matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$  and  $\mathbb{V} := \{\mathcal{V}_i, i \in [N]\}$  denotes a collection of core tensors.

**Problem setting** - *structure learning* of tensor network (TN) decomposition

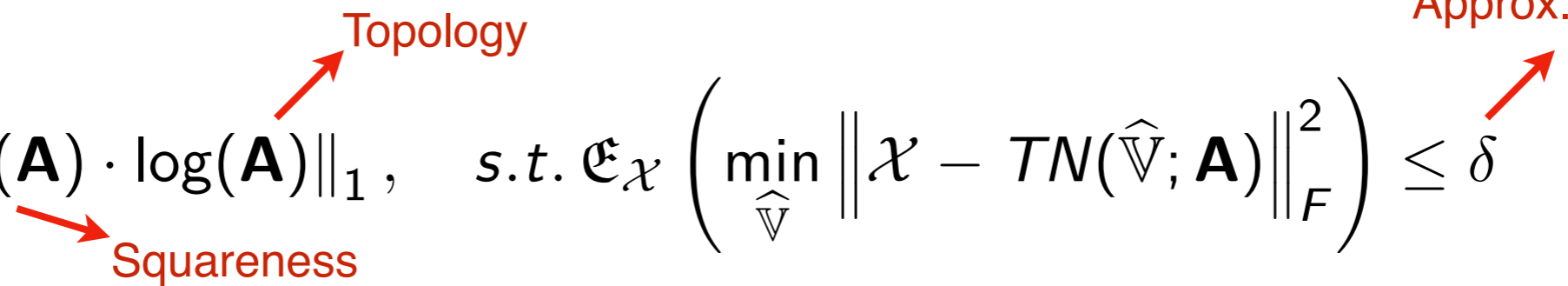
## Formal statement

Given a population of tensors  $\{\mathcal{X}\}$ , the structure learning of TN is to find the *optimal* adjacency matrix  $\mathbf{A}_0$ , such that for every  $\mathcal{X}_{(i)} \in \{\mathcal{X}\}$  there exists a set of tensors  $\hat{\mathbb{V}}_{(i)}$  that satisfies Eq. (1).

# Optimization Model

Optimization model

$$\min_{\mathbf{A} \in \mathbb{A}} \|D(\mathbf{A}) \cdot \log(\mathbf{A})\|_1, \quad s.t. \mathbb{E}_{\mathcal{X}} \left( \min_{\hat{\mathbf{V}}} \|\mathcal{X} - TN(\hat{\mathbf{V}}; \mathbf{A})\|_F^2 \right) \leq \delta \quad (2)$$



where  $D(\cdot)$  denotes filling the non-diagonal elements of a matrix by zeros,  $\log(\cdot)$  and  $\|\cdot\|$  denotes the element-wise logarithm and 1-norm of a matrix.

## Proposition (roughly)

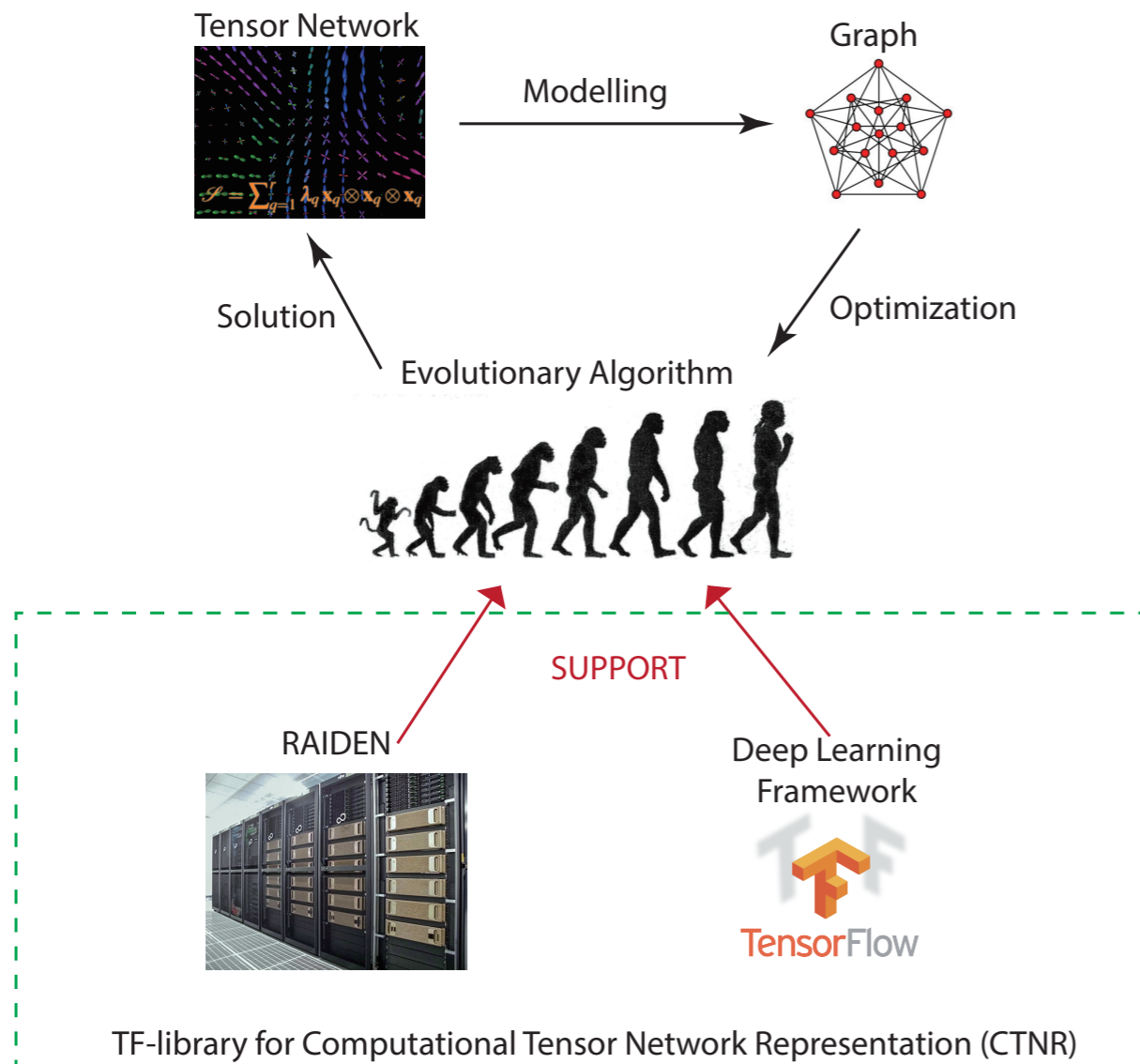
The logarithm of models' compression ratio is bounded by the objective function in Eq. (2) up to a fixed affine transform.

Integer programming is generally NP-complete.

Part 3:  
TN meets Genetic Algorithm



# Genetic Algorithm: An Efficient Heuristic



## Pros:

- ▶ Global optimization
- ▶ Multiobjective friendly
- ▶ Parallel computation

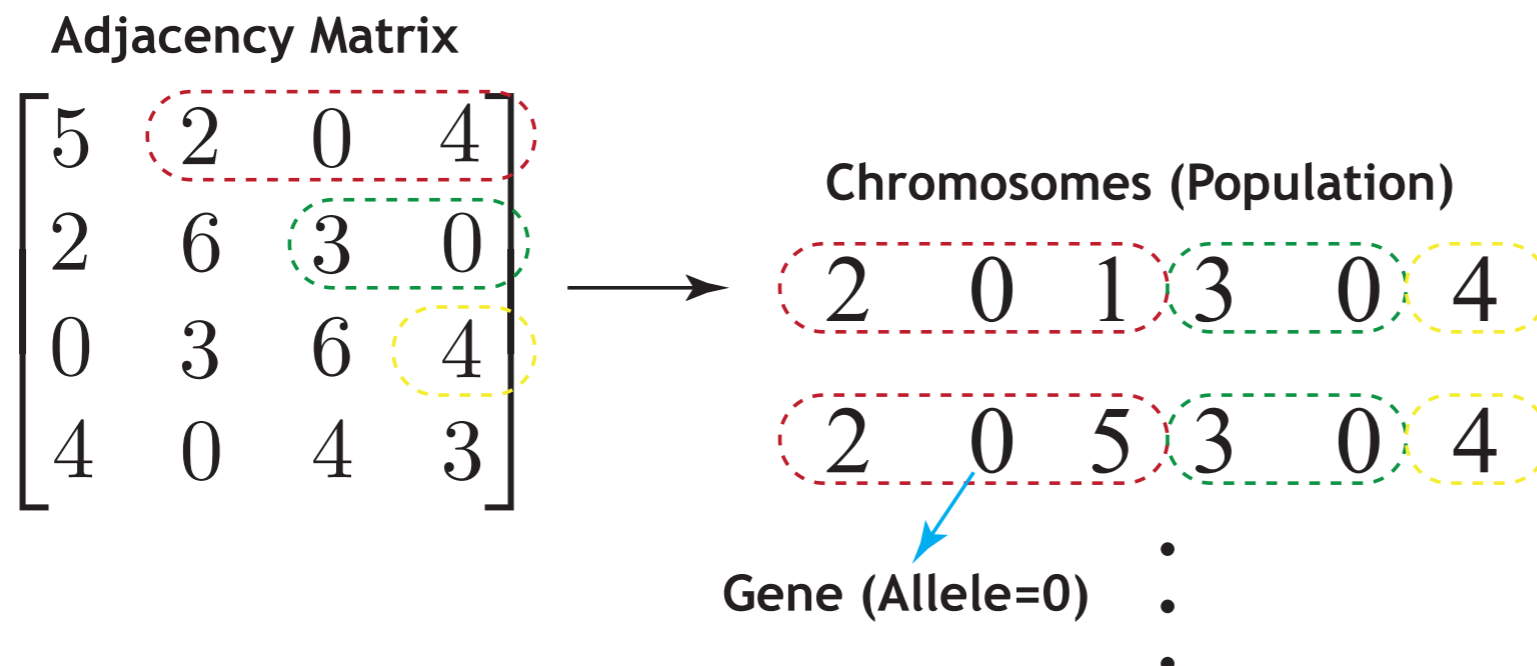
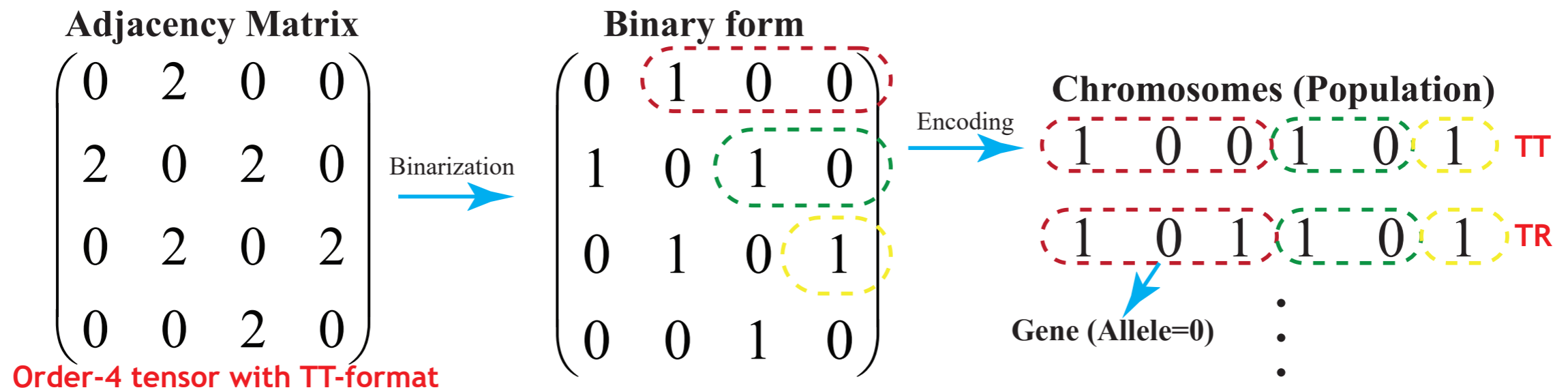
## Cons:

- ▶ Prohibitive computation
- ▶ No theoretical guarantee

The computational requirement is partially satisfied by the boost development of GPUs (HPC-Tensorflow).

# Structure Encoding

Encoding the Adjacency matrices into fixed length strings.



# Meta-algorithm

---

**Algorithm 1** GA-based meta-algorithm for topology learning of TN

---

**Input:** Tensor  $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_d}$ ; Number of of the vertices  $N \geq d$  and labeled vertices by the physical modes of  $\mathcal{X}$  or internal.

**1. Population Initialization.** //Each individual corresponds to different TN structure.

**Iteration until convergence:**

**2. Fitness evaluation** //Use accuracy and simplicity of the model to evaluate the graphs.

**3. Elimination** //Remove the graphs with bad fitness.

**4. Recombination** //Keep the structures of the superior graphs into the next generation.

**5. Mutation** //Slightly change each graph for the diversity of the solutions.

---

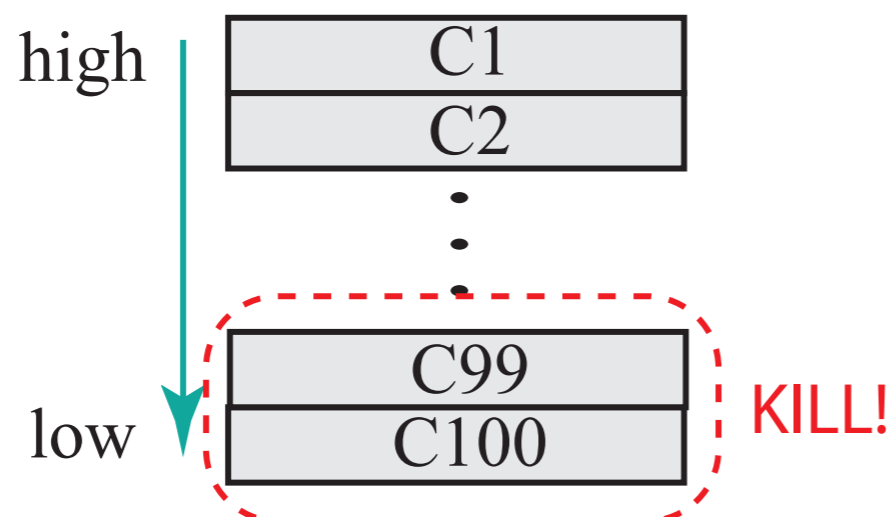
# Genetic Operators: Fitness and Elimination

## 1. Fitness score:

$$F(\hat{\mathbf{A}}) = \left\| D(\hat{\mathbf{A}}) \cdot \log(\hat{\mathbf{A}}) \right\|_1 + \lambda \cdot \underbrace{\min_{\hat{\mathbf{V}}} \left\| \mathcal{X} - TN(\hat{\mathbf{V}}; \hat{\mathbf{A}}) \right\|_F^2 / \|\mathcal{X}\|_F^2}_{\text{relative square error (RSE)}}, \quad (3)$$

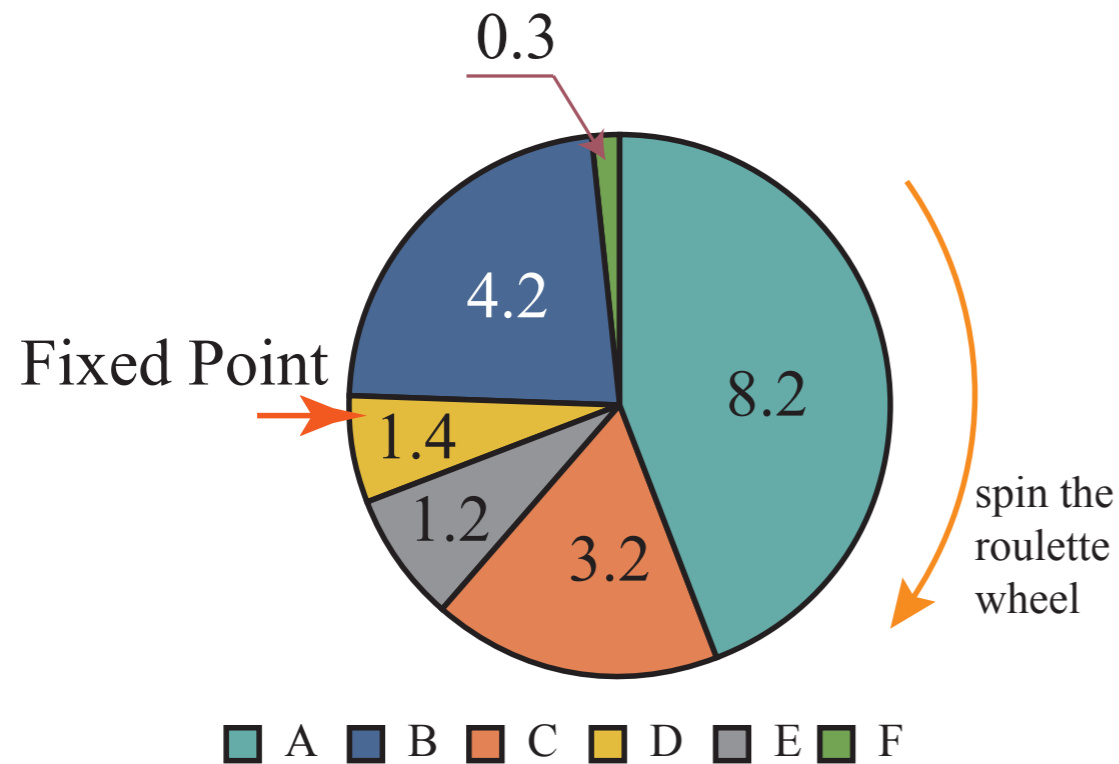
The fitness scores are used to rank the individuals.

## 2. Elimination



# Genetic Operators: Recombination

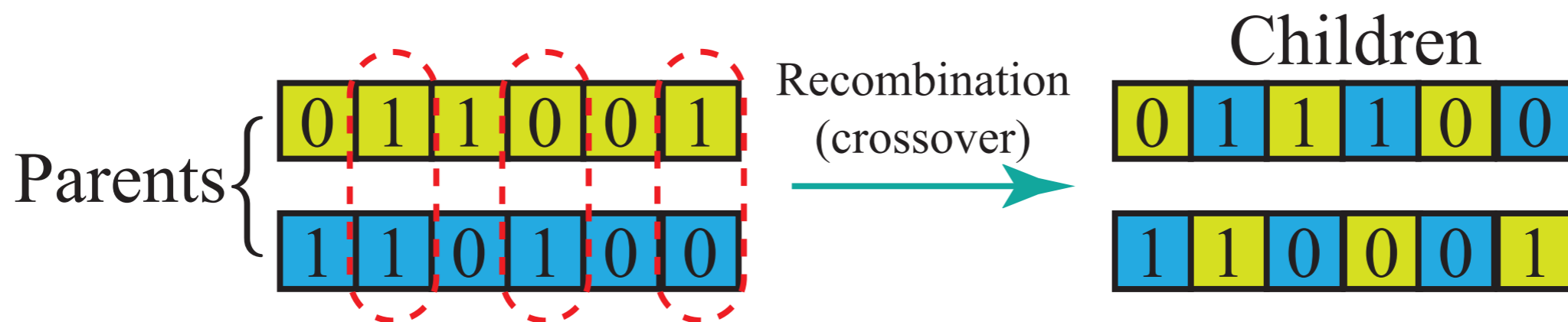
## 3.1 Parent selection (Russian roulette process)



$$Pr = \max \left\{ 0.01, \ln \left( \frac{\alpha}{eps + \beta \cdot rank} \right) \right\}$$

where  $\alpha, \beta > 0$  denote tuning parameters.

## 3.2 Crossover



# Genetic Operators: Mutation

## 4. Mutation



## Discussion:

- ▶ Mutation is the key operator to obtain the global minimum.
- ▶ The algorithm design follows the maximum entropy principle.
- ▶ More cost results in better performance.

# Tensor Decomposition on Synthetic Data

The synthetic data are randomly (Gaussian,  $\sigma = 0.1$ ) generated with random graph (Erods-Renyi model  $G(n, p)$ ,  $n = 4, \dots, 8$ ,  $p \in [0.15, 0.85]$ ).

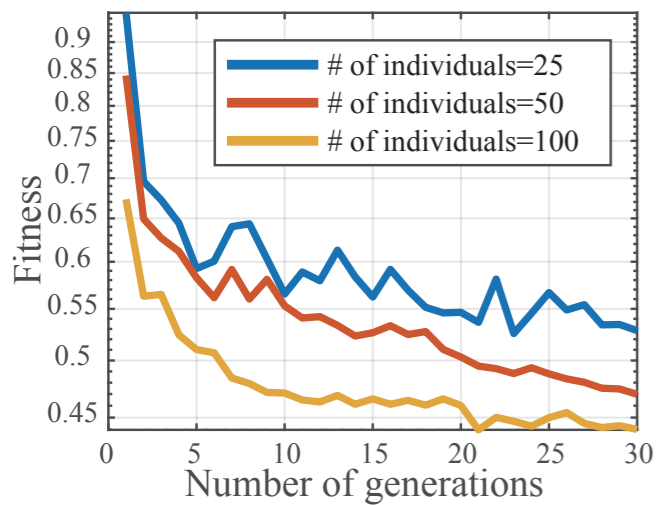
**Gen.**↓ – The number of generations we obtain the results.

**Eff.**↑ – Parameter ratio compared to the ground-truth.

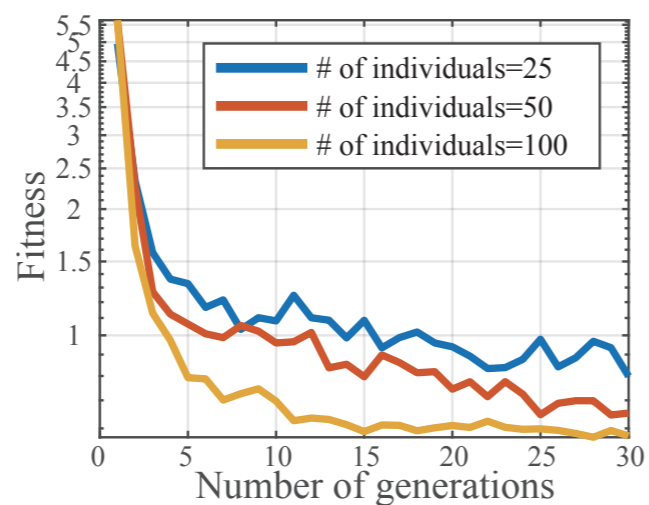
Trial	Order									
	4	4	5	5	6	6	7	7	8	8
	Gen.	Eff.	Gen.	Eff.	Gen.	Eff.	Gen.	Eff.	Gen.	Eff.
A	001	1.333	003	1.084	005	1.052	003	1.052	001	1.396
B	001	1.500	003	1.690	006	1.062	008	1.096	005	1.000
C	001	1.500	002	1.000	004	1.000	003	1.000	001	1.320
D	001	1.000	001	1.445	003	1.000	001	1.000	020	1.000
E	001	1.000	002	1.000	005	1.052	003	1.000	005	1.000

The proposed method obtains the same (**Eff.**= 1.000) or **even better** (**Eff.**> 1.000) compression ratio compared to the ground-truth.

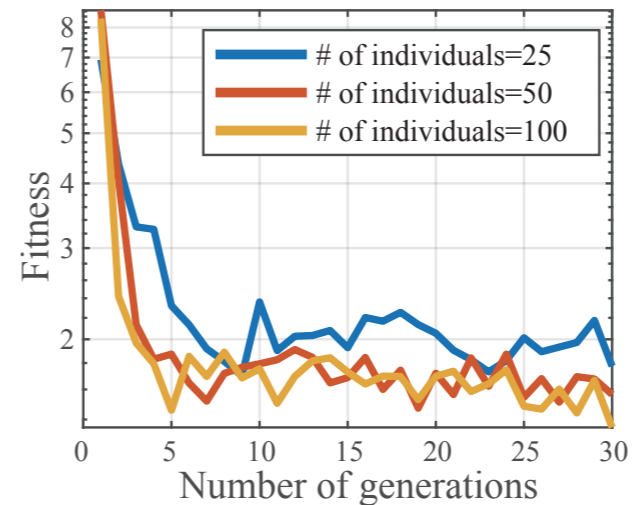
# Convergence



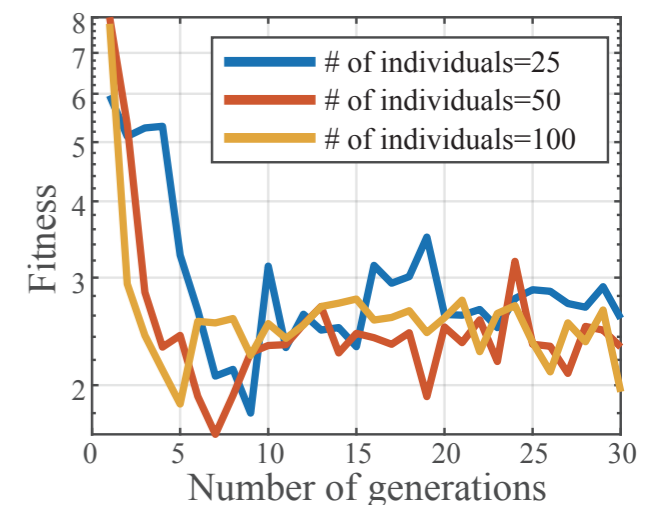
(a) min



(b) median

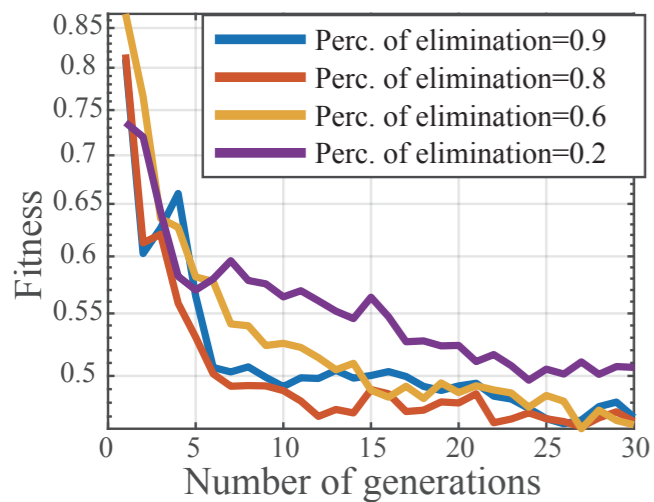


(c) mean

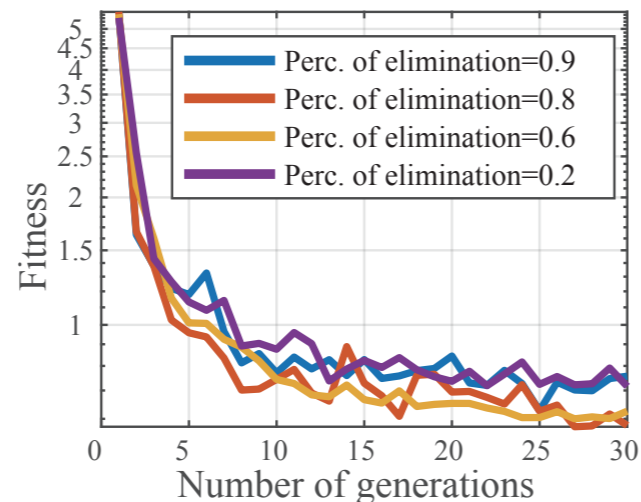


(d) std

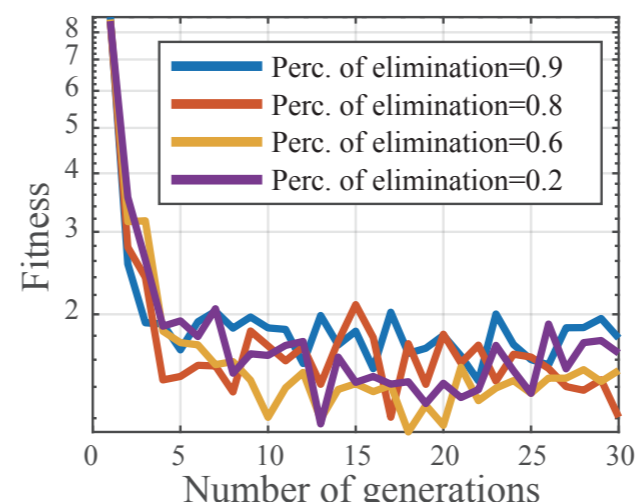
## Convergence with various number of individuals



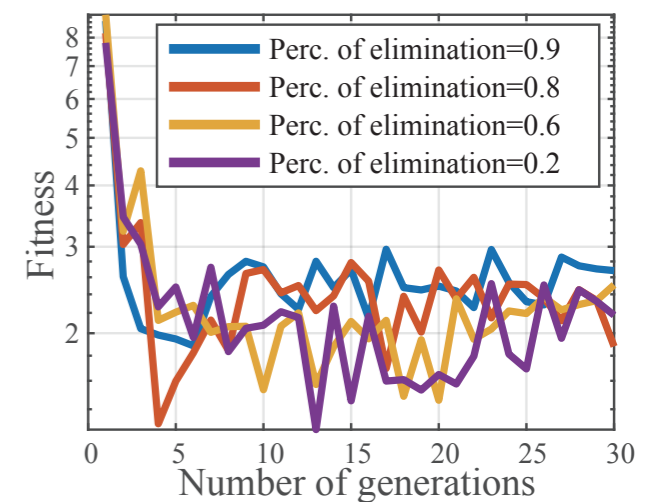
(a) min



(b) median



(c) mean



(d) std

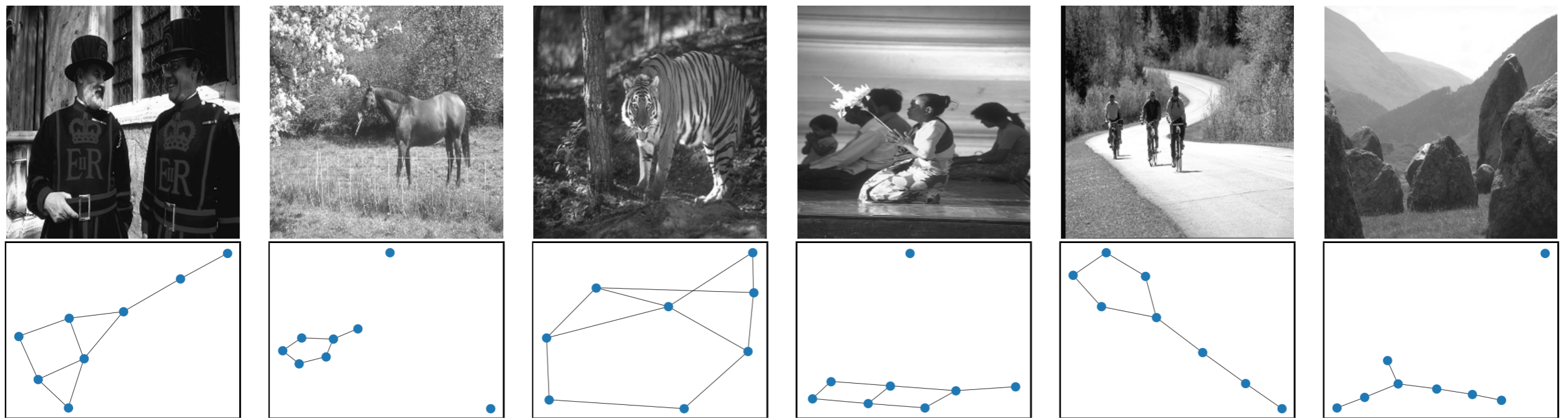
## Convergence with various elimination percentage

Trade off between “elitism” and “populism”.



# TN Decomposition on Natural Images

10 natural images are randomly selected from LIVE dataset [Sheikh et al., 2006], and reshaped as order-8 tensors.



The learned structures are more complex than path, tree or cyclic graphs

# Discussion and Concluding Remarks

Computationally efficient? **No, and yes.**

- ▶ Additional computation requirement is unavoidable.
- ▶ Lots of near-optimal structures outperforms the existing models.

Potential applications? **Model compression.**

Takeaway messages and future work? **Small-world network.**

- ▶ The learned topology is far away from the well-developed models
- ▶ Intuitively the results are more close to Watts–Strogatz graphs