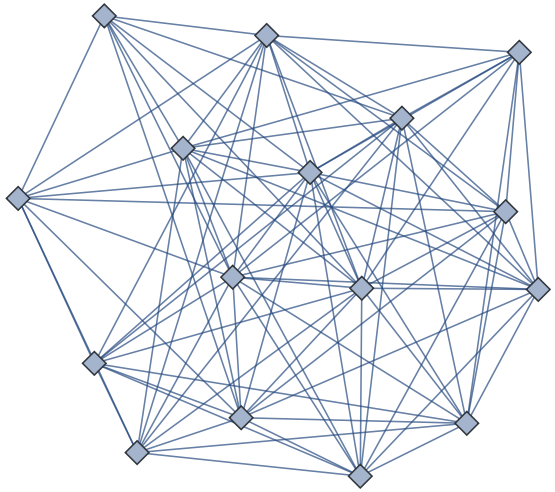
Algorithms for Tensor Network Contraction Ordering

Frank Schindler  University of Zurich[UZH]

Adam Jermyn 

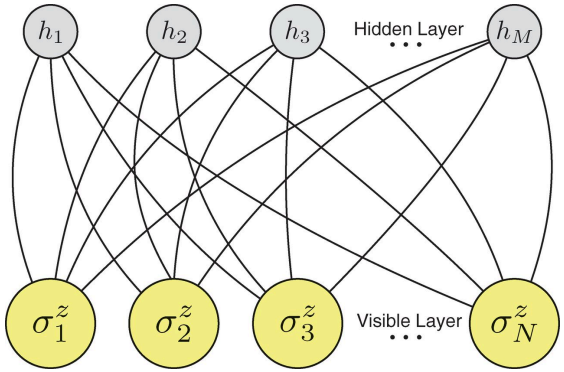
Benasque, February 17th 2020



Motivation: Machine Learning in Physics

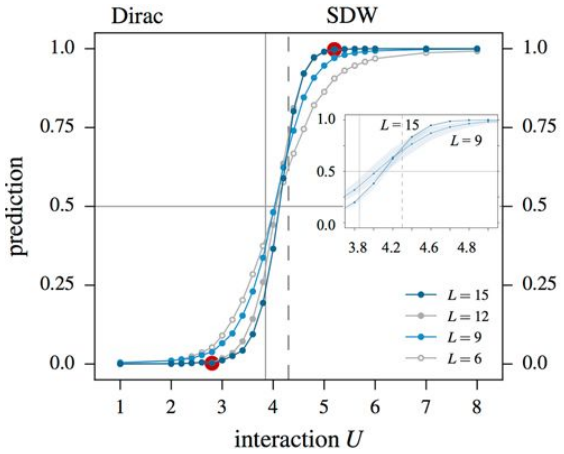
Previously: Use ML models (neural networks, …) as models of nature

ML models as variational wavefunctions

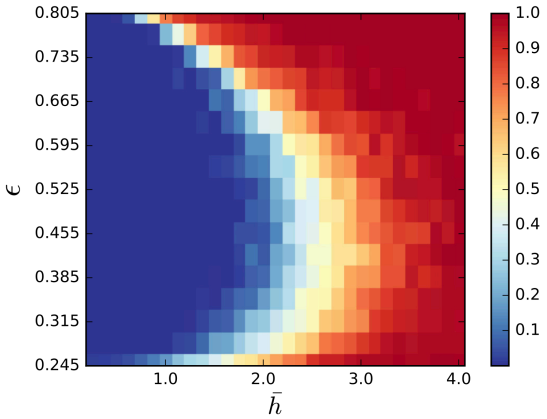


Restricted Boltzmann Machines
Carleo & Troyer, Science (2017)

ML models as phase classifiers



Many-body systems w/ sign problem
Broecker, Carrasquilla, Melko & Trebst, Scientific Reports (2017)



Many-body localization
Schindler, Regnault & Neupert, PRB (2017)

— and many more —

These were good exploratory proof of concepts, but we need to move on!

physically motivated, interpretable, controlled models of nature

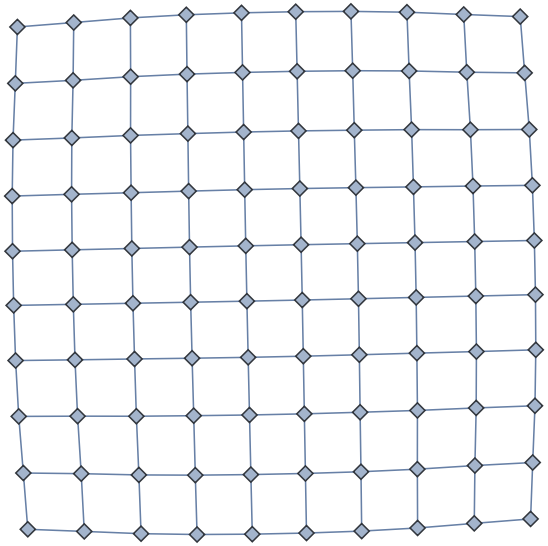
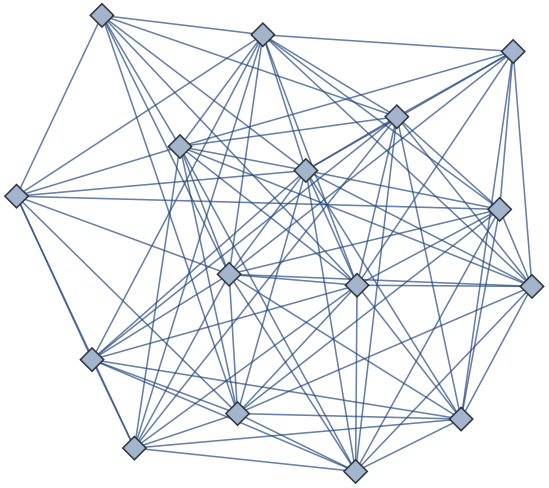
replace by

use to find and improve

machine learning algorithms

Tensor Network Contraction

Tensor networks are physically motivated, interpretable, controlled variational wavefunctions

$\Psi =$  , 

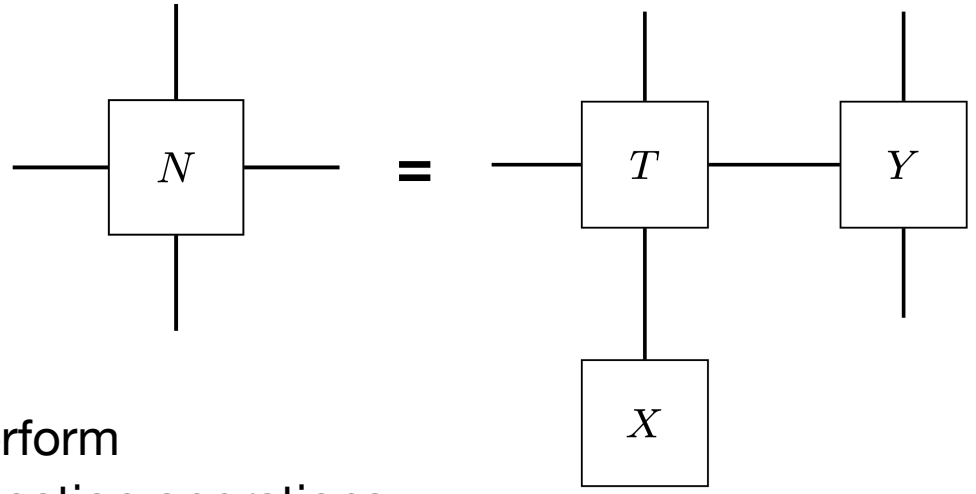
PEPS **Random Graphs**

(1) contracting TNs exactly without special structure is exponentially costly in D > 1
many successful approximation schemes (MPOs, CTM, TRG, TNR, …)

(2) finding exact optimal contraction sequences is exponentially costly in D > 1
NETCON algorithm: *Pfeifer, Haegeman & Verstraete, PRE (2014)*
Are there good approximation schemes?

Tensor contraction involves simultaneous sums:

$$N_{klmn} = \sum_{ij}^{\chi} T_{ijkl} X_i Y_{jmn} :$$



Let $\chi = 2$: Then naively need to perform
(1+2)*2^6 = **192** addition and multiplication operations

There are two ways to split up the sum:

$$N_{klmn} = \sum_{i}^{\chi} Q'_{iklmn} X_i,$$

$$Q'_{iklmn} = \sum_{j}^{\chi} T_{ijkl} Y_{jmn} .$$

128 + 64 = **192** addition and multiplication operations

$$N_{klmn} = \sum_{j}^{\chi} Q_{jkl} Y_{jmn},$$

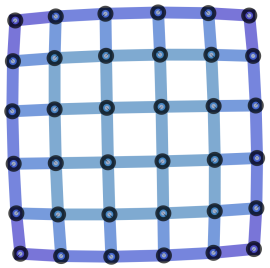
$$Q_{jkl} = \sum_{i}^{\chi} T_{ijkl} X_i .$$

32 + 64 = **96** addition and multiplication operations

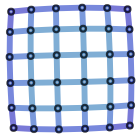
order of summation matters!

Naive approximation: **Greedy algorithm**
always contracts the cheapest bond

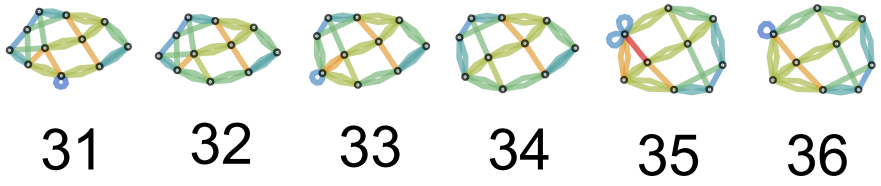
consider a 6x6 square tensor network (PEPS)



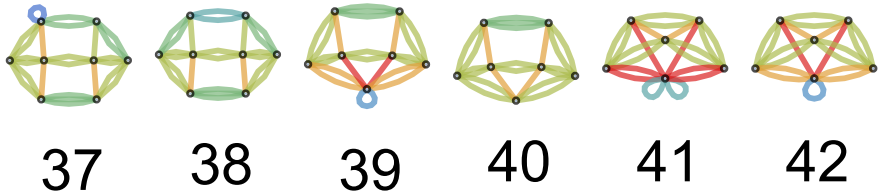
greedy sequence:



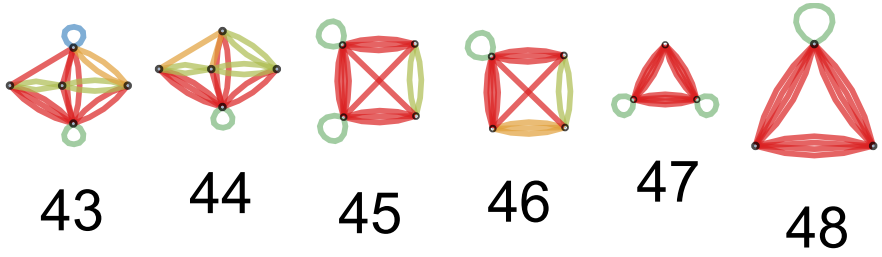
1



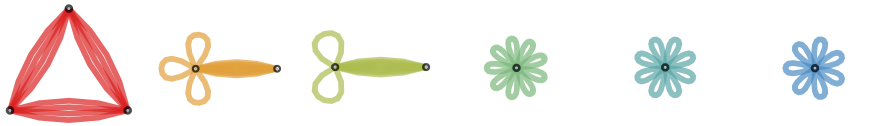
31 32 33 34 35 36



37 38 39 40 41 42

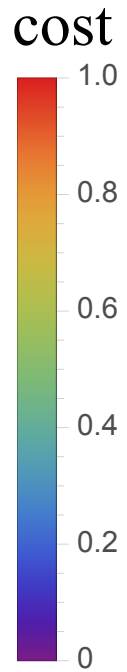


43 44 45 46 47 48



49 50 51 52 53 54

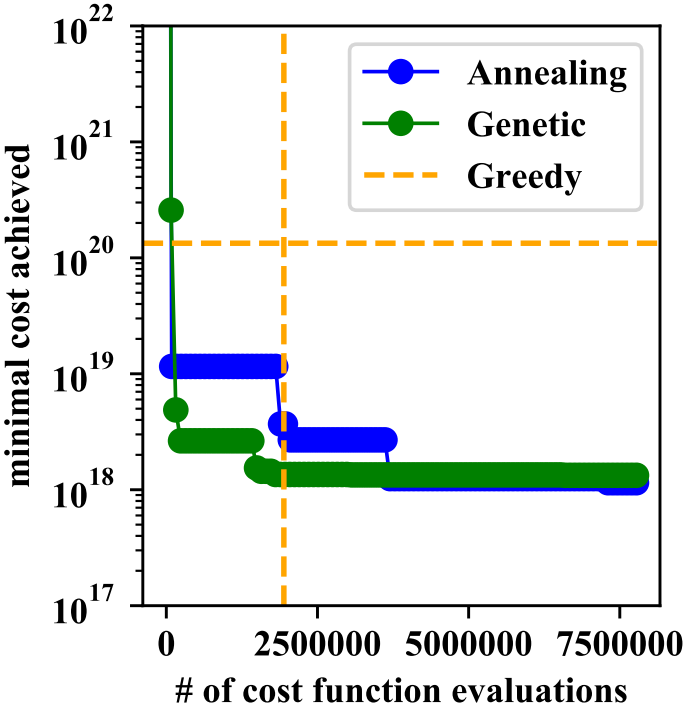
55 56 57 58 59 60



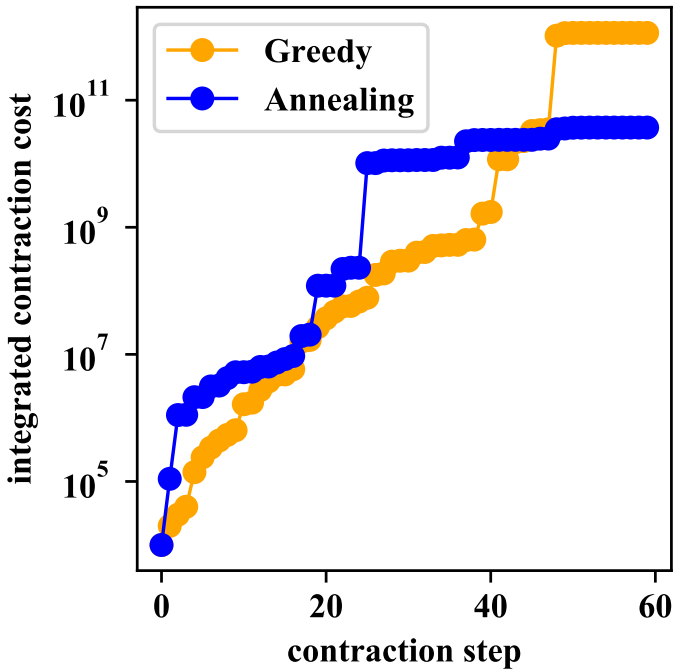
Can we do better than greedy?

Simulated Annealing:
global, Monte-Carlo-inspired search

Genetic Algorithm:
global, biological evolution-inspired search

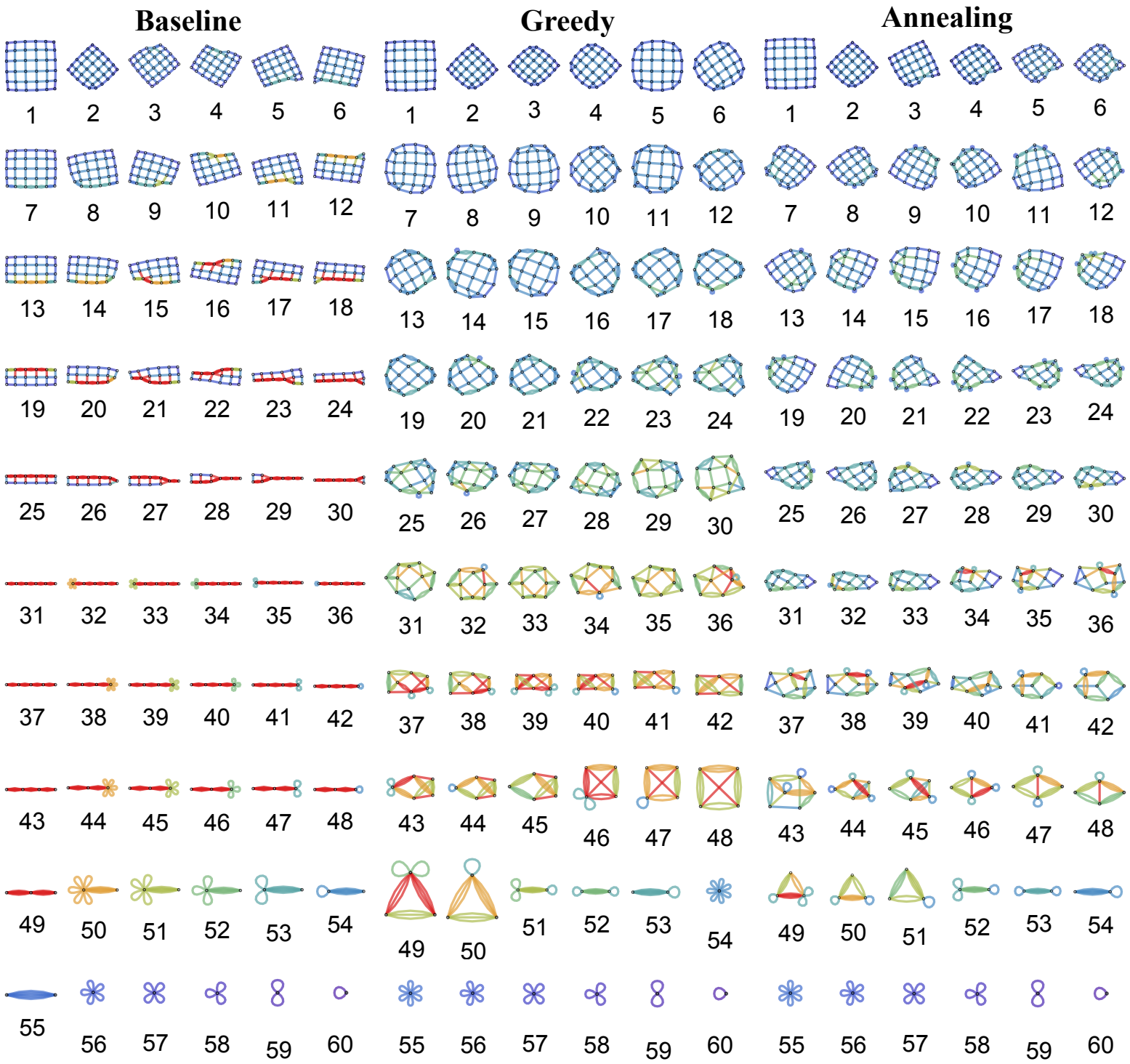


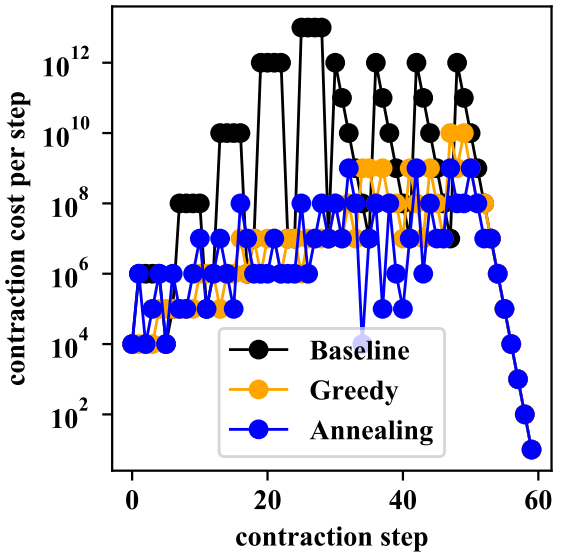
both algorithms often beat Greedy using less runtime
(plot: 10x10 PEPS, $\chi = 10$)

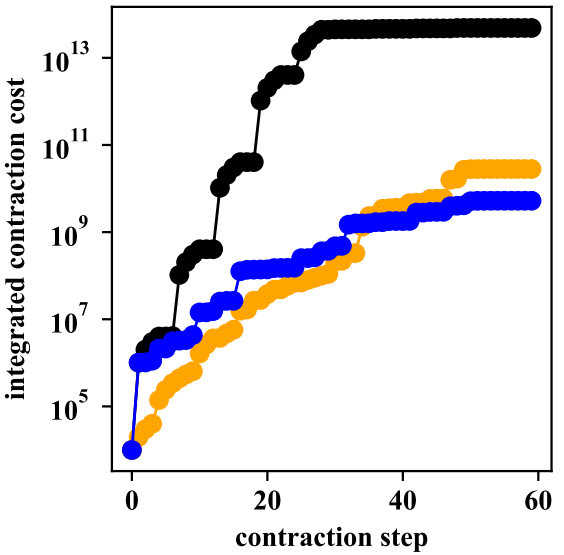


their contraction sequences involve seemingly suboptimal early contractions that avoid high costs later
(plot: 6x6 PEPS, $\chi = 10$)

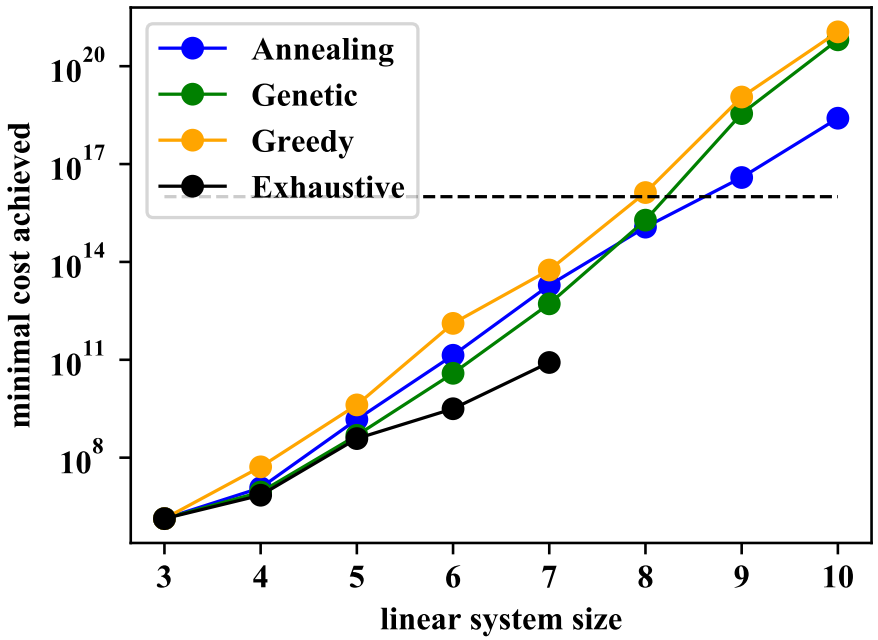
6x6 PEPS contraction Hall of Fame

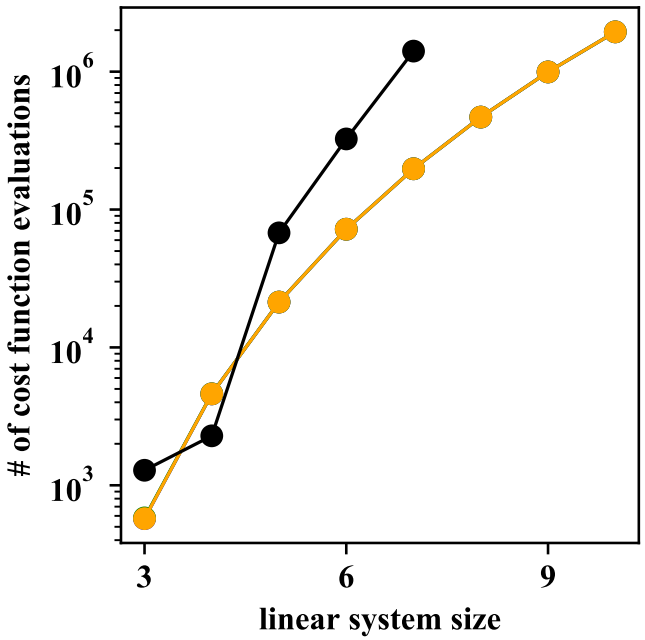


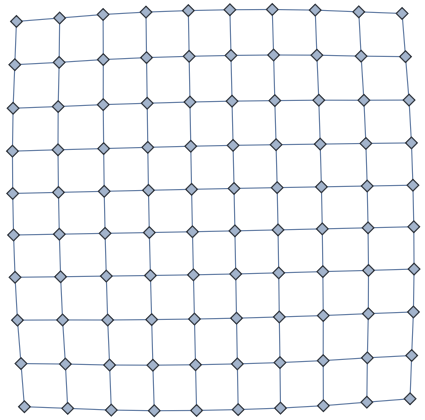




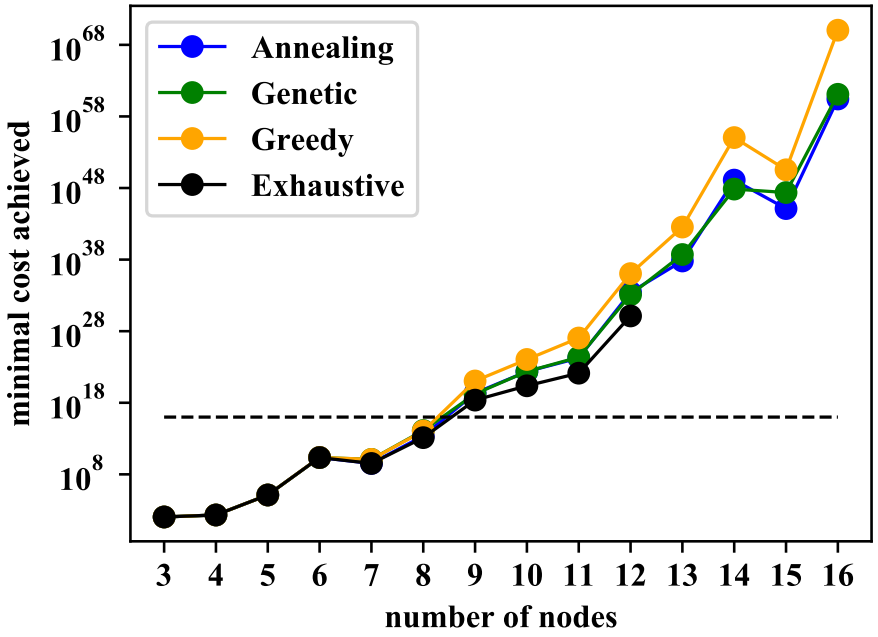
System Size Scaling for PEPS

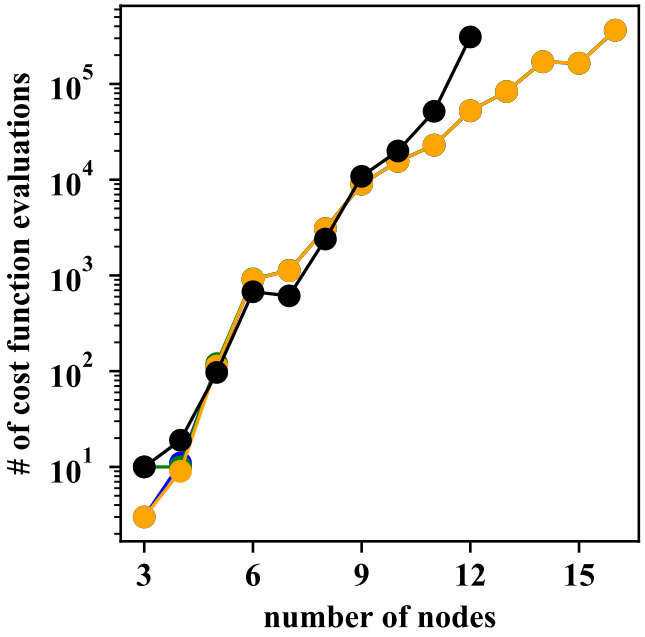


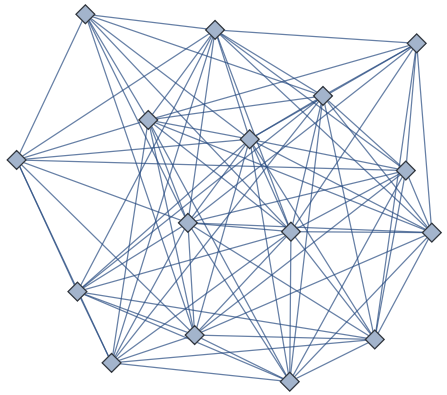




System Size Scaling for Random Graphs







That's it!

for more, check out
arXiv:2001.08063

or find the source code on
https://github.com/frankschindler/OptimizedTensorContraction