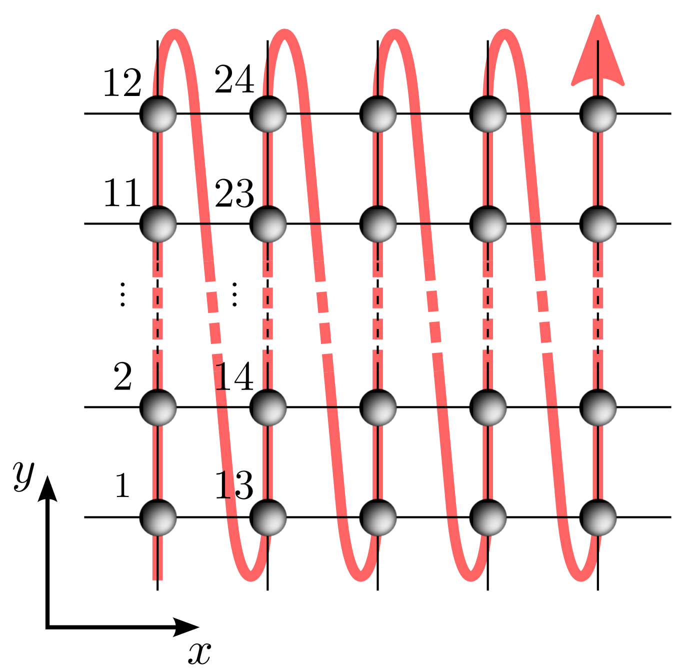
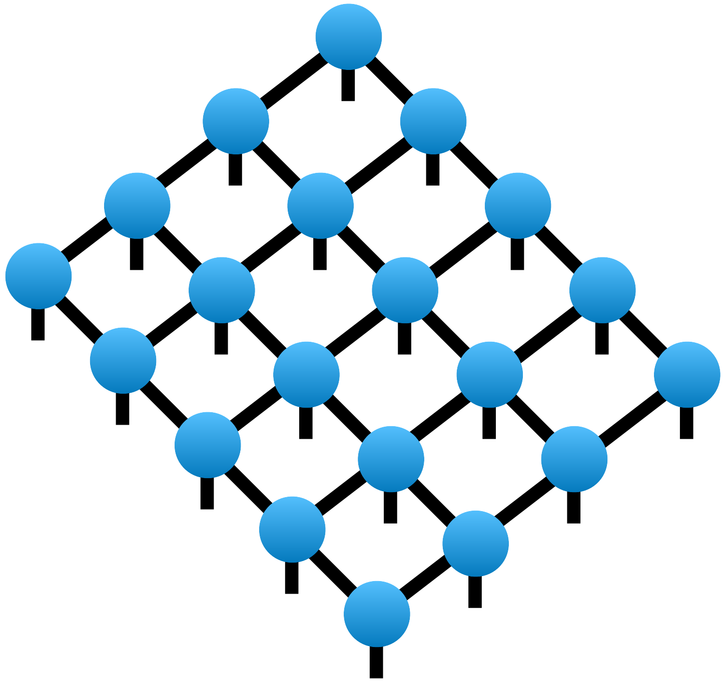
DMRG: 2D Systems and Advanced Topics





E.M. Stoudenmire

Oct 2019 - Pohang

FLATIRON INSTITUTE
SIMONS FOUNDATION

Plan of the Talk

DMRG for two-dimensional (2D) systems

2D DMRG with ITensor

Applications of 2D DMRG

Advanced topics: quantum numbers, fermions

Tomorrow

Introduction to Machine Learning, Science Applications

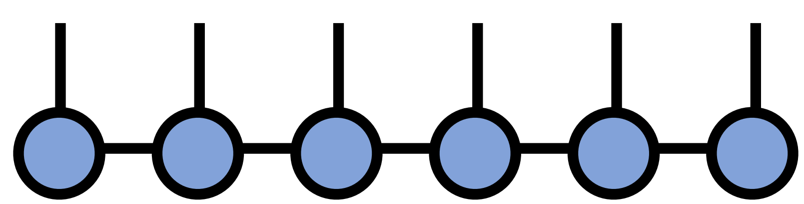
Machine Learning with Tensor Networks

Brief Review of DMRG

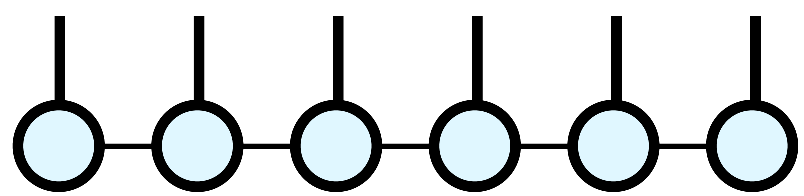
DMRG is an algorithm for finding ground states as MPS tensor networks:

$$\hat{H} = \sum_{ij} (S_i^+ S_j^- + S_i^- S_j^+) + \Delta S_i^z S_j^z$$

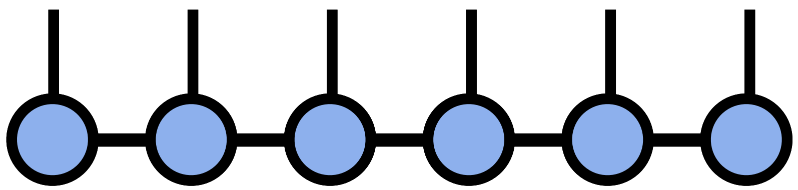
DMRG

$\Psi_0 =$ 

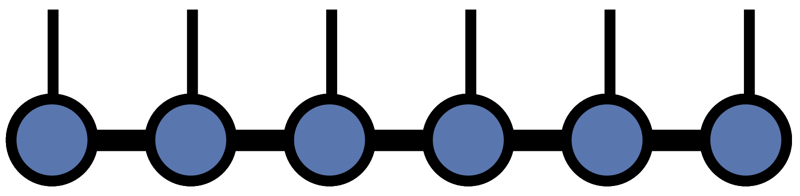
Works by "sweeping" over pairs or tensors from one side to the other



Works by "sweeping" over pairs or tensors from one side to the other

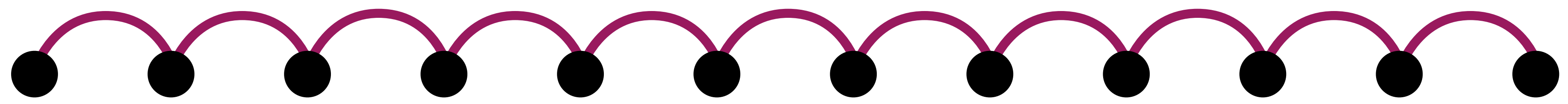


Works by "sweeping" over pairs or tensors from one side to the other



Nominally for 1D systems

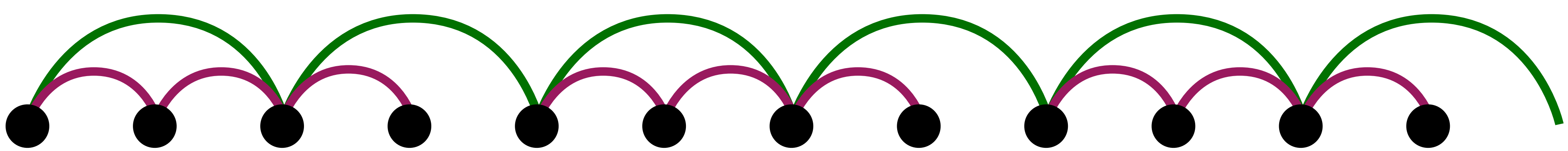
Nominally for 1D systems



Yet ok to have irregular, longer-range interactions

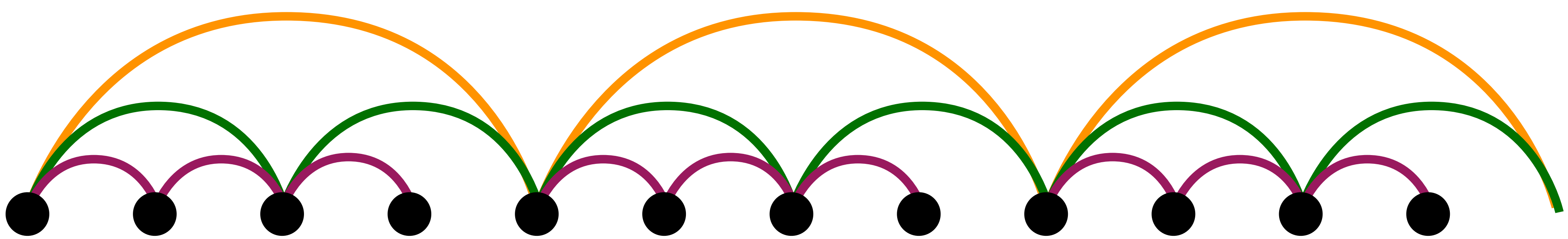
Nominally for 1D systems

Yet ok to have irregular, longer-range interactions

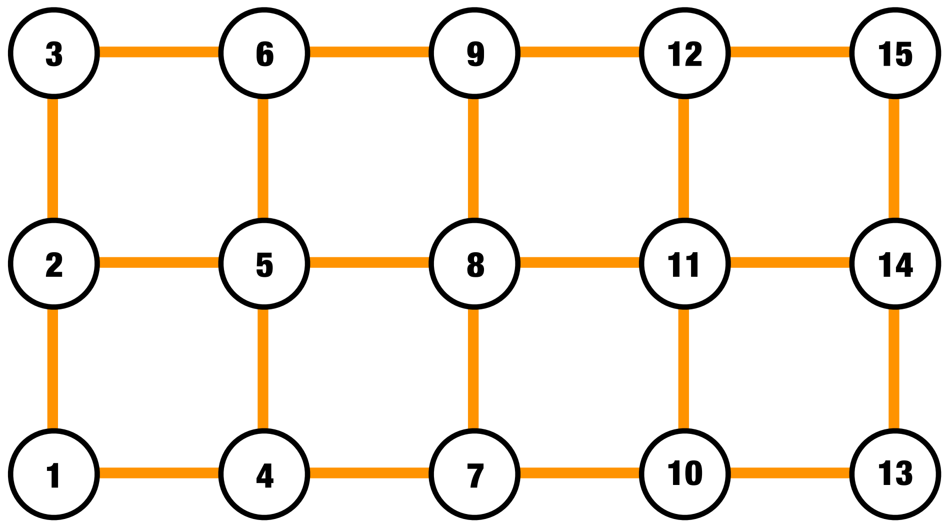


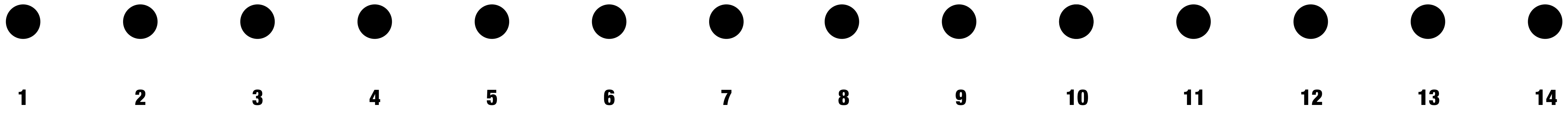
Nominally for 1D systems

Yet ok to have irregular, longer-range interactions

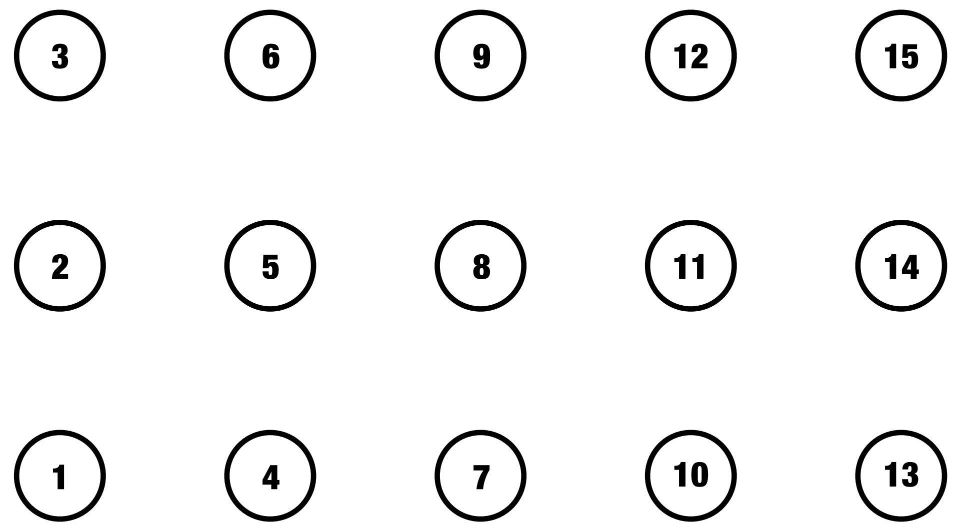


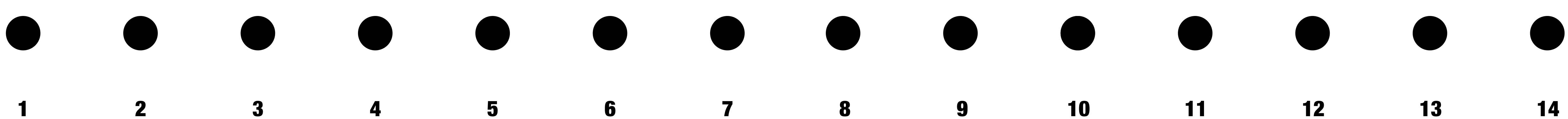
Trick to use a 2D Hamiltonian:



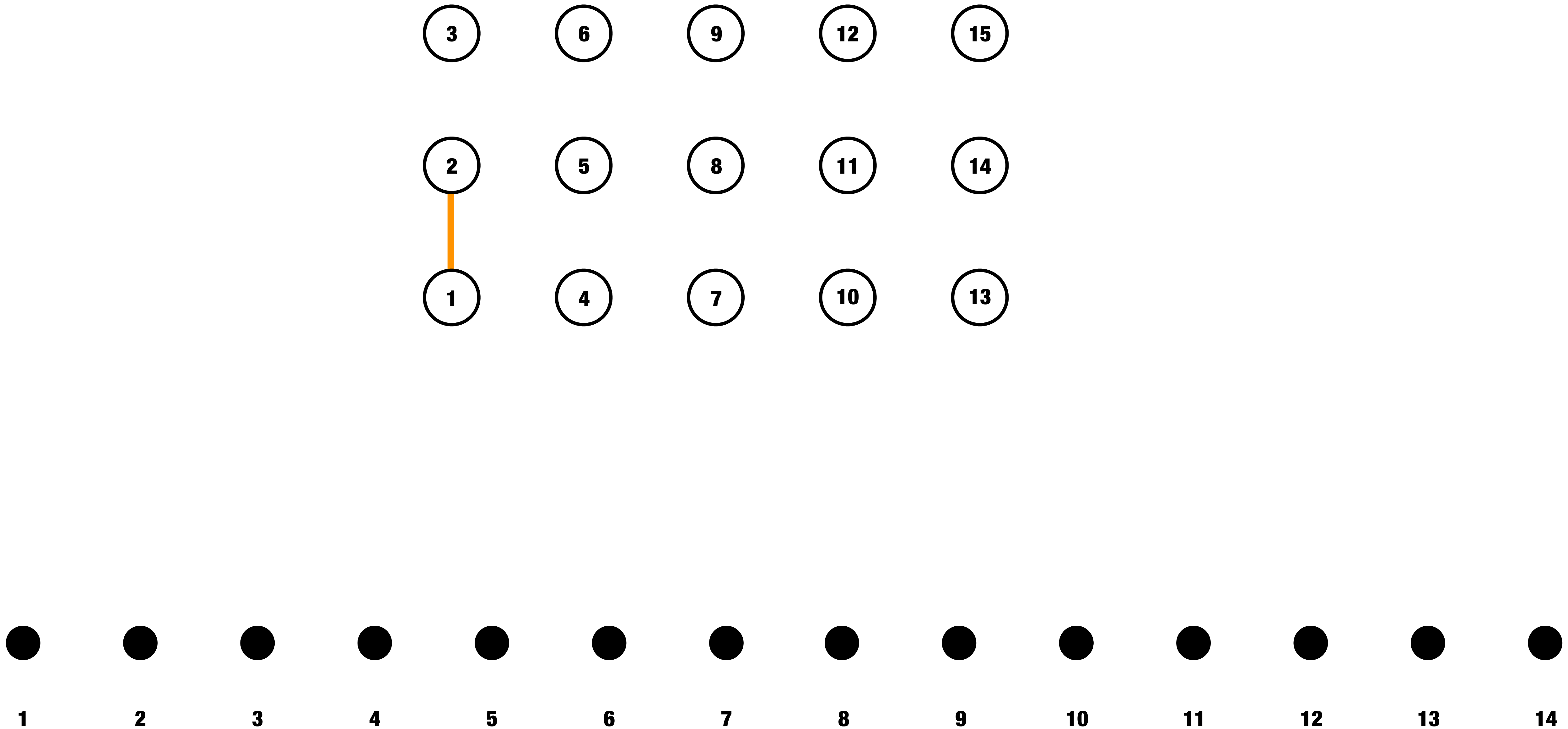


Trick to use a 2D Hamiltonian:

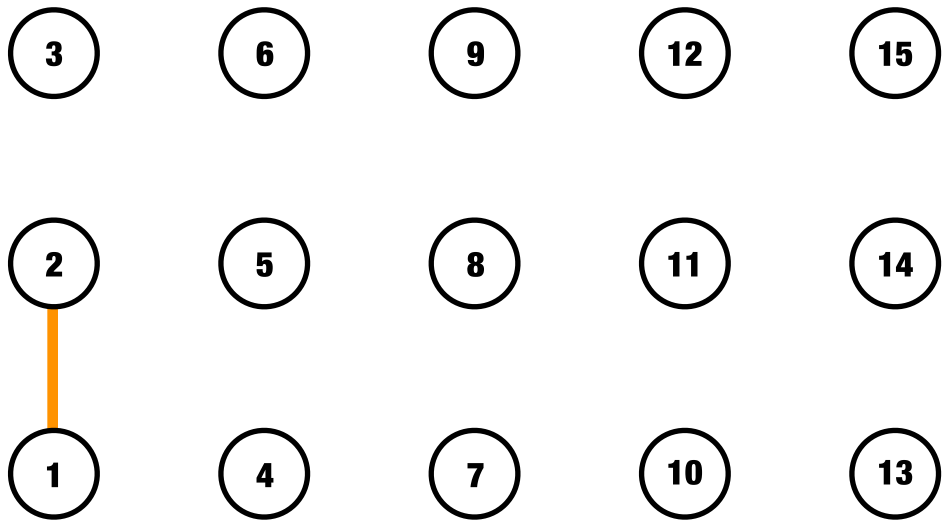


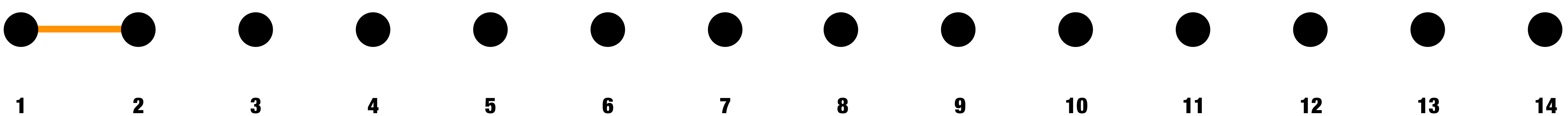


Trick to use a 2D Hamiltonian:

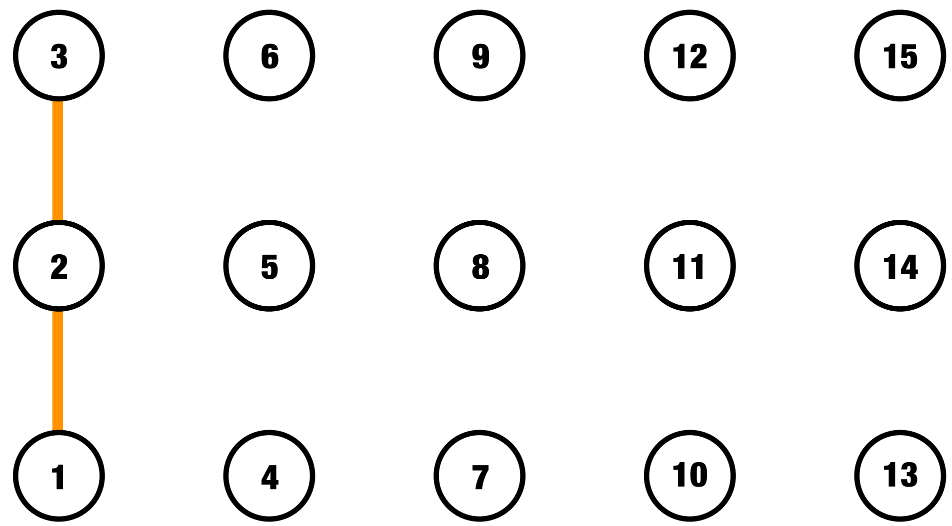


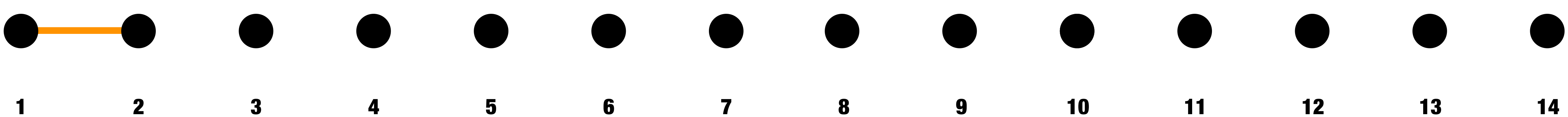
Trick to use a 2D Hamiltonian:



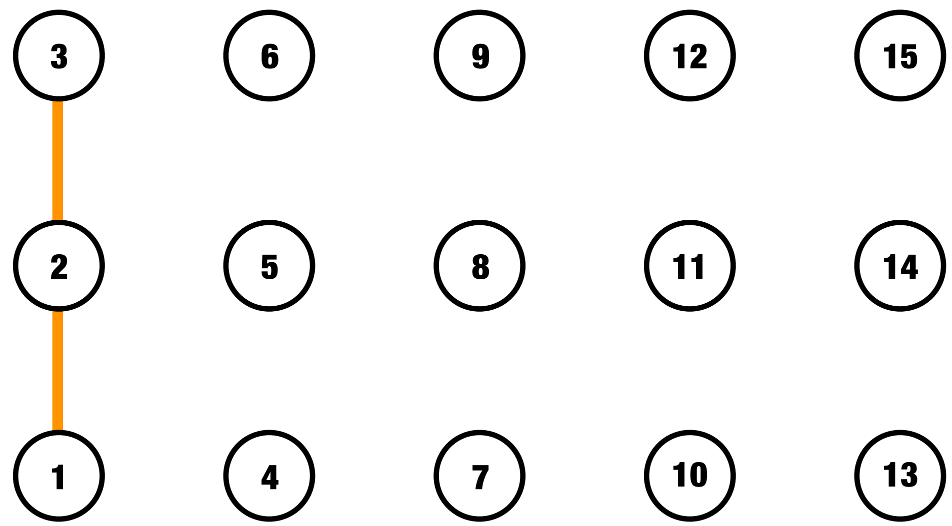


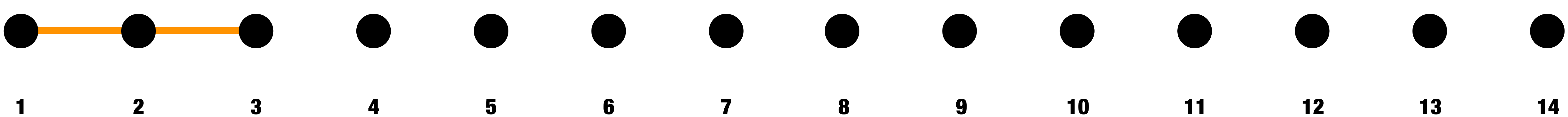
Trick to use a 2D Hamiltonian:



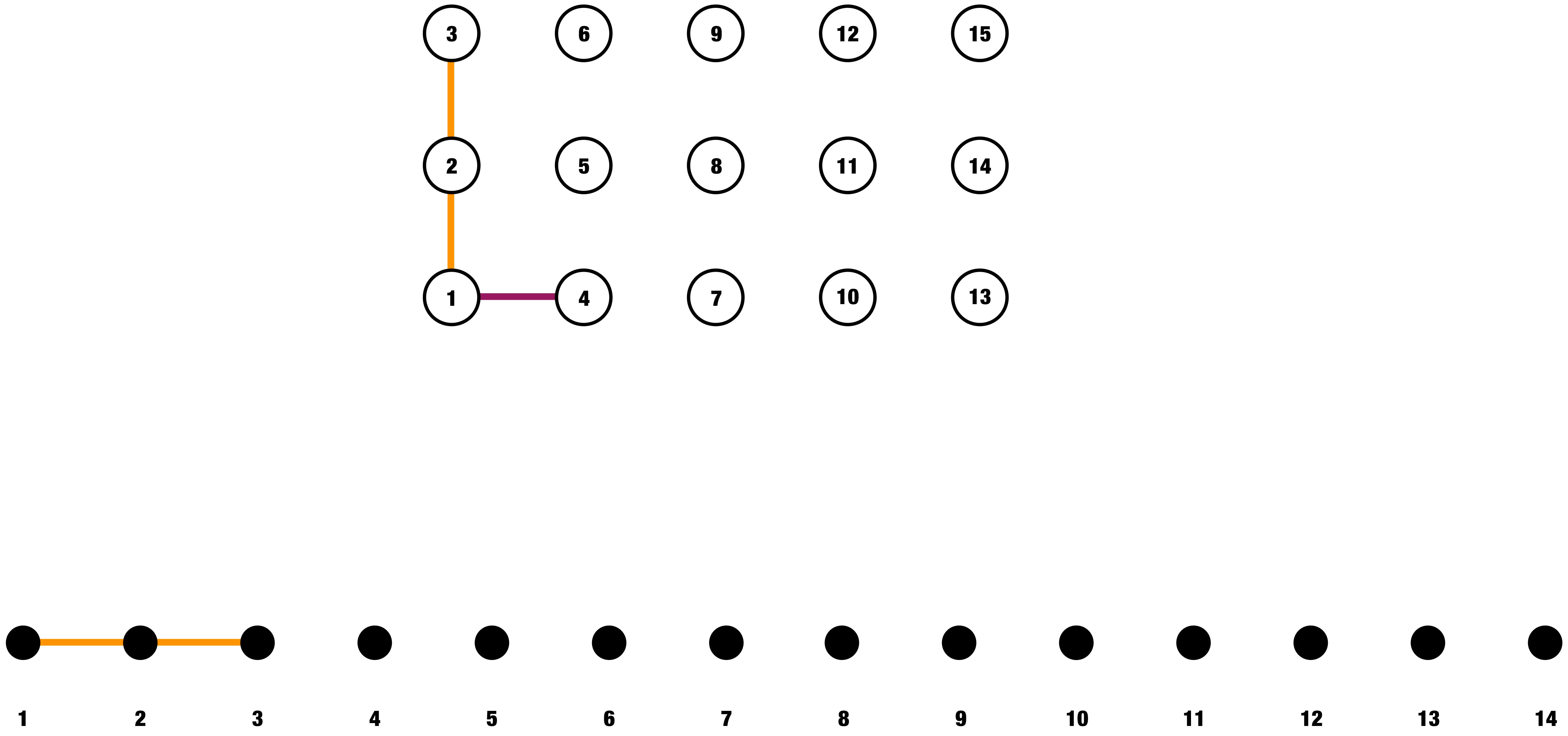


Trick to use a 2D Hamiltonian:

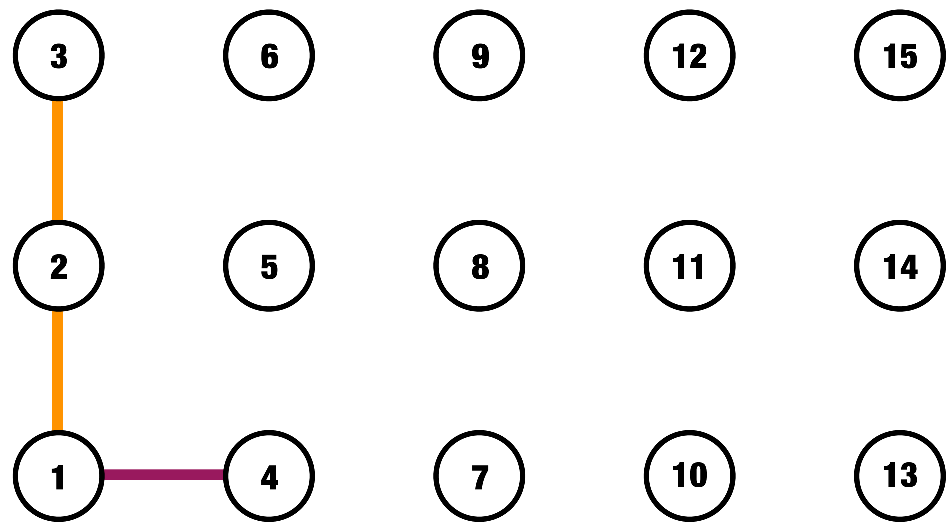


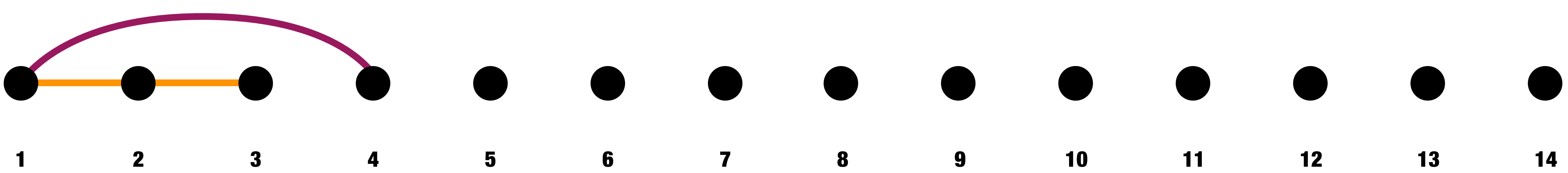


Trick to use a 2D Hamiltonian:

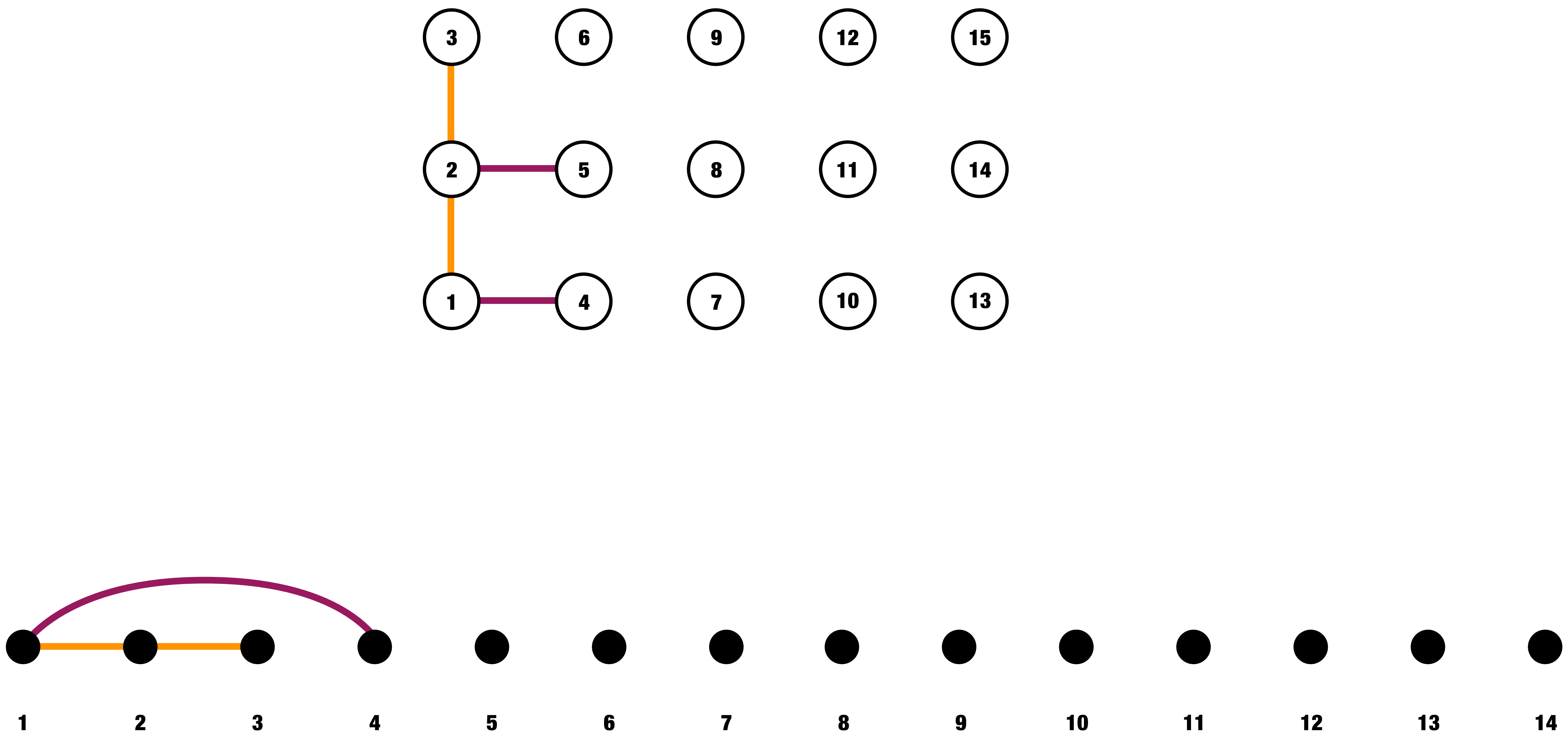


Trick to use a 2D Hamiltonian:

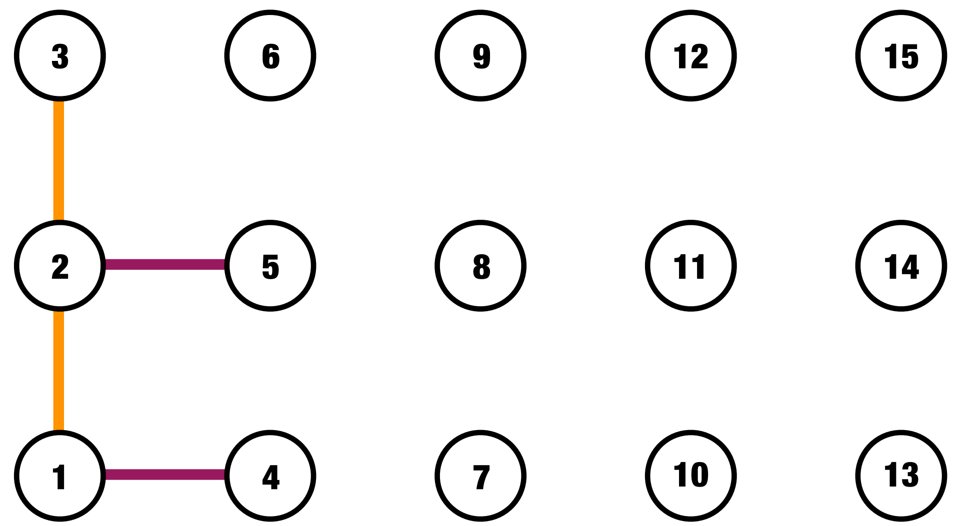


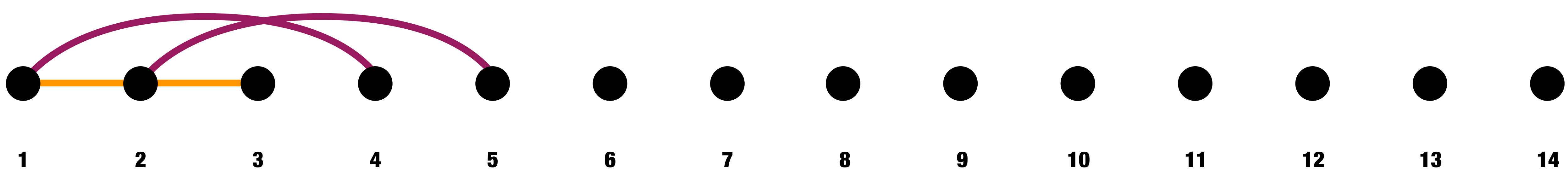


Trick to use a 2D Hamiltonian:

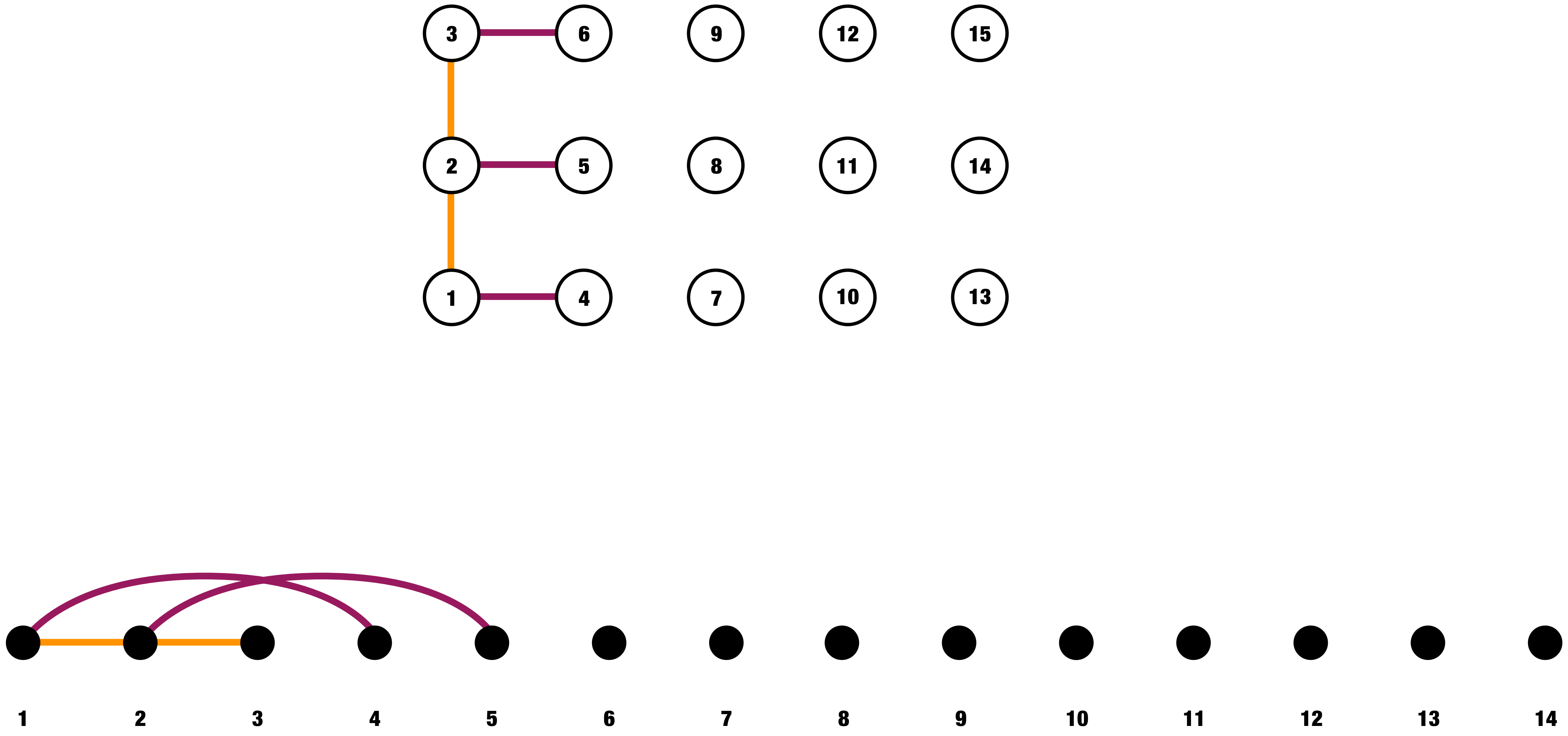


Trick to use a 2D Hamiltonian:

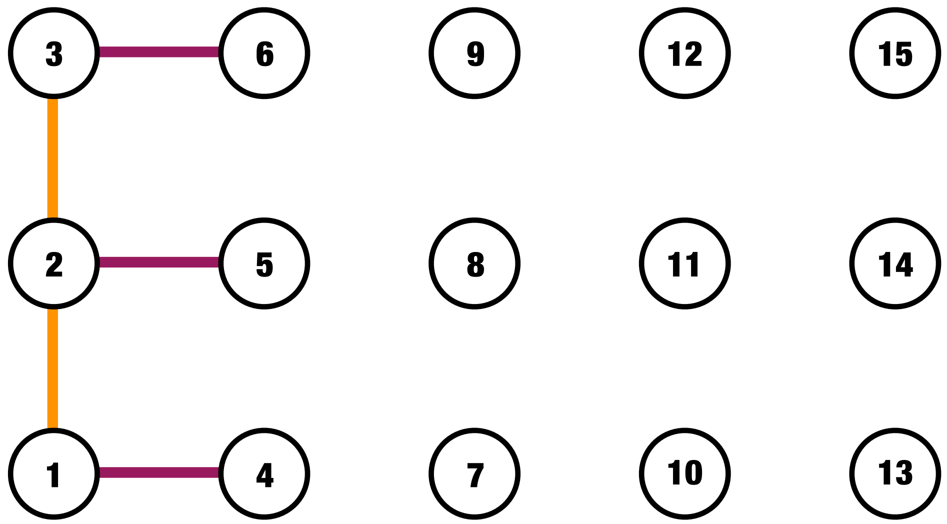


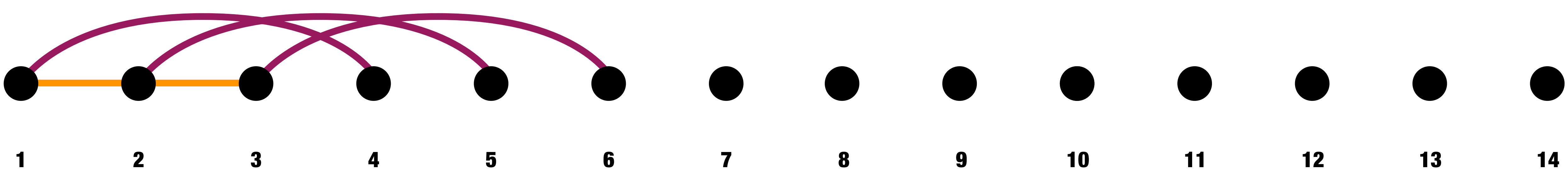


Trick to use a 2D Hamiltonian:

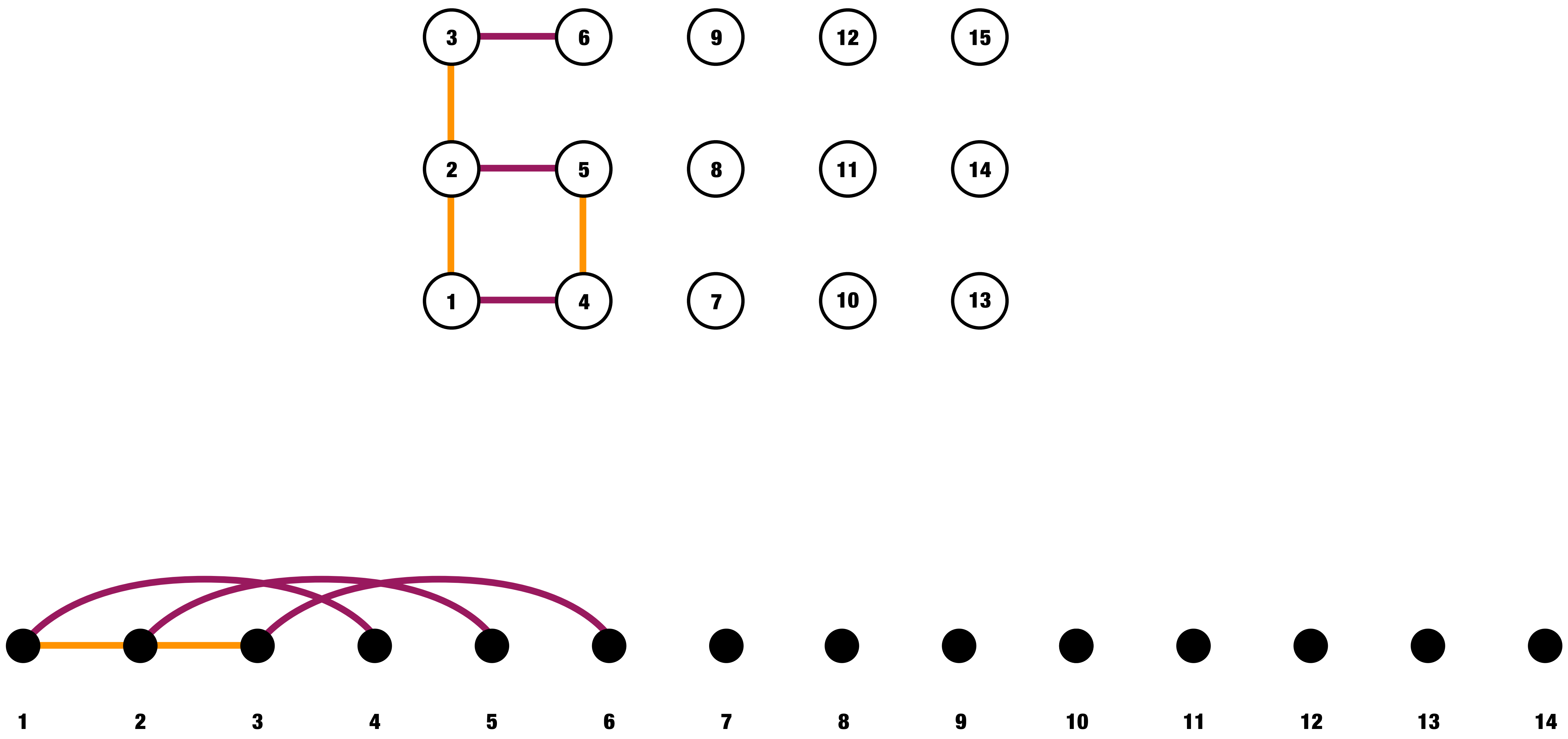


Trick to use a 2D Hamiltonian:

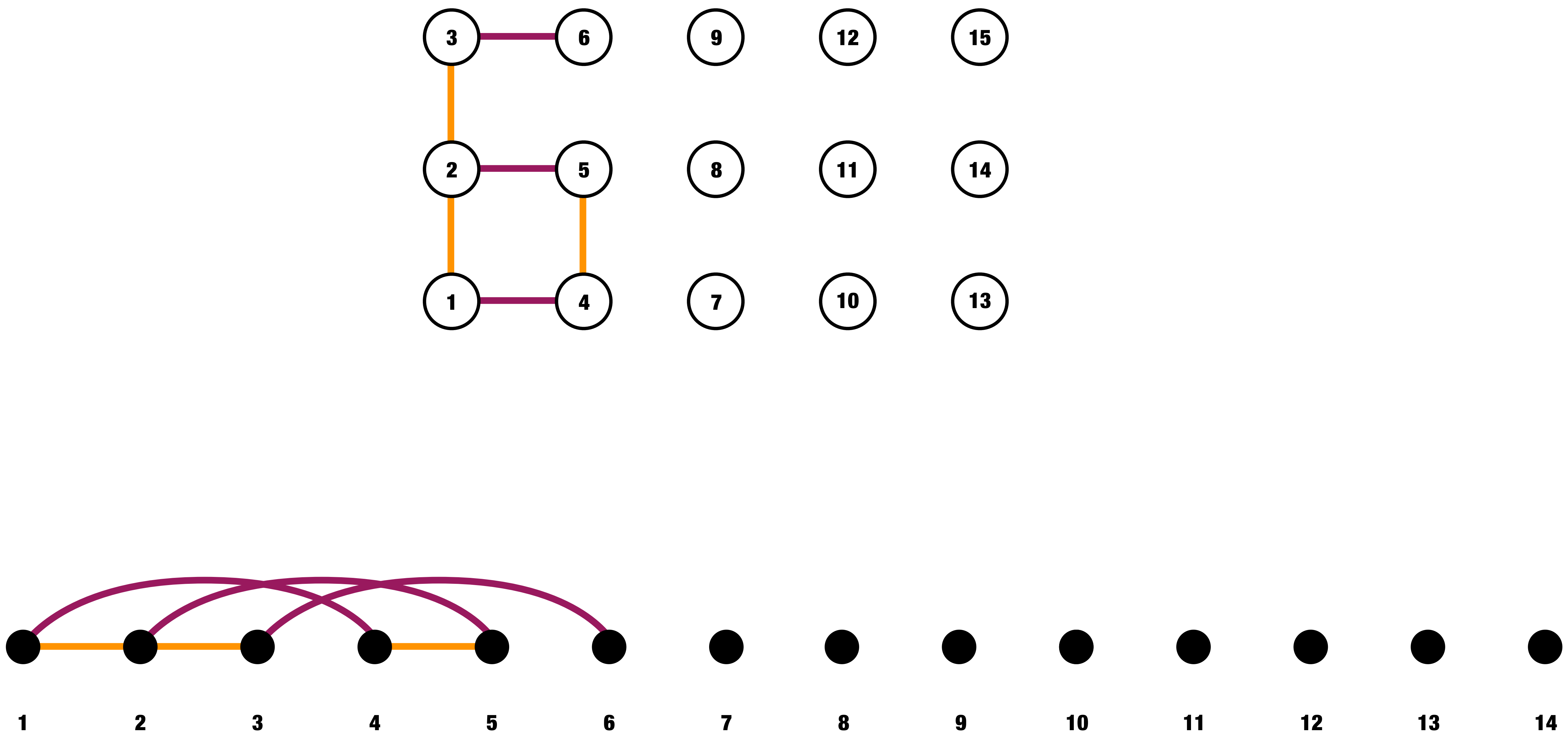




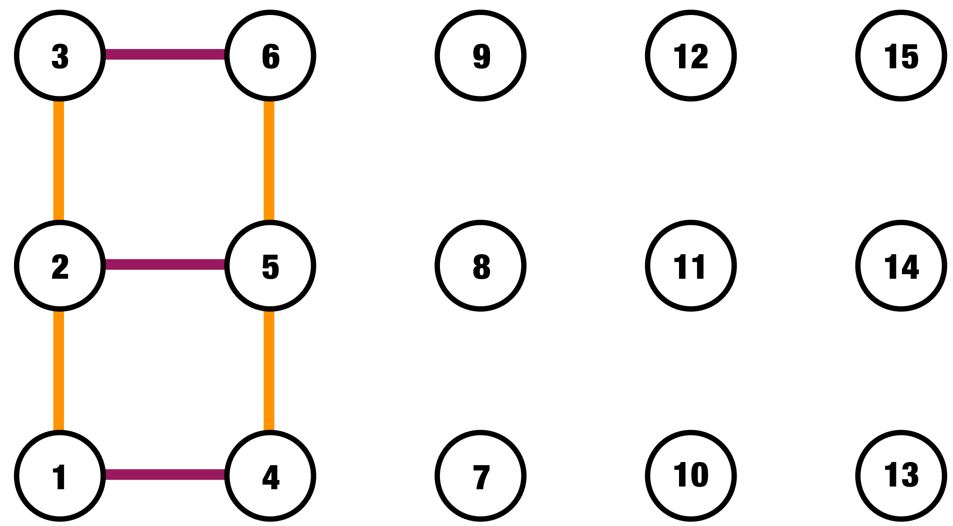
Trick to use a 2D Hamiltonian:

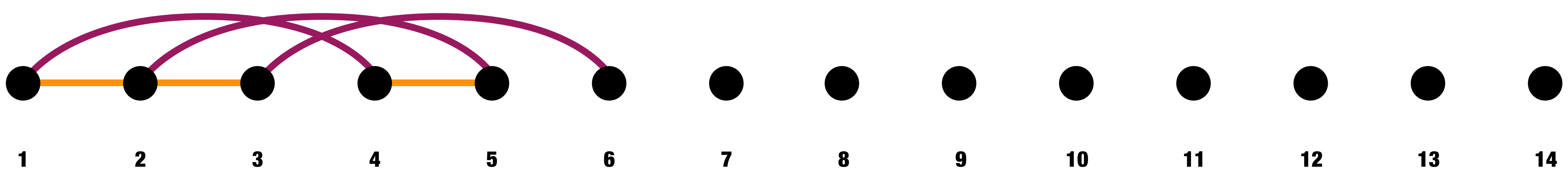


Trick to use a 2D Hamiltonian:

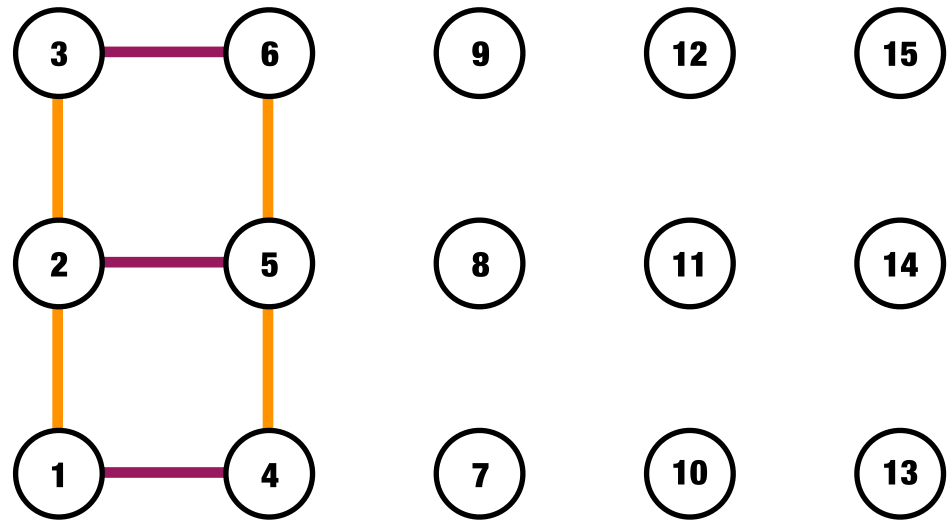


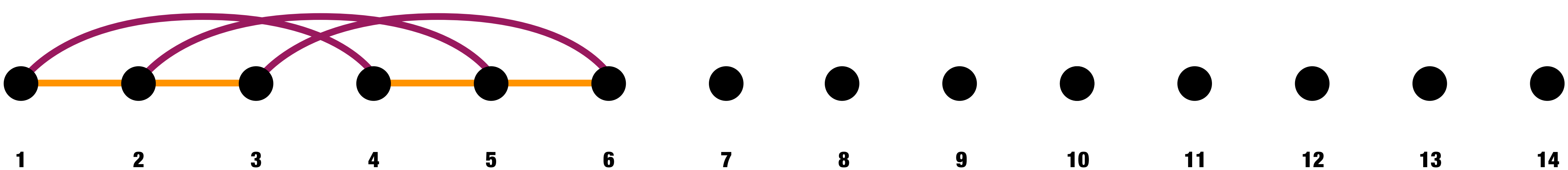
Trick to use a 2D Hamiltonian:



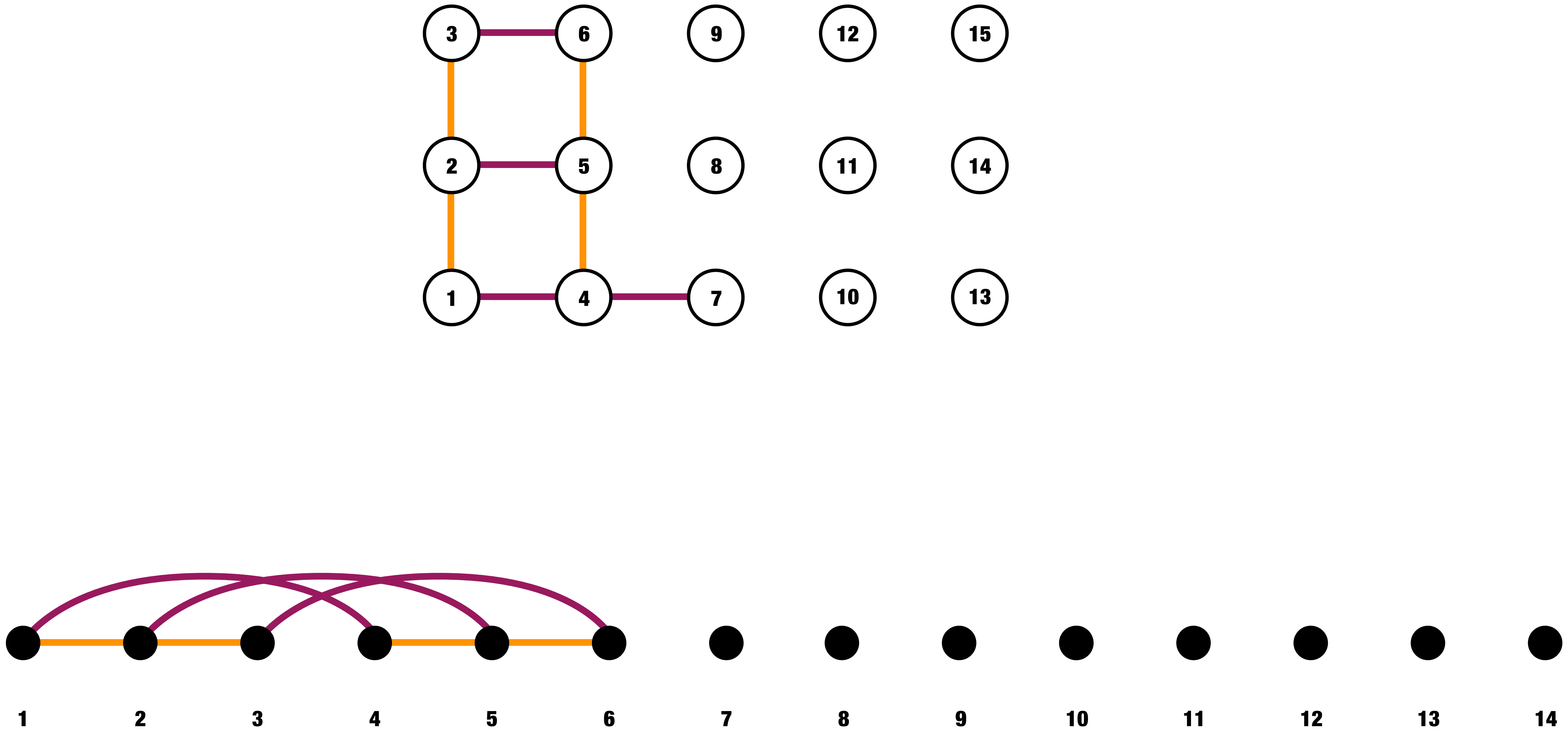


Trick to use a 2D Hamiltonian:

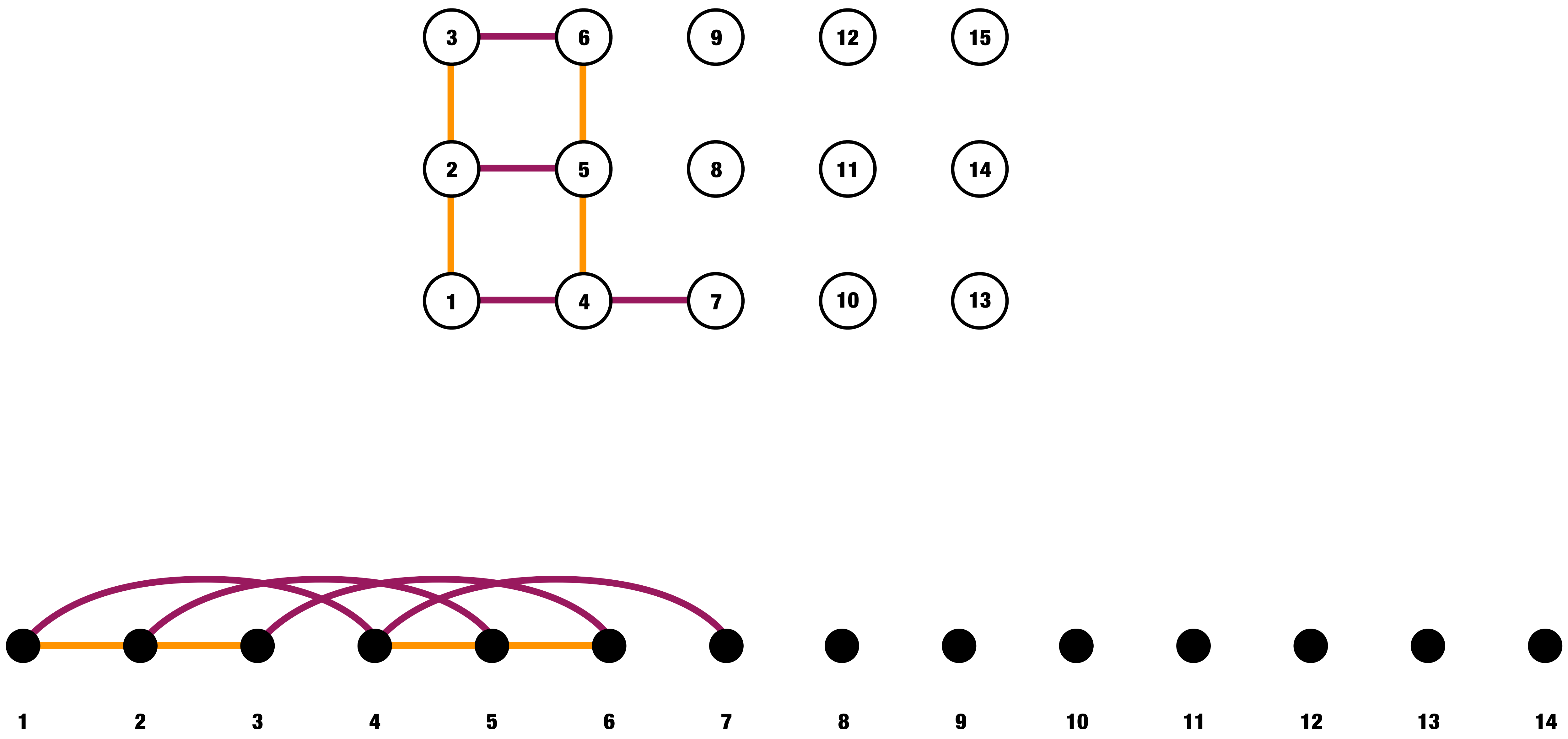




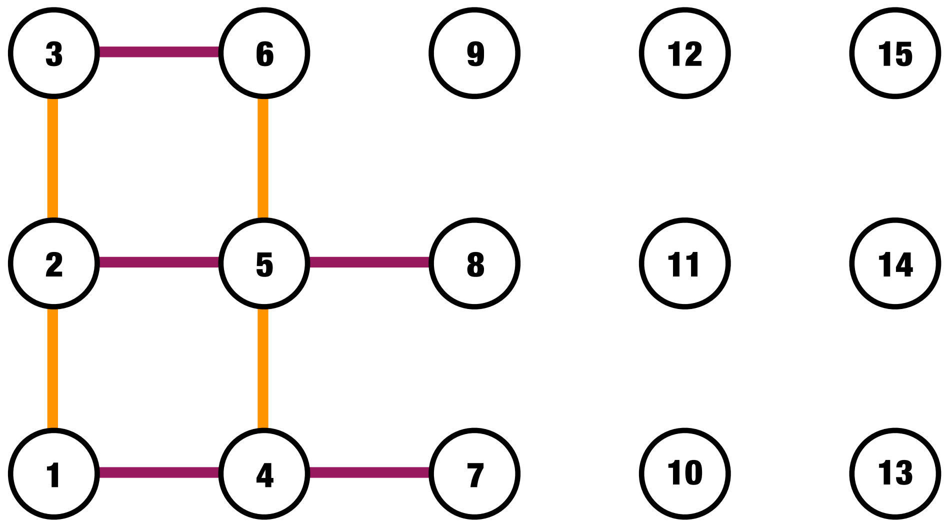
Trick to use a 2D Hamiltonian:

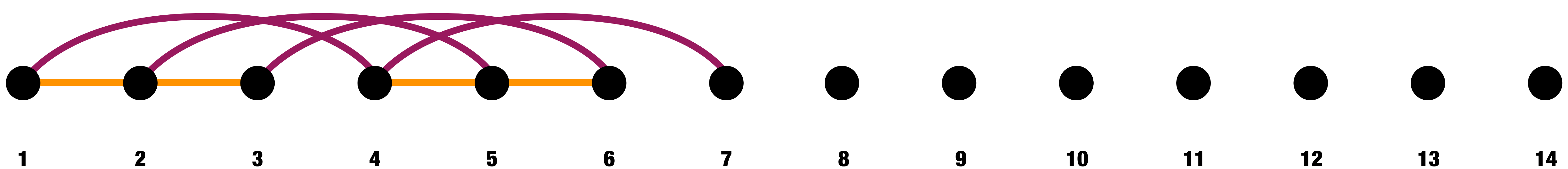


Trick to use a 2D Hamiltonian:

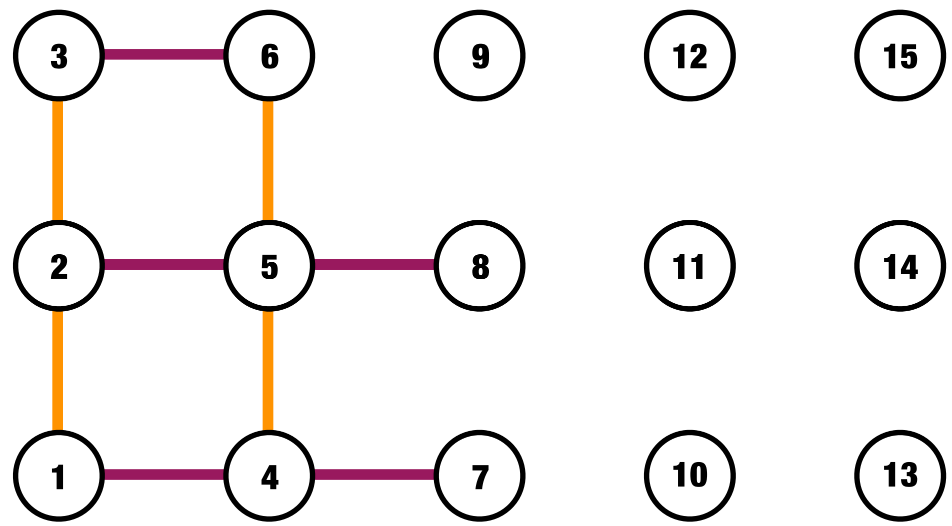


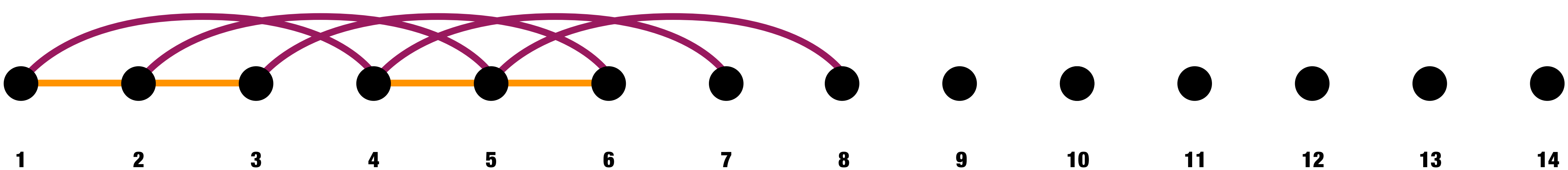
Trick to use a 2D Hamiltonian:



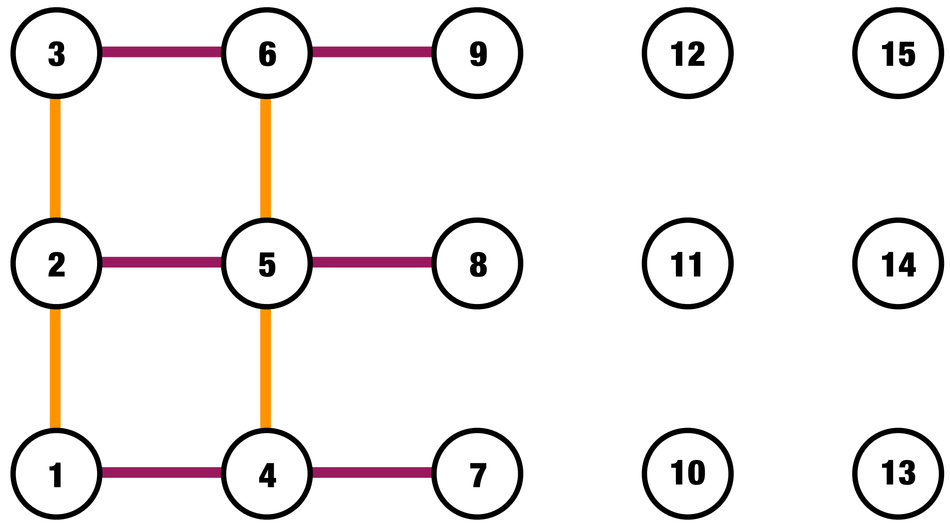


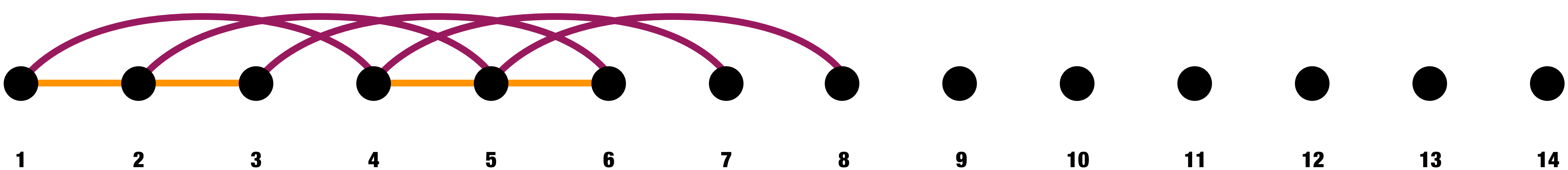
Trick to use a 2D Hamiltonian:



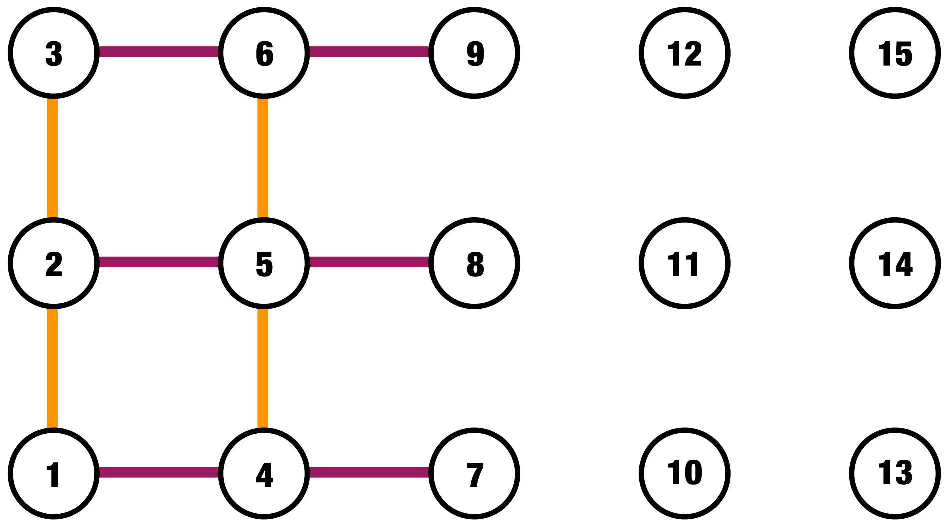


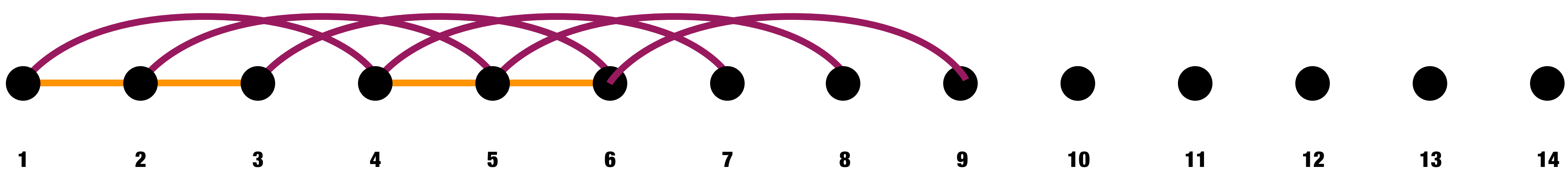
Trick to use a 2D Hamiltonian:



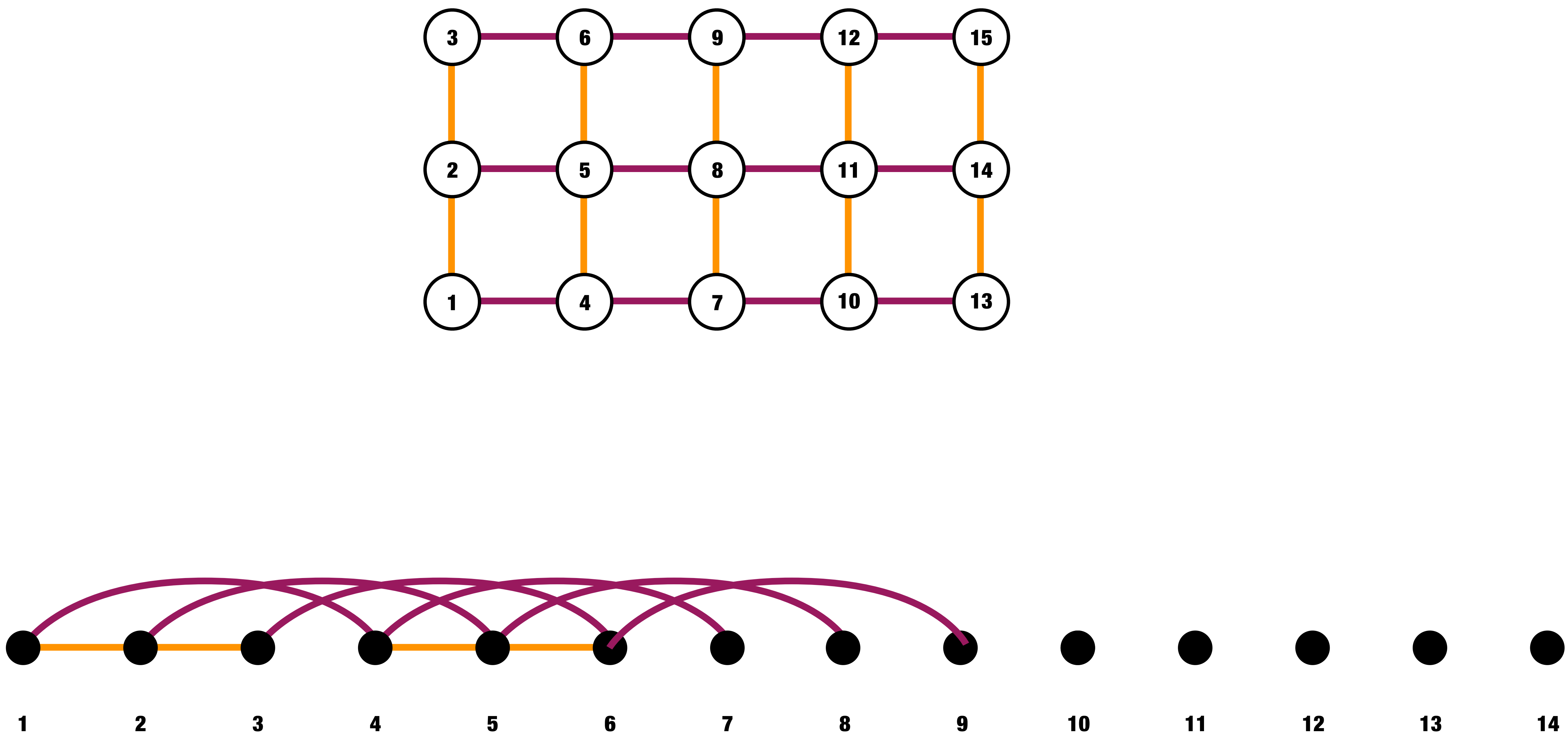


Trick to use a 2D Hamiltonian:

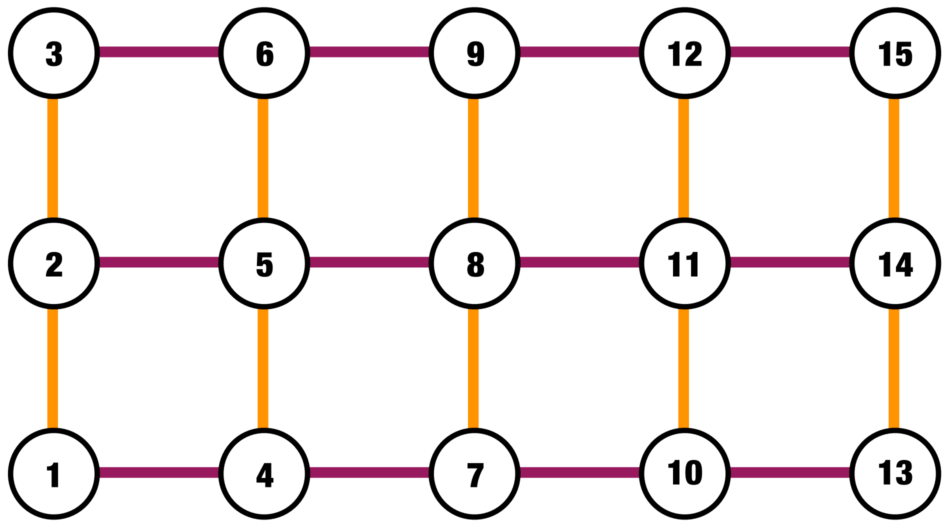


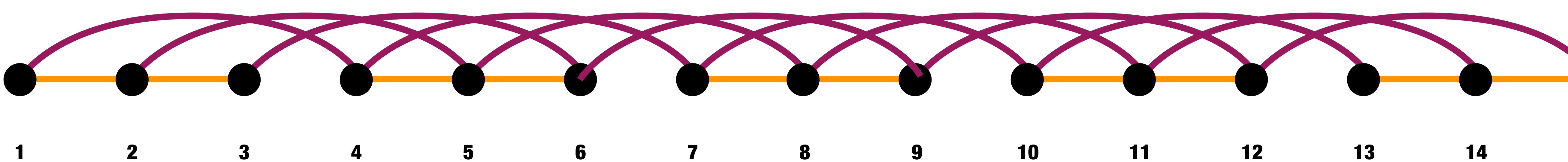


Trick to use a 2D Hamiltonian:

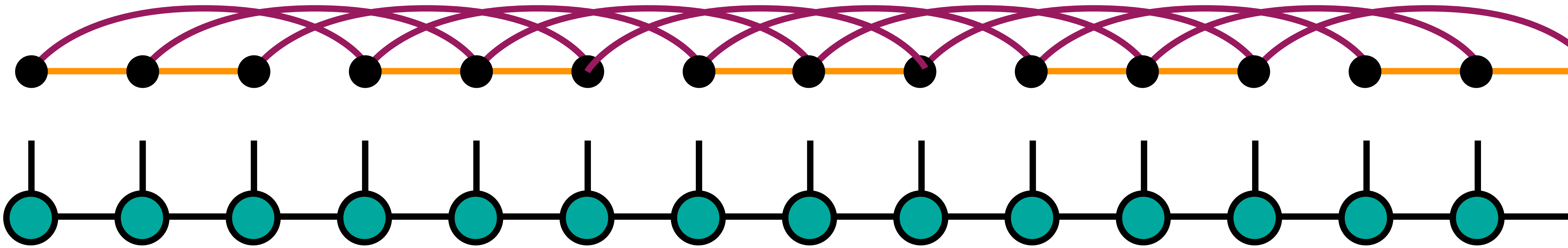


Trick to use a 2D Hamiltonian:

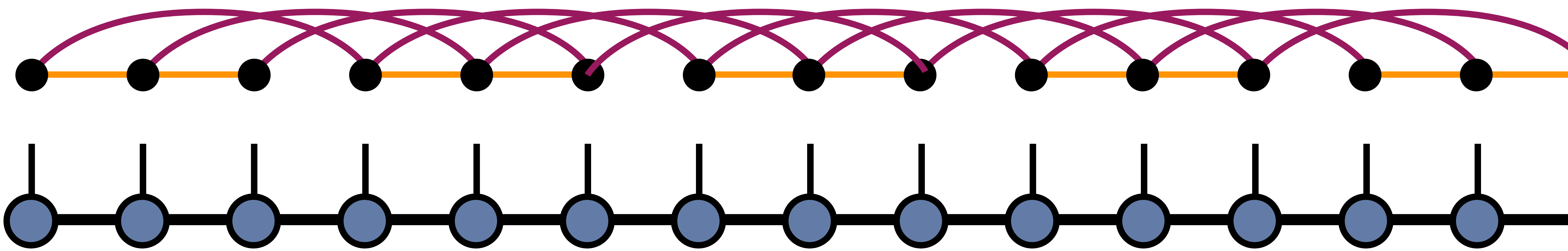




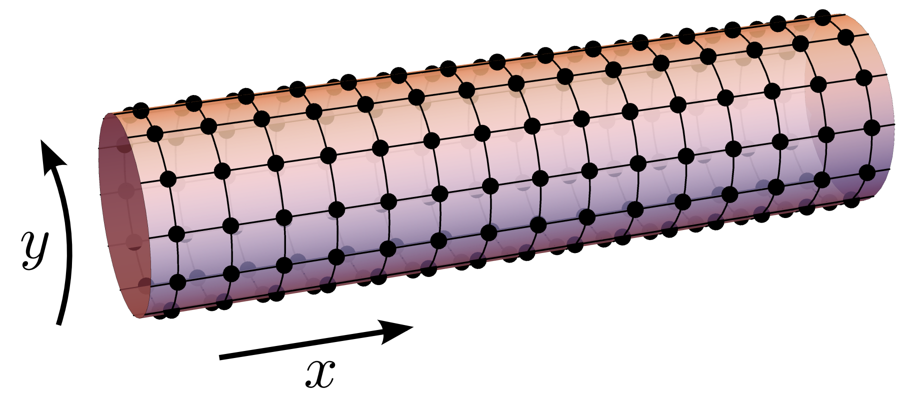
Use this Hamiltonian in DMRG,
get correct results for 2D system, if converged



Use this Hamiltonian in DMRG,
get correct results for 2D system, if converged



General approach called 2D DMRG with snaking path 🐍 :



MPS path:

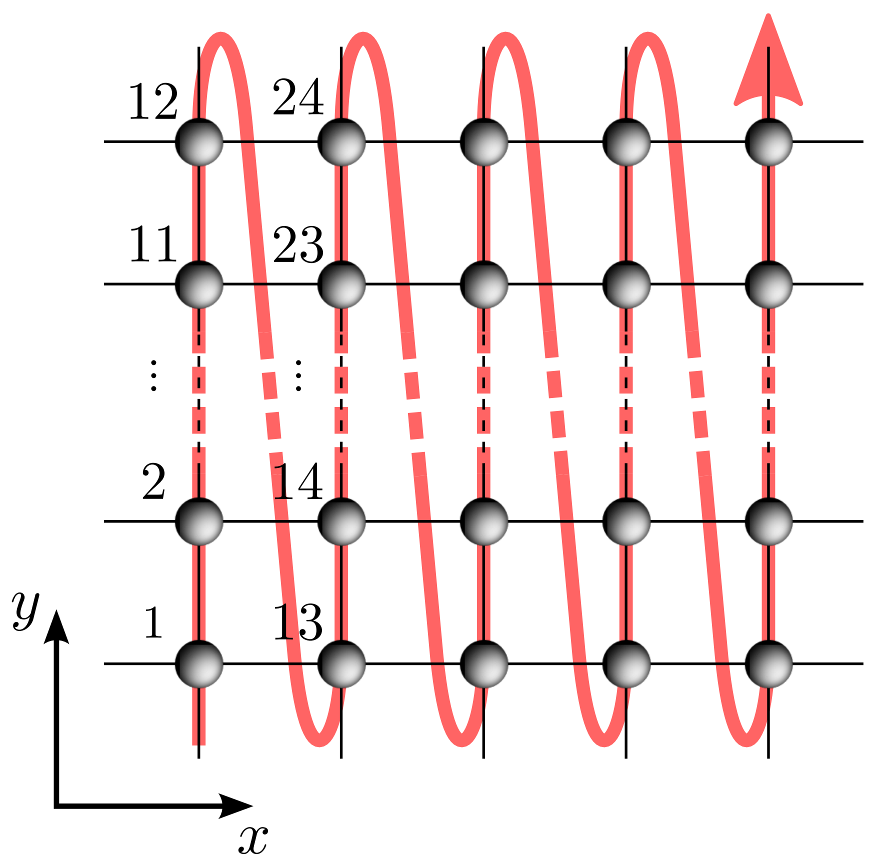
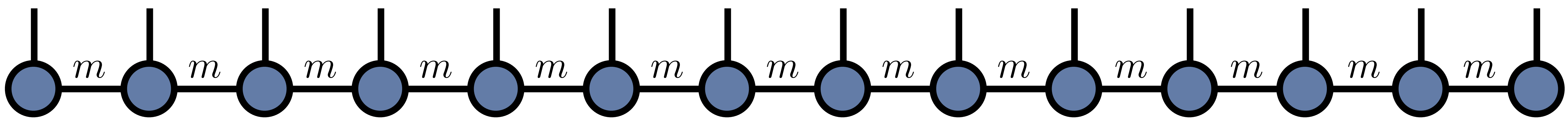


Figure from: Motruk, Zaletel, Mong, Pollmann, arxiv:1512.03318

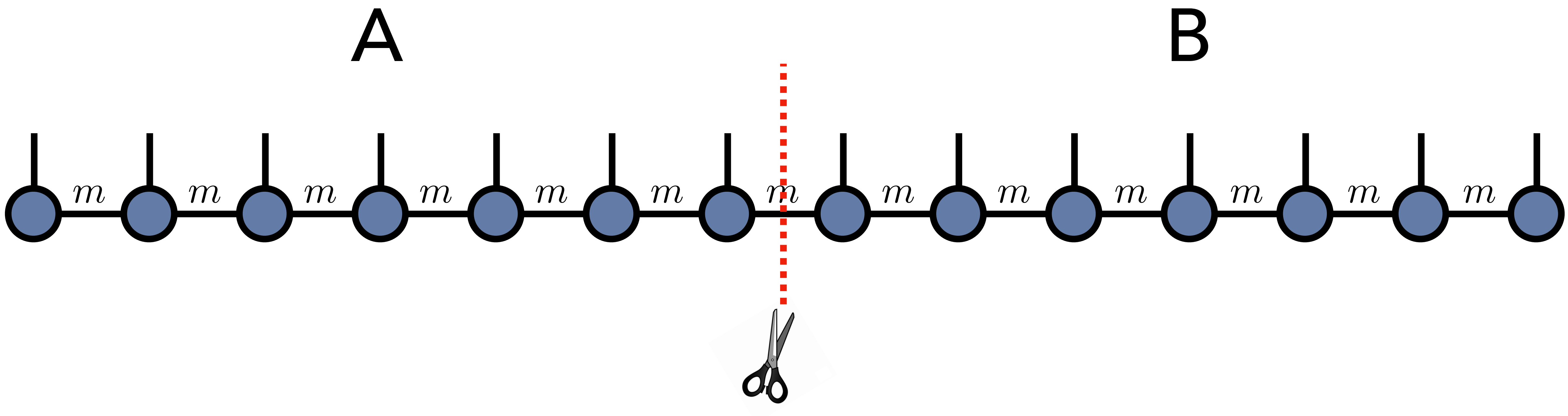
How well does this idea work?

🤔

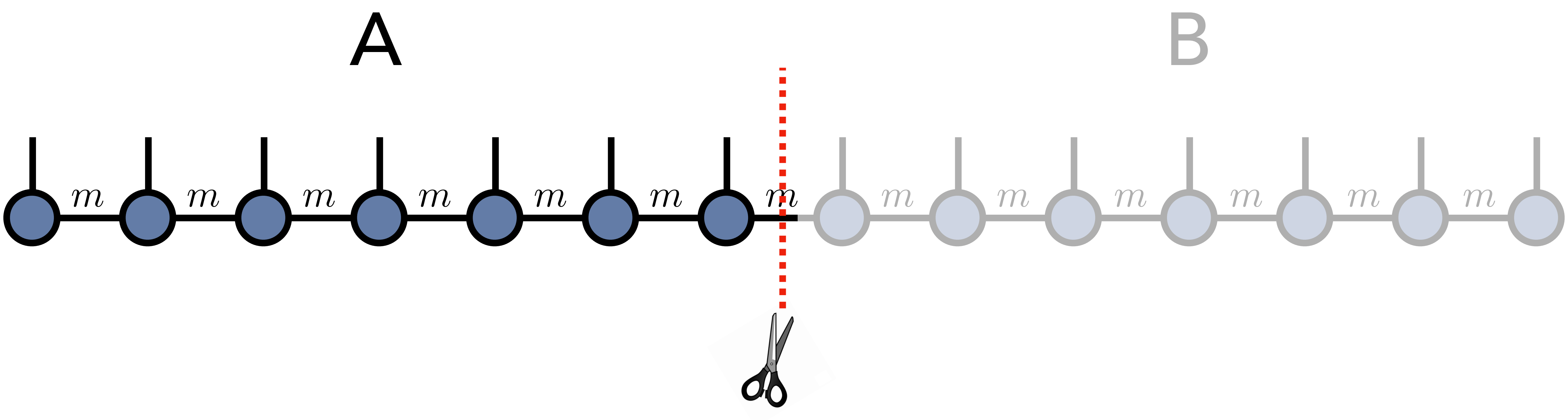
Note that bond dimension ***m*** of MPS related to ***entanglement entropy*** of wavefunction



Note that bond dimension ***m*** of MPS related to ***entanglement entropy*** of wavefunction



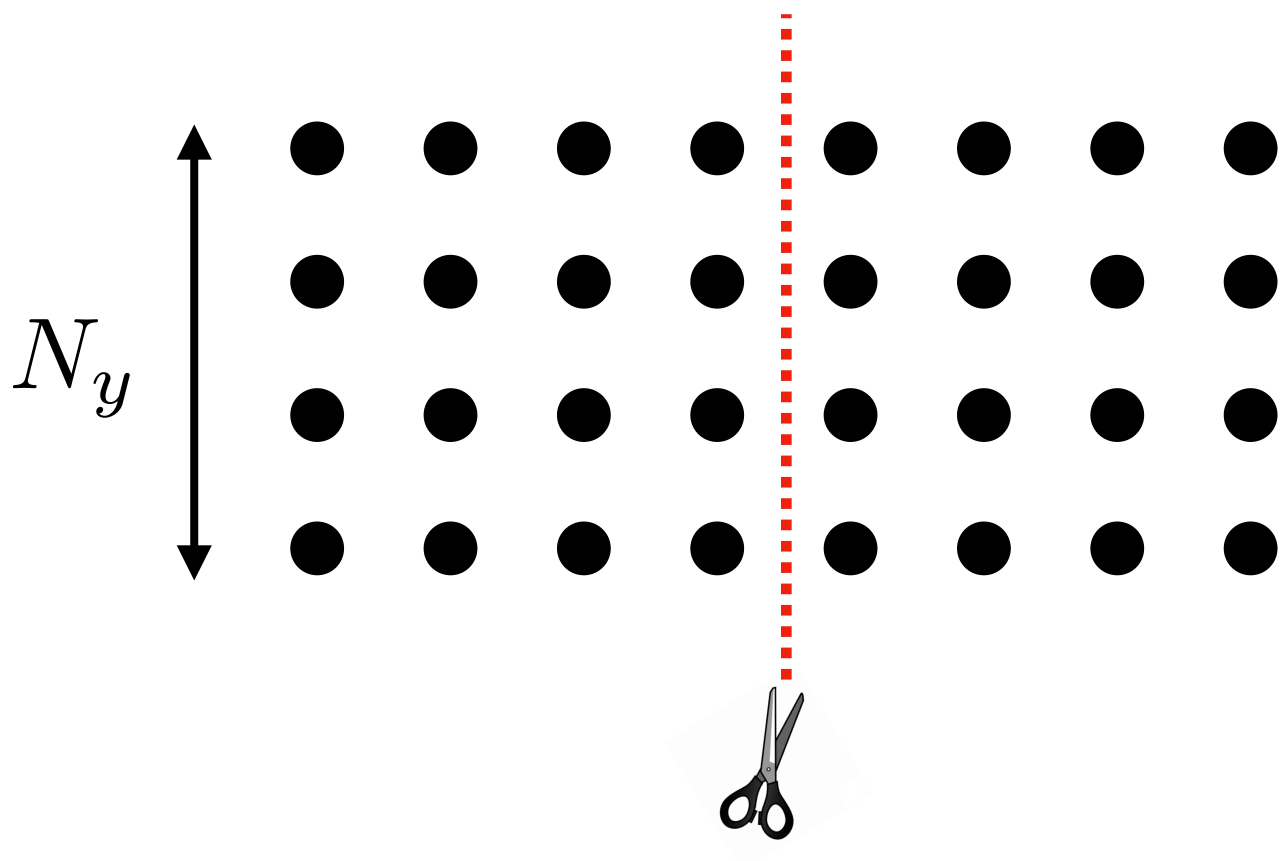
Note that bond dimension ***m*** of MPS related to ***entanglement entropy*** of wavefunction



$$S_A \sim \log m$$

$$m \sim e^{S_A}$$

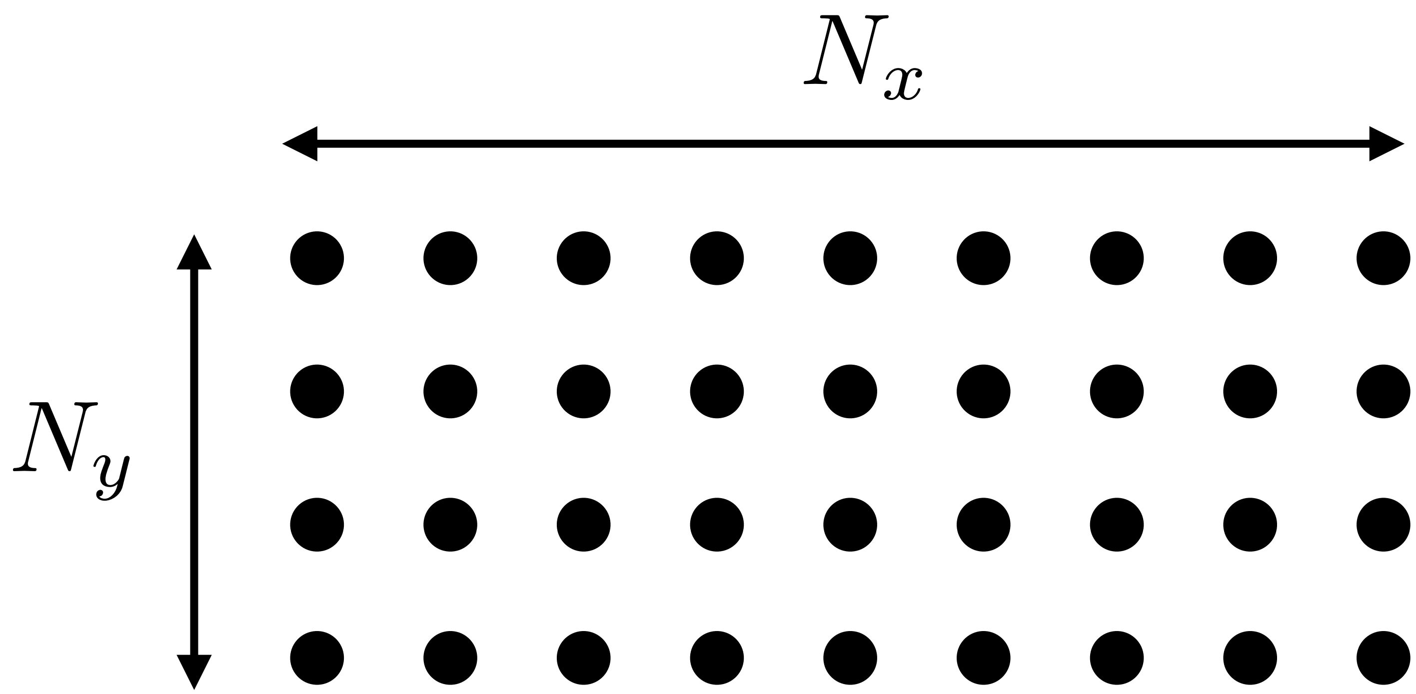
If 2D ground state obeys boundary law (area law),
means $S \sim N_y$



Entanglement of MPS is bounded by log(m)

$$\implies m \sim e^{S_A} \sim e^{N_y} \quad !!$$

DMRG for two-dimensional systems (cylinders) requires extreme care



Scaling is: $N_x \, e^{a N_y}$

Like exact diagonalization, but only exponential in one direction (N_y), linear in other direction

Only N_y ~ 8–12 usually reachable

However, 2D DMRG can be very effective

Advantages over other 2D methods:

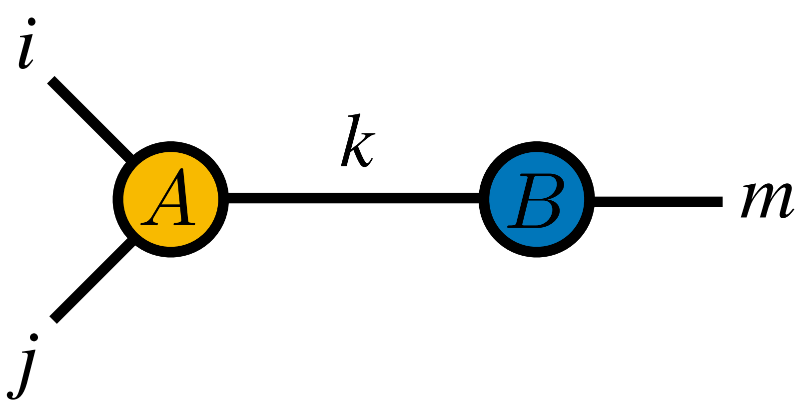
- robust convergence for fixed bond-dimension
- no statistical error (as in QMC)
- treat any Hamiltonian (no sign problem)
- can measure most any observable, including correlation functions and entanglement

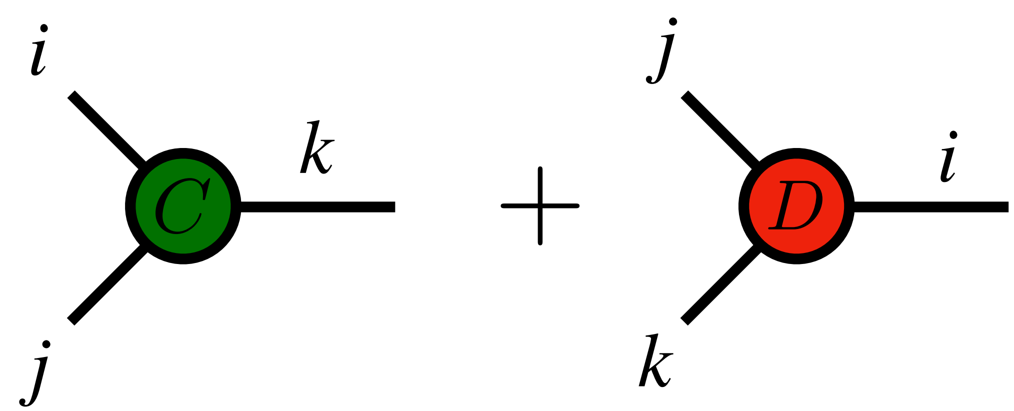
"Only" limitation is **poor scaling with N_y**

2D DMRG in ITensor

ITensor

itensor.org






```
A * B
```

```
C + D
```

Automatic tensor contractions

"Magnetic" indices "snap" together

ITensor library DMRG interface:

```
int N = 100;
auto sites = SpinHalf(N);


auto ampo = AutoMPO(sites);
for(auto j : range1(N-1))
  {
  ampo += 0.5,"S+",j,"S-",j+1;
  ampo += 0.5,"S-",j,"S+",j+1;
  ampo +=     "Sz",j,"Sz",j+1;
 }


auto H = toMPO(ampo);


auto state = InitState(sites);
for(auto j : range1(N))
  {
  state.set(j,(j%2==1 ? "Up" : "Dn"));
  }
auto psi0 = MPS(state);


auto sweeps = Sweeps(5);
sweeps.maxdim() = 10,20,100;
sweeps.cutoff() = 1E-6;
auto [energy,psi] = dmrg(H,psi0,sweeps);
```

ITensor library DMRG interface:

ITensor library DMRG interface:

```
int N = 100;
auto sites = SpinHalf(N);
```

ITensor library DMRG interface:

```
int N = 100;
auto sites = SpinHalf(N);
```

```
auto ampo = AutoMPO(sites);
for(auto j : range1(N-1))
  {
  ampo += 0.5,”S+",j,"S-",j+1;
  ampo += 0.5,"S-",j,"S+",j+1;
  ampo +=     ”Sz”,j,"Sz",j+1;
 }
```

$$\hat{H} = \sum_j \frac{1}{2}(S^+_j S^-_{j+1} + S^-_j S^+_{j+1}) + S^z_j S^z_{j+1}$$

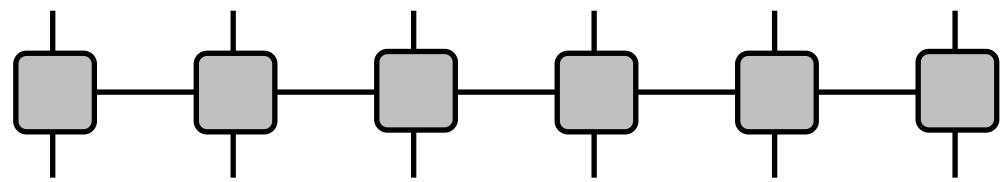
ITensor library DMRG interface:

```
int N = 100;
auto sites = SpinHalf(N);
```

```
auto ampo = AutoMPO(sites);
for(auto j : range1(N-1))
  {
  ampo += 0.5,”S+",j,"S-",j+1;
  ampo += 0.5,"S-",j,"S+",j+1;
  ampo +=     ”Sz”,j,"Sz",j+1;
 }
```

$$\hat{H} = \sum_j \frac{1}{2}(S^+_j S^-_{j+1} + S^-_j S^+_{j+1}) + S^z_j S^z_{j+1}$$

```
auto H = toMPO(ampo);
```



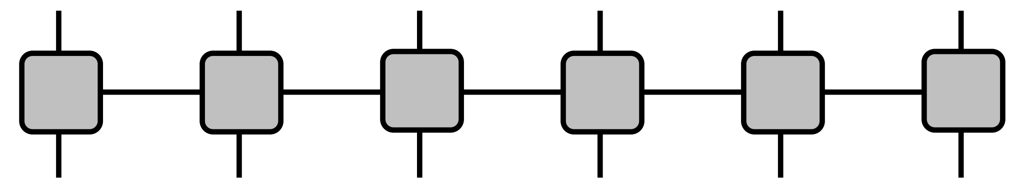
ITensor library DMRG interface:

```
int N = 100;
auto sites = SpinHalf(N);
```

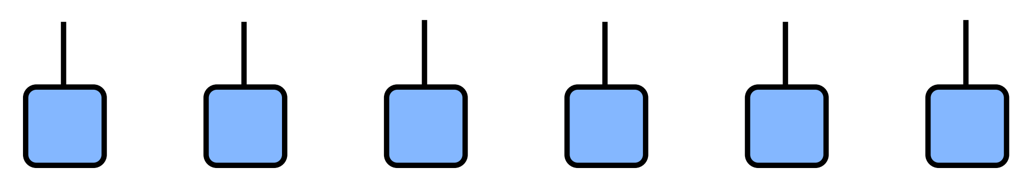
```
auto ampo = AutoMPO(sites);
for(auto j : range1(N-1))
  {
  ampo += 0.5,"S+",j,"S-",j+1;
  ampo += 0.5,"S-",j,"S+",j+1;
  ampo +=     "Sz",j,"Sz",j+1;
 }
```

$$\hat{H} = \sum_j \frac{1}{2}(S^+_j S^-_{j+1} + S^-_j S^+_{j+1}) + S^z_j S^z_{j+1}$$

```
auto H = toMPO(ampo);
```



```
auto state = InitState(sites);
for(auto j : range1(N))
  {
  state.set(j,(j%2==1 ? "Up" : "Dn"));
  }
auto psi0 = MPS(state);
```



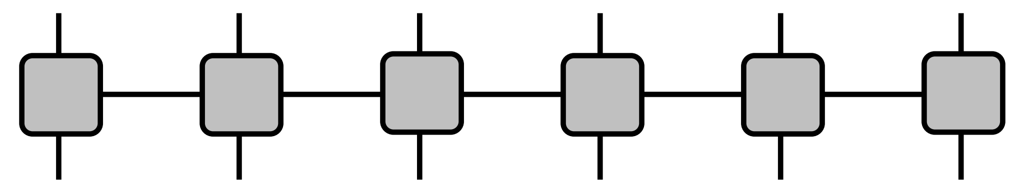
ITensor library DMRG interface:

```
int N = 100;
auto sites = SpinHalf(N);
```

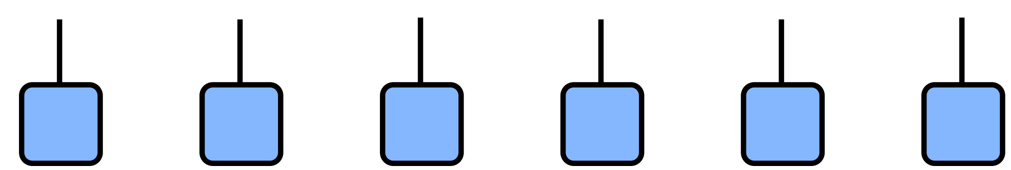
```
auto ampo = AutoMPO(sites);
for(auto j : range1(N-1))
  {
  ampo += 0.5,”S+",j,"S-",j+1;
  ampo += 0.5,"S-",j,"S+",j+1;
  ampo +=     ”Sz”,j,"Sz",j+1;
 }
```

$$\hat{H} = \sum_j \frac{1}{2}(S^+_j S^-_{j+1} + S^-_j S^+_{j+1}) + S^z_j S^z_{j+1}$$

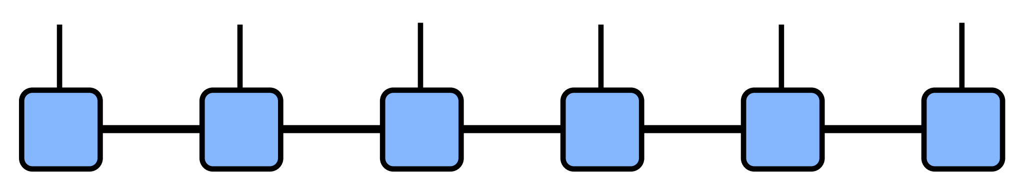
```
auto H = toMPO(ampo);
```



```
auto state = InitState(sites);
for(auto j : range1(N))
  {
  state.set(j,(j%2==1 ? "Up" : "Dn"));
  }
auto psi0 = MPS(state);
```



```
auto sweeps = Sweeps(5);
sweeps.maxdim() = 10,20,100;
sweeps.cutoff() = 1E-6;
auto [energy,psi] = dmrg(H,psi0,sweeps);
```



Have entire wavefunction afterward,
can do any measurements:

```
for(int j = 1; j <= N; ++j)
    {
    psi.position(j);

    Real Szj = elt(psi(j)* op(sites,"Sz",j) * dag(prime(psi(j),"Site")));

    println("Sz_",j," = ",Szj);
    }
```

```
Sz_1 = 0.405242
Sz_2 = -0.202632
Sz_3 = 0.119827
...
```

2D DMRG with ITensor:

```
auto lattice = squareLattice(Nx,Ny,{"YPeriodic",true});

auto ampo = AutoMPO(sites);
for(auto b : lattice)
    {
    ampo += 0.5,"S+",b.s1,"S-",b.s2;
    ampo += 0.5,"S-",b.s1,"S+",b.s2;
    ampo +=     "Sz",b.s1,"Sz",b.s2;
    }
auto H = MPO(ampo);
```

squareLattice function returns array (vector) of structs labeling site pairs defining square lattice

ITensor coming to Julia Language

```
#include "itensor/all.h"
using namespace itensor;

int
main()
    {
    int N = 100;
    auto sites = SpinOne(N);

    auto ampo = AutoMPO(sites);
    for(auto j : range1(N-1))
        {
        ampo += 0.5,"S+",j,"S-",j+1;
        ampo += 0.5,"S-",j,"S+",j+1;
        ampo +=     "Sz",j,"Sz",j+1;
        }
    auto H = toMPO(ampo);

    auto psi0 = randomMPS(sites);

    auto sweeps = Sweeps(5);
    sweeps.maxdim() = 10,20,100,100,200;
    sweeps.cutoff() = 1E-10;
    println(sweeps);

    auto [energy,psi] = dmrg(H,psi0,sweeps,"Quiet");

    return 0;
    }
```

Matt Fishman

Katie Hyatt

ITensor coming to Julia Language

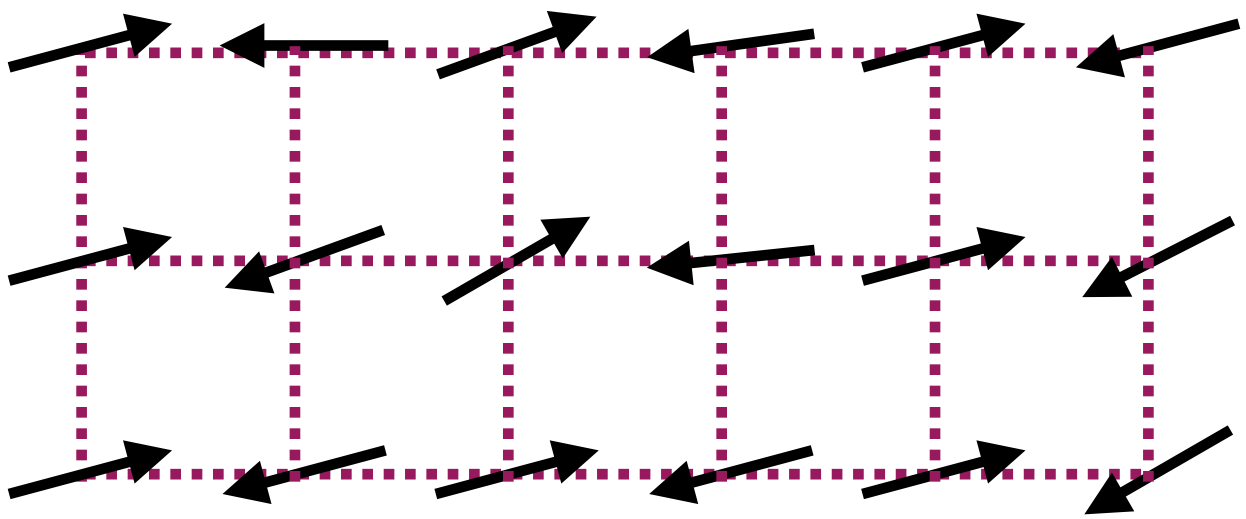
```
using ITensors, Printf
let
  N = 100
  sites = spinOneSites(N)

  ampo = AutoMPO()
  for j=1:N-1
      add!(ampo,"Sz",j,"Sz",j+1)
      add!(ampo,0.5,"S+",j,"S-",j+1)
      add!(ampo,0.5,"S-",j,"S+",j+1)
  end
  H = toMPO(ampo,sites)

  psi0 = randomMPS(sites)

  sweeps = Sweeps(5)
  maxdim!(sweeps, 10,20,100,100,200)
  cutoff!(sweeps, 1E-10)
  @show sweeps

  energy, psi = dmrg(H,psi0, sweeps)
  @printf("Final energy = %.12f\n",energy)
end
```

Matt Fishman

Katie Hyatt

Applications of 2D DMRG

2D quantum magnets



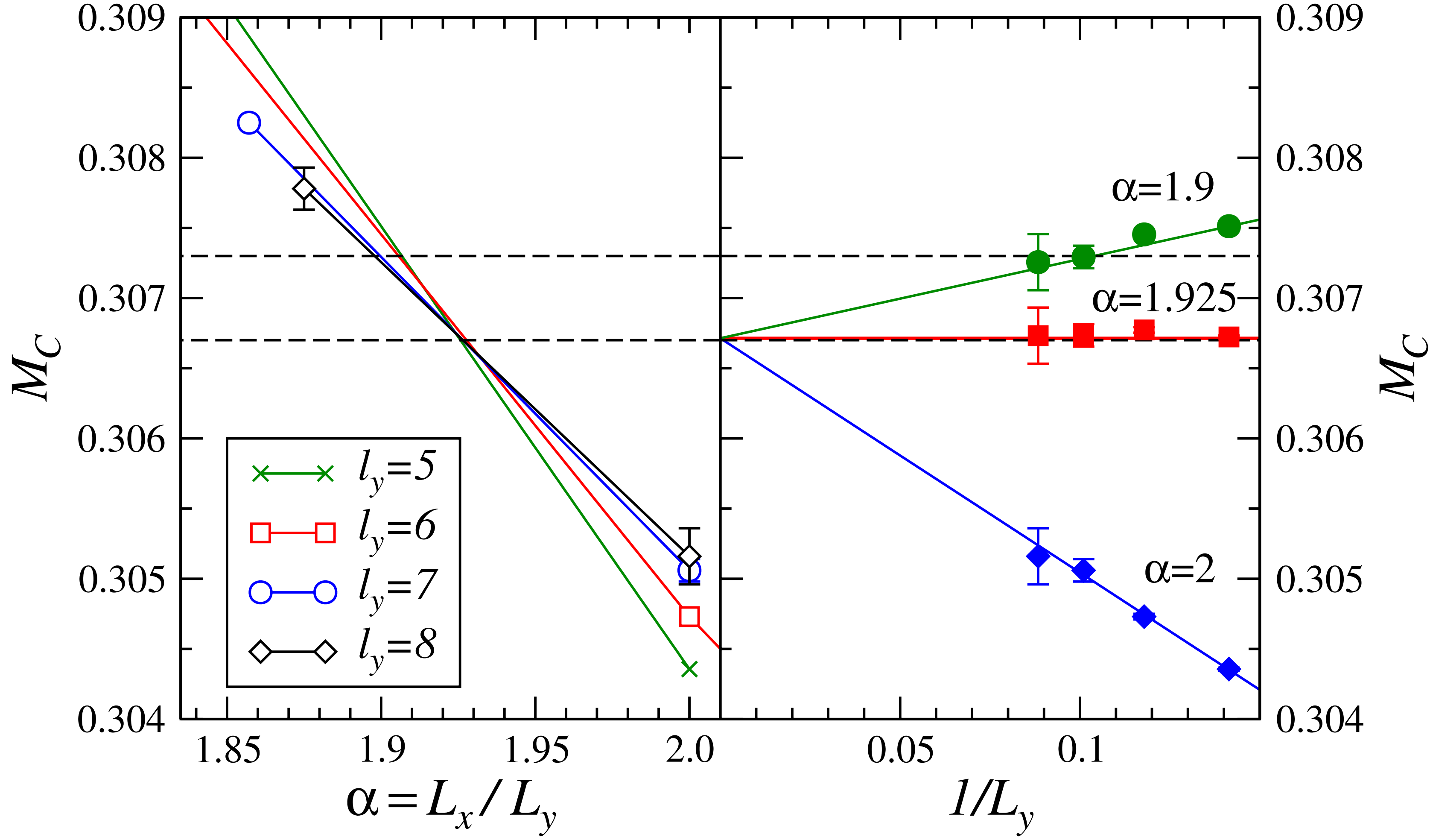
Prototypical model is the *Heisenberg model*

$$\hat{H} = \sum_{\langle ij \rangle} \mathbf{S}_i \cdot \mathbf{S}_j$$

$$= \sum_{\langle ij \rangle} (S_i^+ S_j^- + S_i^- S_j^+) + S_i^z S_j^z$$

Magnetization of square-lattice Heisenberg model (using DMRG):

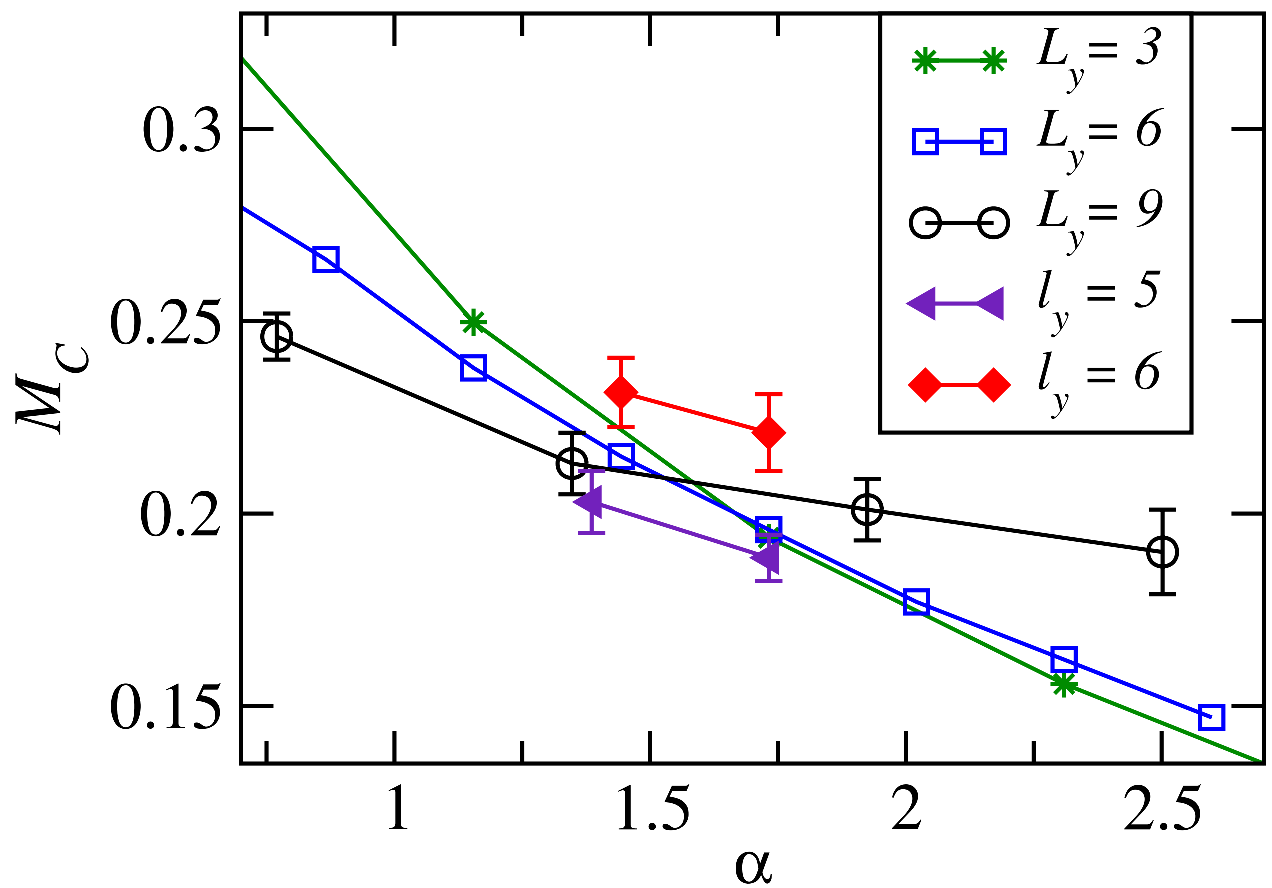
QMC bounds: {



With careful finite-size scaling,
2D DMRG competitive with quantum Monte Carlo

White, Chernyshev, PRL **99**, 127004 (2007)

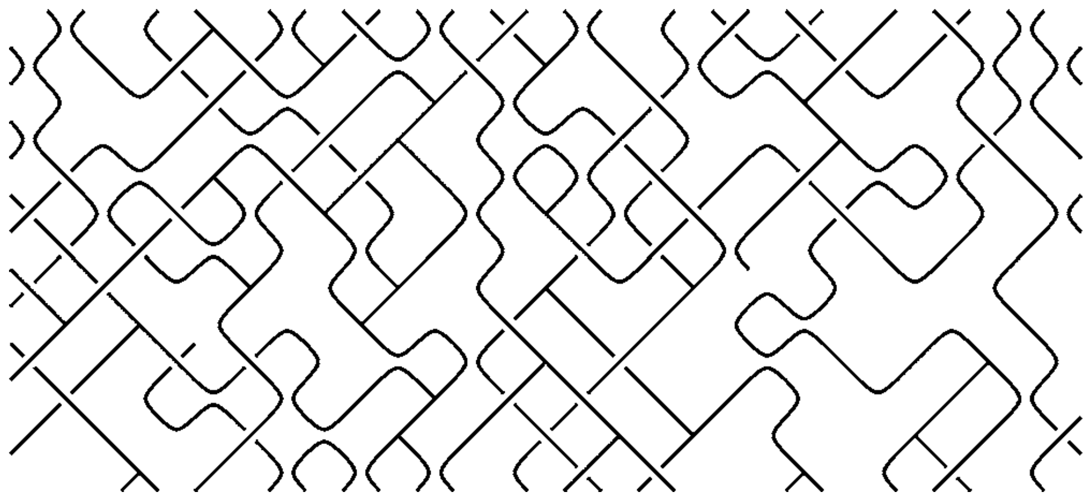
Magnetization of **triangular-lattice** Heisenberg model (using DMRG):



Beyond ability of most other methods to treat

White, Chernyshev, PRL **99**, 127004 (2007)

2D topological phases ("topological order")

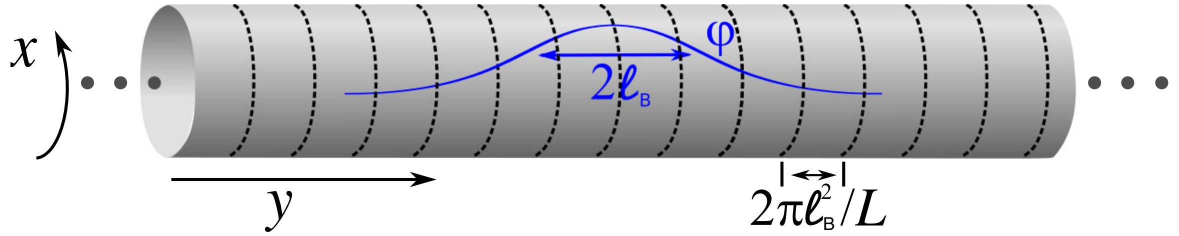


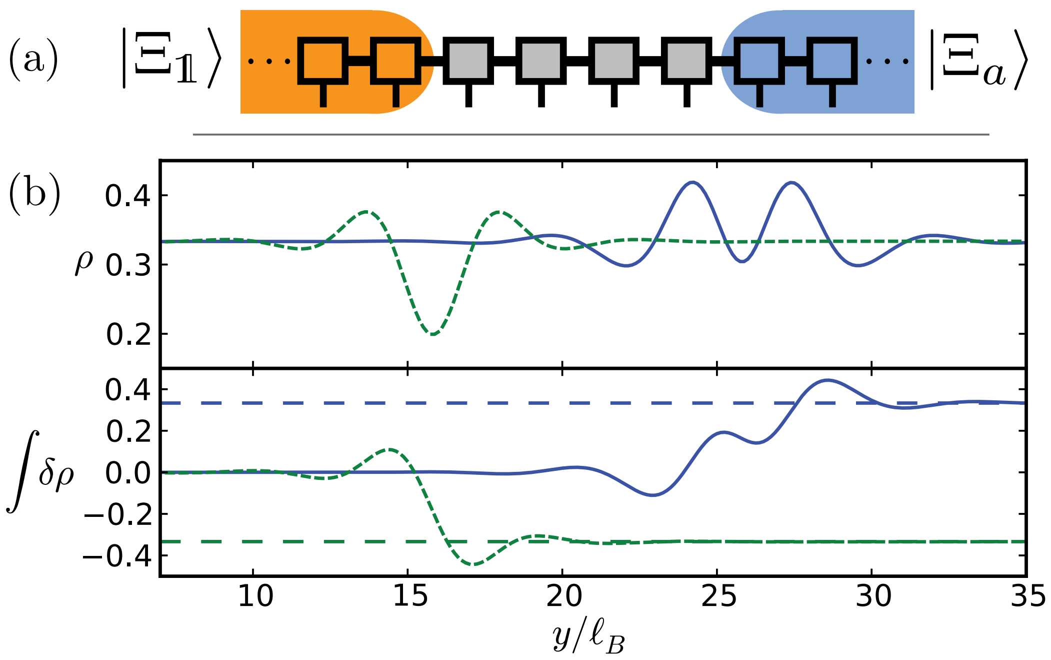
- phase transitions not due to conventional order
- intrinsically robust to perturbations
- 'anyon' quasiparticle excitations
- physical edge can have special properties

Quantum Hall systems: prototypical topological phase

Hamiltonian defined in ***the continuum***

Approximate continuum by set of orbital functions wrapping around a cylinder



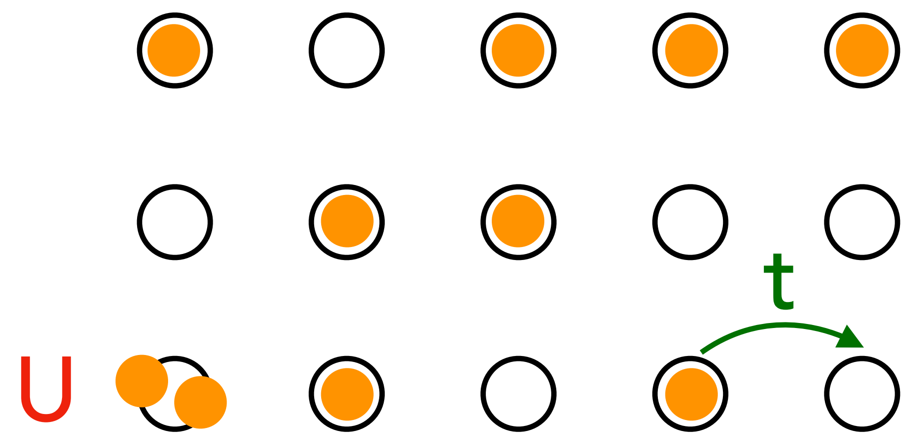


Directly observe fractional-charge quasiparticles

Zaletel, Mong, Pollmann, PRL **110**, 236801 (2013)

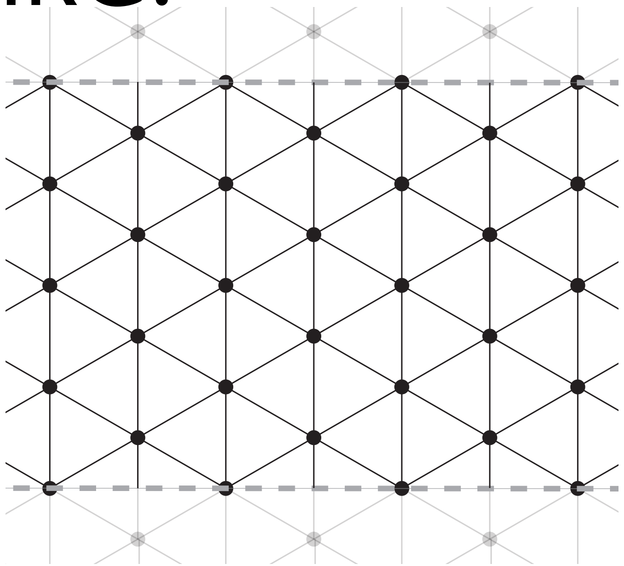
See also: Zaletel, Mong, PRB **86**, 245305 (2012)

2D strongly correlated electrons

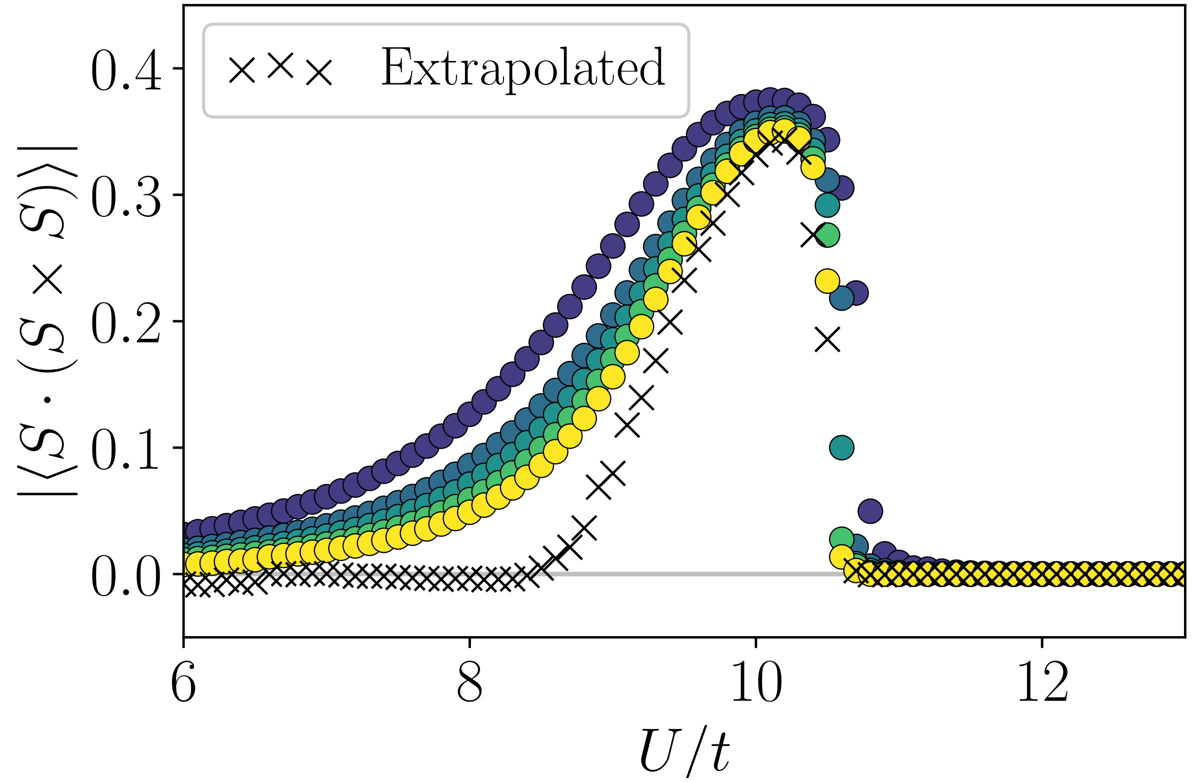


- model systems (Hubbard, tJ) often studied
- qualitative understanding of high-Tc superconductivity?
- possibility of seeing exotic Mott insulator physics (spin liquids & topological order)

Spin liquid observed in triangular lattice Hubbard model with DMRG:



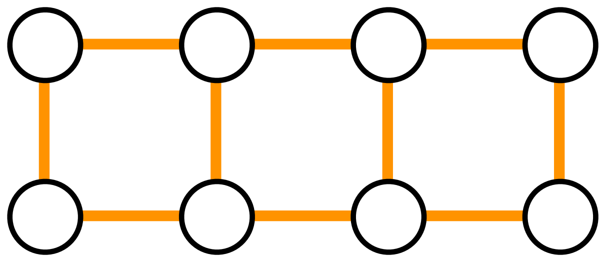
$$H = -t \sum_{\langle ij \rangle \sigma} c^{\dagger}_{i\sigma} c_{j\sigma} + \text{H.c.} + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$



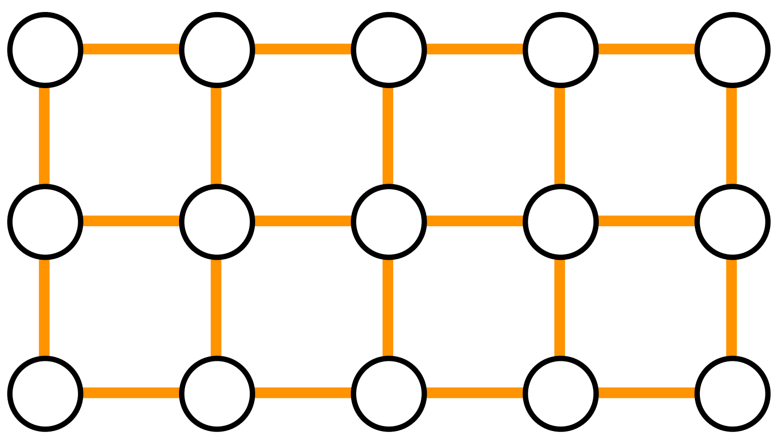
Szasz, Motruk, Zaletel, Moore, arxiv:1808.00463 (2018)

Best Practices for 2D DMRG

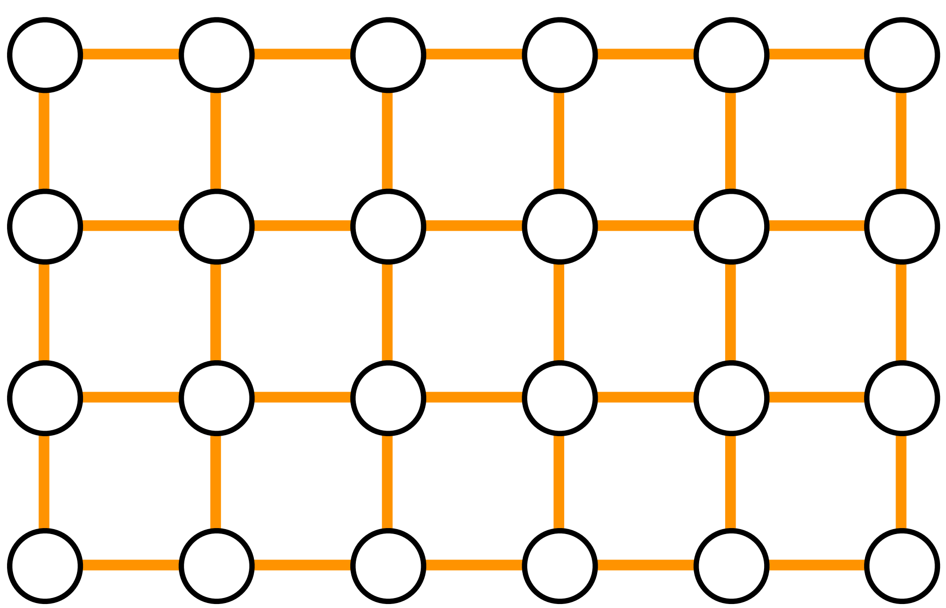
Treat each transverse size N_y as its own system:



$N_y = 2$

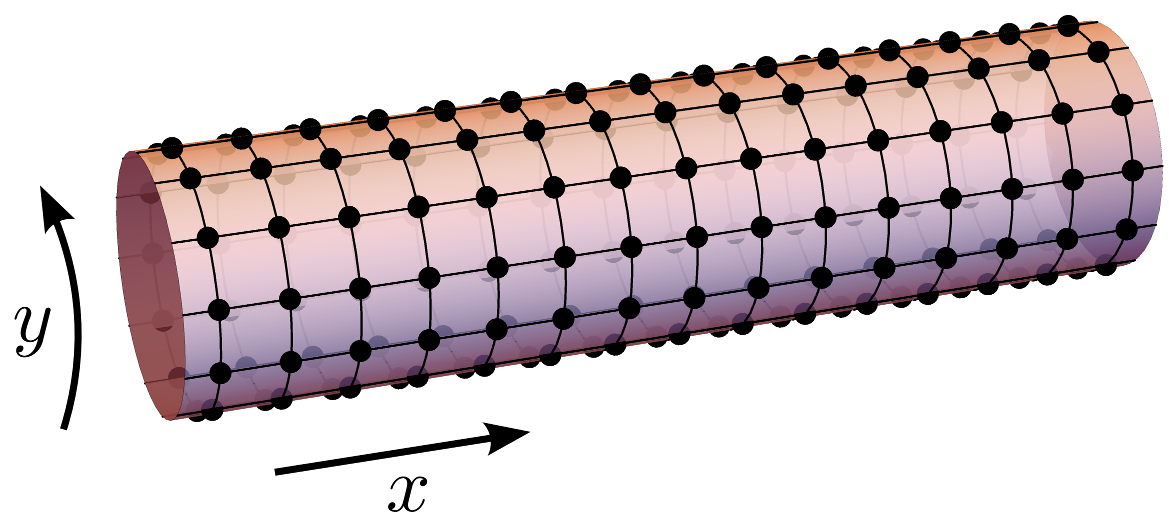


$N_y = 3$



$N_y = 4$

Prefer periodic boundary conditions in y, open boundary conditions in x:



- y direction is small: periodic helps
- x direction is large: DMRG doesn't like fully periodic

Figure from: Motruk, Zaletel, Mong, Pollmann, arxiv:1512.03318

Take advantage of quantum numbers

Very useful example is k_y momentum:

MPS path:

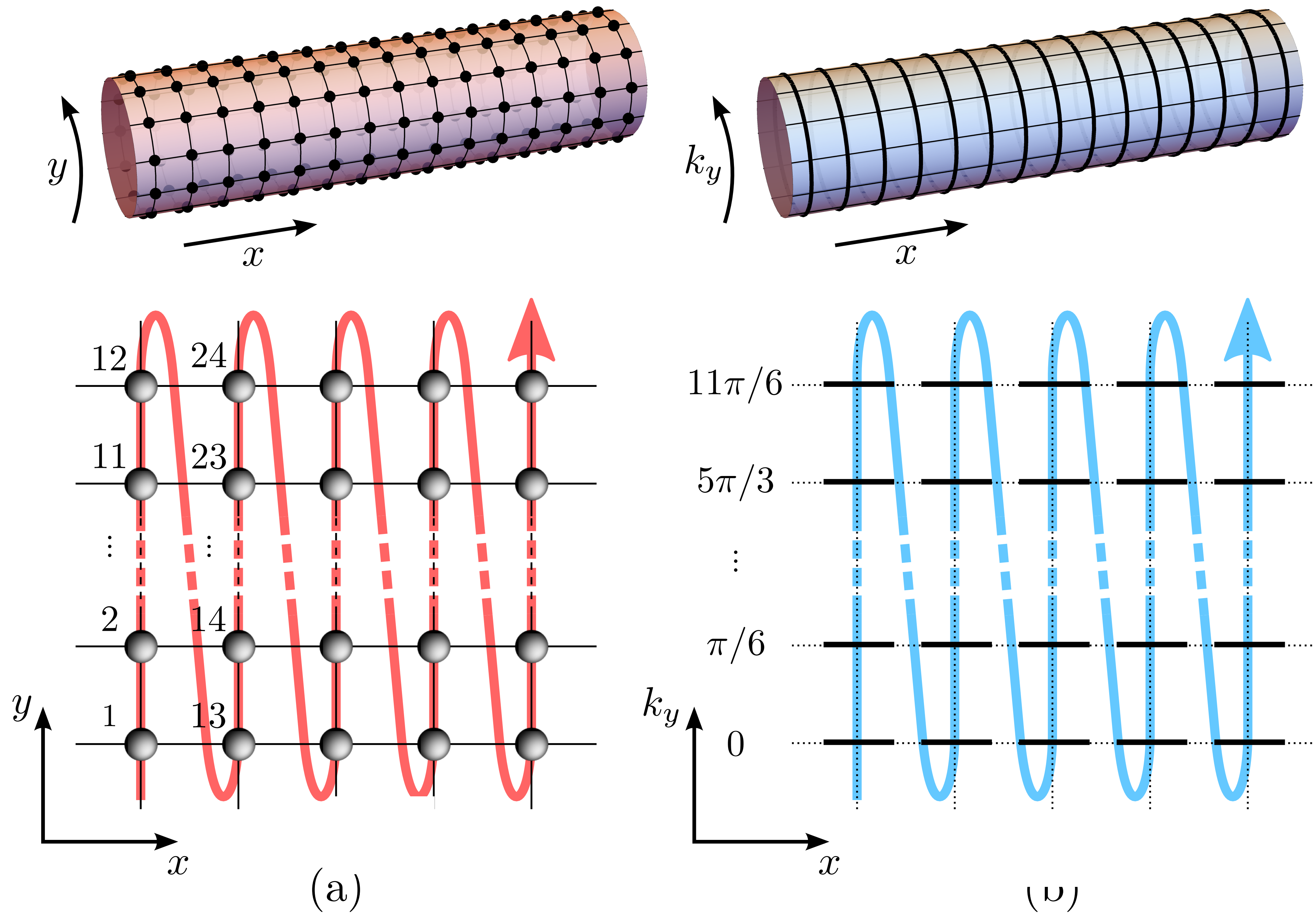


Figure from: Motruk, Zaletel, Mong, Pollmann, arxiv:1512.03318

Advanced DMRG Topics

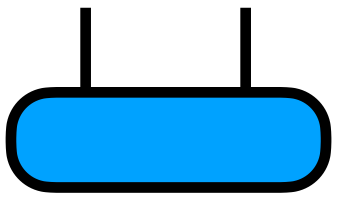
Let's discuss two more technical aspects of DMRG:

- Abelian quantum number symmetry

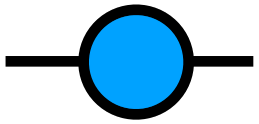
- fermions

Key example of Abelian quantum number is
particle number conservation

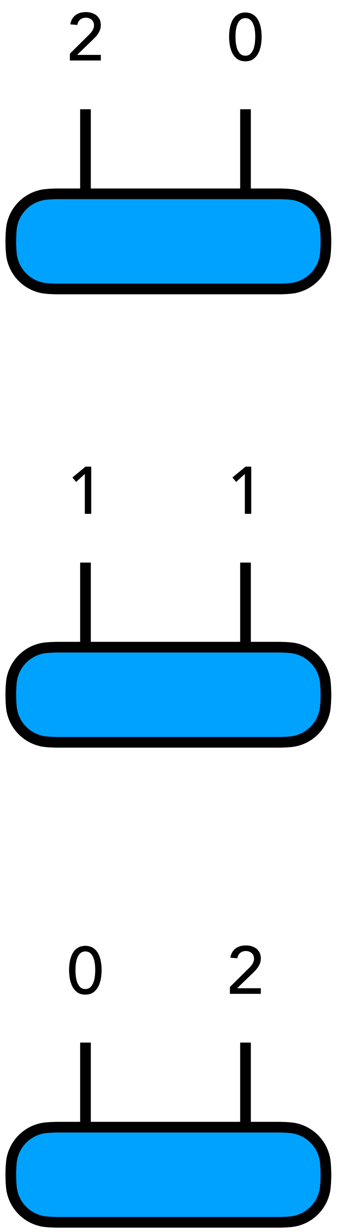
Consider two-site boson wavefunction:



It is equivalent to a matrix (two-index tensor):



Consider case of 2 particles:
possible configurations are



All other configurations must have zero amplitude

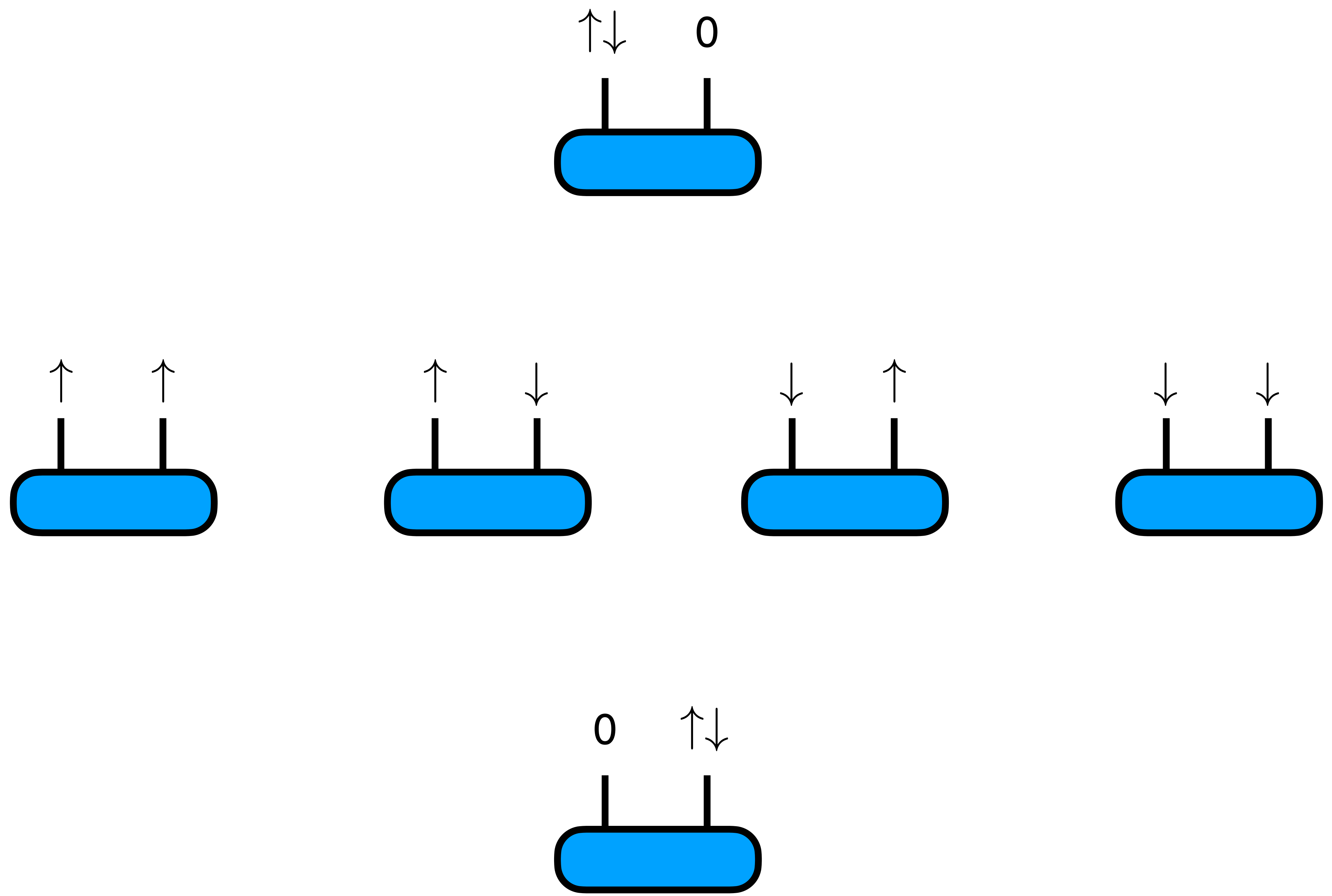
Matrix form of 2-particle wavefunction:

$$
\begin{array}{c|ccc|}
 & 0 & 1 & 2 \\
\hline
0 & 0 & 0 & \psi_{02} \\
1 & 0 & \psi_{11} & 0 \\
2 & \psi_{20} & 0 & 0
\end{array}
$$

Don't have to store 0's once we know particle number

Those entries remain **always** zero

Now consider 2 electrons ***with spin*** *(spin not conserved)*



All other configurations must have zero amplitude

Matrix form of 2-electron wavefunction:

	0	$\uparrow$	$\downarrow$	$\uparrow\downarrow$
0	0	0	0	$\psi_{0,\uparrow\downarrow}$
$\uparrow$	0	$\psi_{\uparrow,\uparrow}$	$\psi_{\uparrow,\downarrow}$	0
$\downarrow$	0	$\psi_{\downarrow,\uparrow}$	$\psi_{\downarrow,\downarrow}$	0
$\uparrow\downarrow$	$\psi_{\uparrow\downarrow,0}$	0	0	0

Non-zero elements & zero elements form ***blocks***

Matrix form of 2-electron wavefunction:

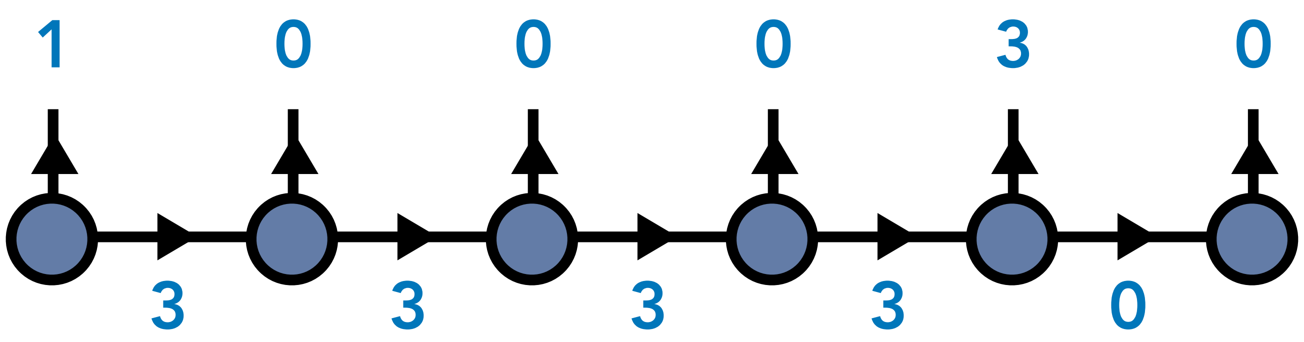
$$
\begin{array}{cc|cccc|}
 & & 0 & 1 & & 2 \\
 & & 0 & \uparrow & \downarrow & \uparrow\downarrow \\
\hline
0 & 0 & 0 & 0 & 0 & \psi_{0,\uparrow\downarrow} \\
1 & \uparrow & 0 & \psi_{\uparrow,\uparrow} & \psi_{\uparrow,\downarrow} & 0 \\
 & \downarrow & 0 & \psi_{\downarrow,\uparrow} & \psi_{\downarrow,\downarrow} & 0 \\
2 & \uparrow\downarrow & \psi_{\uparrow\downarrow,0} & 0 & 0 & 0
\end{array}
$$

Non-zero elements & zero elements form ***blocks***

Blocks associated to quantum number of each index

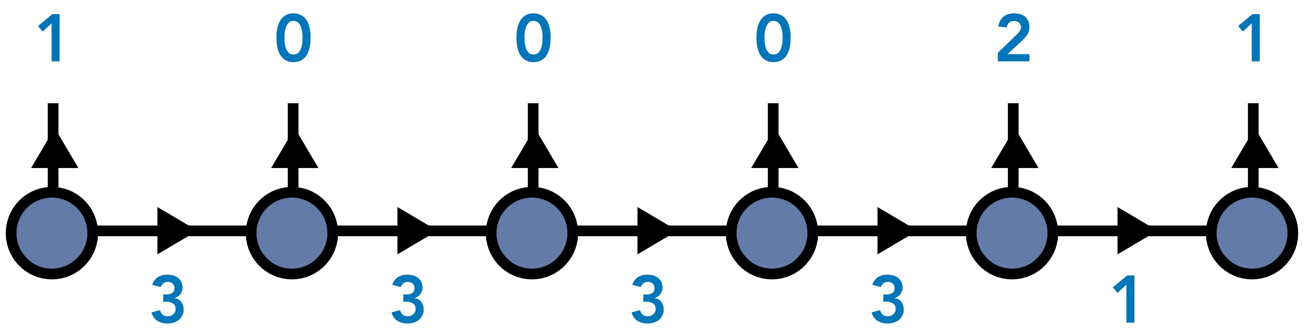
Block structure holds for tensors, and tensor networks

Helpful to put arrows on indices, corresponding to flux of particles:



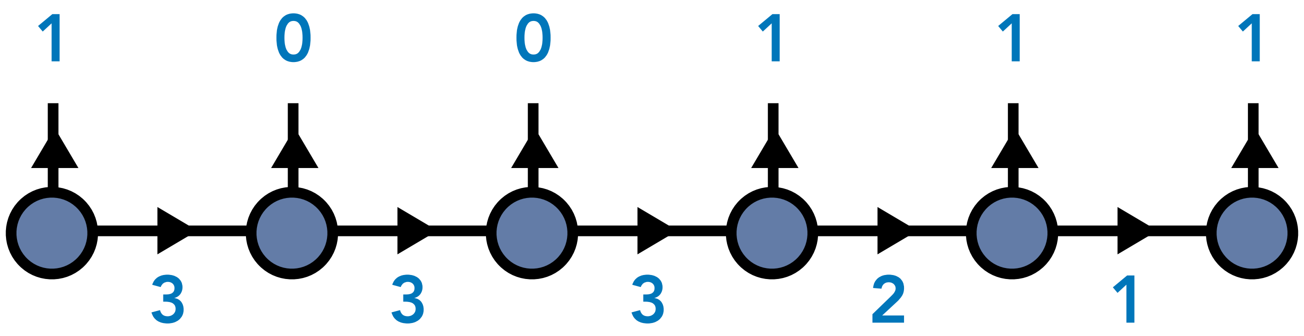
Block structure holds for tensors, and tensor networks

Helpful to put arrows on indices, corresponding to flux of particles:



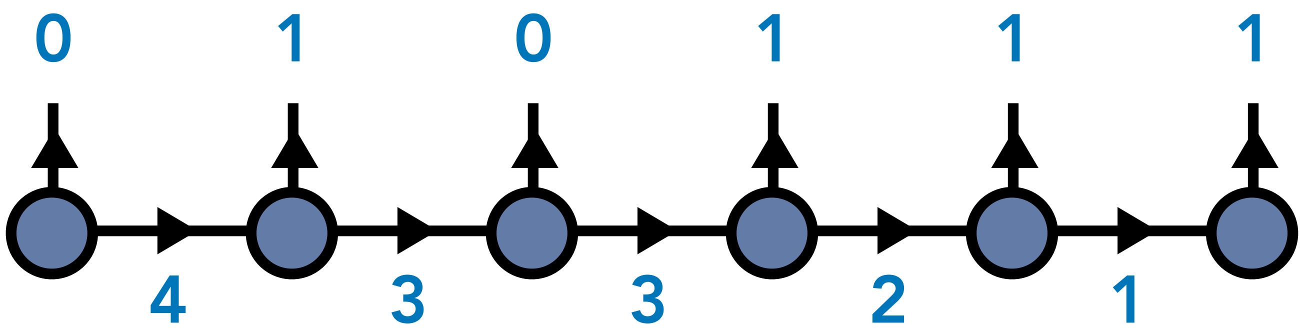
Block structure holds for tensors, and tensor networks

Helpful to put arrows on indices, corresponding to flux of particles:



Block structure holds for tensors,
and tensor networks

Helpful to put arrows on indices, corresponding to flux of particles:



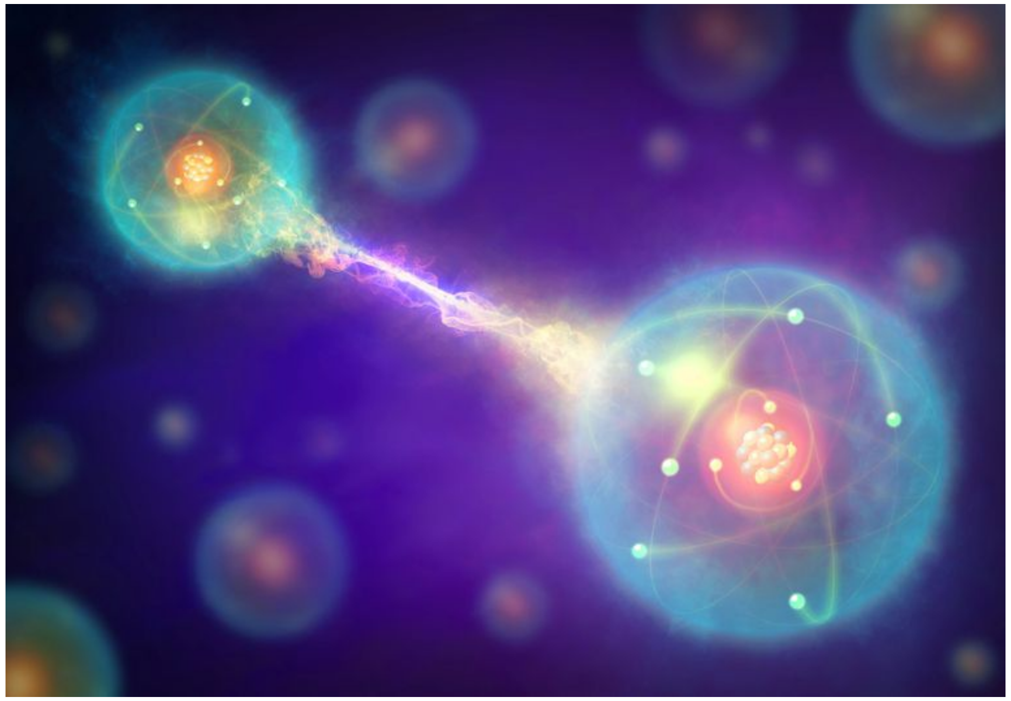
Gains from quantum numbers
& storing non-zero blocks only:

- treat important physical conservation laws
- save memory
- save computation time (less to contract / multiply)
- computational gains can be very large sometimes (10x)

Quantum numbers in ITensor:

just specify quantum numbers of physical indices, then perform algorithms – blocks are tracked for you!

Fermions are an important degree of freedom in condensed matter physics (electrons)



Credit: MARK GARLICK/SCIENCE PHOTO LIBRARY/Getty Images

Typical tensor network approach uses
second quantization

This means:

$$|\Psi\rangle = \psi^{s_1 s_2 s_3 s_4} |s_1 s_2 s_3 s_4\rangle \qquad s_j = 0, 1$$

Typical tensor network approach uses
second quantization

This means:

$$|\Psi\rangle = \psi^{s_1 s_2 s_3 s_4} |s_1 s_2 s_3 s_4\rangle \qquad s_j = 0, 1$$

$$= \psi^{s_1 s_2 s_3 s_4} \, (\hat{c}_1^\dagger)^{s_1} (\hat{c}_2^\dagger)^{s_2} (\hat{c}_3^\dagger)^{s_3} (\hat{c}_4^\dagger)^{s_4} |0\rangle$$

Typical tensor network approach uses
second quantization

This means:

$$|\Psi\rangle = \psi^{s_1 s_2 s_3 s_4} |s_1 s_2 s_3 s_4\rangle \qquad s_j = 0, 1$$

$$= \psi^{s_1 s_2 s_3 s_4} \underbrace{(\hat{c}_1^\dagger)^{s_1} (\hat{c}_2^\dagger)^{s_2} (\hat{c}_3^\dagger)^{s_3} (\hat{c}_4^\dagger)^{s_4} |0\rangle}$$

*all antisymmetry
handled in this part*

Typical tensor network approach uses
second quantization

This means:

$$|\Psi\rangle = \psi^{s_1 s_2 s_3 s_4} |s_1 s_2 s_3 s_4\rangle \qquad s_j = 0, 1$$

$$= \underbrace{\psi^{s_1 s_2 s_3 s_4}}_{\text{can be any tensor}} \underbrace{(\hat{c}_1^\dagger)^{s_1} (\hat{c}_2^\dagger)^{s_2} (\hat{c}_3^\dagger)^{s_3} (\hat{c}_4^\dagger)^{s_4} |0\rangle}_{\text{all antisymmetry handled in this part}}$$

can be any tensor

all antisymmetry handled in this part

Typical tensor network approach uses
second quantization

This means:

$$|\Psi\rangle = \psi^{s_1 s_2 s_3 s_4} |s_1 s_2 s_3 s_4\rangle \qquad s_j = 0,1$$

$$= \underbrace{\psi^{s_1 s_2 s_3 s_4}}_{\text{can be any tensor}} \underbrace{(\hat{c}_1^\dagger)^{s_1} (\hat{c}_2^\dagger)^{s_2} (\hat{c}_3^\dagger)^{s_3} (\hat{c}_4^\dagger)^{s_4} |0\rangle}_{\text{all antisymmetry handled in this part}}$$

can be any tensor

all antisymmetry handled in this part

No need to antisymmetrize (or symmetrize) amplitude tensor represented by tensor network

When do the signs enter in?

When using operators:

- applying Hamiltonian
- computing observables

When do the signs enter in?

When using operators:

- applying Hamiltonian
- computing observables

$$\hat{c}_2 \left[\psi^{s_1 s_2 s_3 s_4} \; (\hat{c}_1^\dagger)^{s_1} (\hat{c}_2^\dagger)^{s_2} (\hat{c}_3^\dagger)^{s_3} (\hat{c}_4^\dagger)^{s_4} \right] |0\rangle$$

When do the signs enter in?

When using operators:

- applying Hamiltonian
- computing observables

$$\hat{c}_2 \left[\psi^{s_1 s_2 s_3 s_4} \, (\hat{c}_1^\dagger)^{s_1} (\hat{c}_2^\dagger)^{s_2} (\hat{c}_3^\dagger)^{s_3} (\hat{c}_4^\dagger)^{s_4} \right] |0\rangle$$

Sign of result will depend on value of s_1 index

Fermion minus signs & tensor networks

Programming approaches – 3 alternatives:

- map fermionic operators to non-local bosonic operators (Jordan-Wigner transformation);
  work only with these

- choose canonical, reference ordering of sites and always permute basis states to this order

- anti-commuting tensor indices (*newest approach*)

Jordan-Wigner string approach to fermions

Consider fermionic operators: $\hat{c}_i$ $\{\hat{c}_i, \hat{c}_j^\dagger\} = \delta_{ij}$

Now define commuting (bosonic) operators: $\hat{b}_i$ $[\hat{b}_i, \hat{b}_j^\dagger] = \delta_{ij}$

$$\hat{c}_i^\dagger |0000\rangle = |0010\rangle \qquad \hat{b}_i^\dagger |0000\rangle = |0010\rangle$$

site i site i

Jordan-Wigner string approach to fermions

The b's are related to the c's as follows:

$$\hat{c}_i = \hat{F}_1 \hat{F}_2 \cdots \hat{F}_{i-1} \; \hat{b}_i$$

$$\hat{F}_j = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$$

So expressions like $\hat{c}_i^\dagger \hat{c}_{i+2}$
become:

$$\hat{c}_i^\dagger \hat{c}_{i+2} = \hat{b}_i^\dagger \; \hat{F}_{i+1} \; \hat{b}_{i+2}$$

Jordan-Wigner string approach to fermions

Fortunately this works great for Hamiltonian MPOs!

Consider hopping term:

$$(c_i^\dagger c_j + c_j^\dagger c_i) = (b_i^\dagger F_{i+1} F_{i+2} \cdots F_{j-1} b_j + b_i F_{i+1} F_{i+2} \cdots F_{j-1} b_j^\dagger)$$

Internally, MPOs encode terms like $c_i^\dagger c_j$ as:

$$c_i^\dagger I_{i+1} I_{i+2} \cdots I_{j-1} c_j$$

So just switch identity operators between c's into F operators:

$$b_i^\dagger F_{i+1} F_{i+2} \cdots F_{j-1} b_j$$

Jordan-Wigner string approach to fermions

Putting F operators into Hamiltonian "just works" for correct energy within DMRG

Local measurements also simple

Correlation functions do require explicitly putting F (string) operators, but this is not hard to do

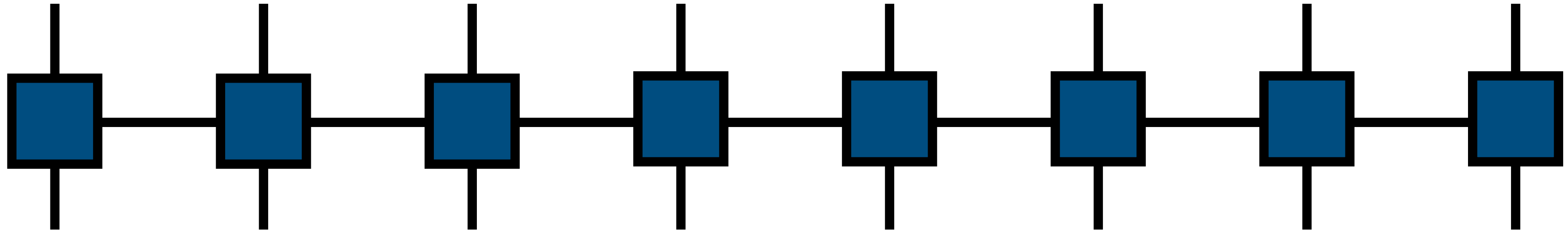
To treat fermions with spin, think of up and down fermions as neighboring "sites" of spinless fermions

Summary

- DMRG a powerful approach for 2D systems
- Applications to magnetism & strongly correlated electrons
- Quantum numbers can be exploited to make tensor networks block-sparse
- Fermions can be treated in DMRG with Jordan-Wigner string operators

Matrix Product Operators

Idea of a matrix product operator (MPO):
chain of tensors like an MPS, but two sets of indices (up and down; bra and ket) just like an operator



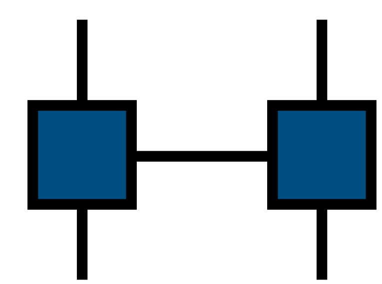
Very useful for algorithms involving MPS, such as DMRG

To motivate MPO construction, consider a two-site operator

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = S_1^z S_2^z + \frac{1}{2} S_1^+ S_2^- + \frac{1}{2} S_1^- S_2^+$$

Write as dot product of operator-valued vectors

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = \begin{bmatrix} S_1^z & \frac{1}{2} S_1^+ & \frac{1}{2} S_1^- \end{bmatrix} \begin{bmatrix} S_2^z \\ S_2^- \\ S_2^+ \end{bmatrix} =$$

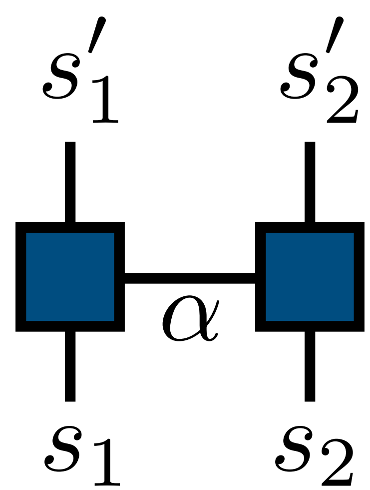


To motivate MPO construction, consider a two-site operator

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = S_1^z S_2^z + \frac{1}{2} S_1^+ S_2^- + \frac{1}{2} S_1^- S_2^+$$

Write as dot product of operator-valued vectors

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = \begin{bmatrix} S_1^z & \frac{1}{2} S_1^+ & \frac{1}{2} S_1^- \end{bmatrix}_\alpha \begin{bmatrix} S_2^z \\ S_2^- \\ S_2^+ \end{bmatrix} =$$



More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z$$

More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad = S_1^z S_2^z I_3 + I_1 S_2^z S_3^z$$

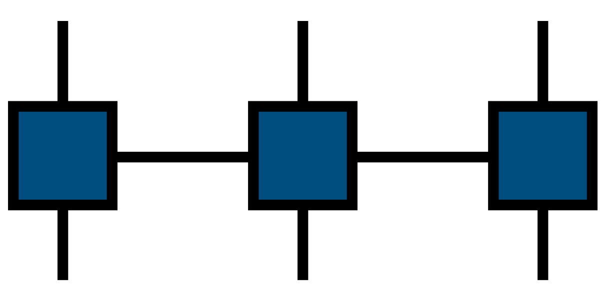
More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad = S_1^z S_2^z I_3 + I_1 S_2^z S_3^z$$

Can write as

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \begin{bmatrix} I_2 & 0 & 0 \\ S_2^z & 0 & 0 \\ 0 & S_2^z & I_2 \end{bmatrix} \begin{bmatrix} I_3 \\ S_3^z \\ 0 \end{bmatrix}$$



More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad (= S_1^z S_2^z I_3 + I_1 S_2^z S_3^z)$$

Can write as

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \left(\begin{bmatrix} I_2 & 0 & 0 \\ S_2^z & 0 & 0 \\ 0 & S_2^z & I_2 \end{bmatrix} \begin{bmatrix} I_3 \\ S_3^z \\ 0 \end{bmatrix} \right) = \begin{bmatrix} I_2 \; I_3 \\ S_2^z \; I_3 \\ S_2^z \; S_3^z \end{bmatrix}$$

More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad (= S_1^z S_2^z I_3 + I_1 S_2^z S_3^z)$$

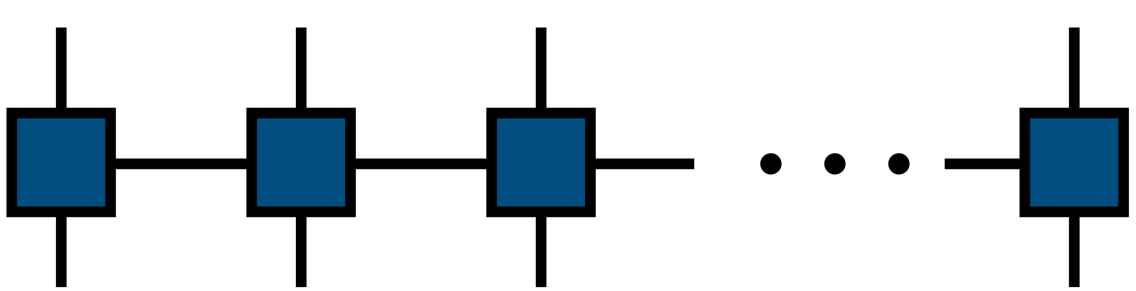
Can write as

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \begin{bmatrix} I_2 \, I_3 \\ S_2^z \, I_3 \\ S_2^z \, S_3^z \end{bmatrix} = S_1^z S_2^z I_3 + I_1 S_2^z S_3^z$$

Chaining the pattern will give Hamiltonian for arbitrarily big system

$$H = \sum_j S^z_j S^z_{j+1}$$

$$\begin{bmatrix} 0 & S^z_1 & I_1 \end{bmatrix} \begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix} \begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix} \cdots \begin{bmatrix} I_N \\ S^z_N \\ 0 \end{bmatrix}$$



Why this pattern?

$$H = \sum_j S_j^z S_{j+1}^z$$

$$\begin{bmatrix} I_j & 0 & 0 \\ S_j^z & 0 & 0 \\ 0 & S_j^z & I_j \end{bmatrix}$$

View as a "machine" or "automaton"

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Result:

View as a "machine" or "automaton"

Start in state 3 →

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Result:

View as a "machine" or "automaton"

Start in state 3 →

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Choose an operator

Result:

View as a "machine" or "automaton"

Start in state 3 →

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & \boxed{I_1} \end{bmatrix}$$

Choose an operator

Result: I_1

View as a "machine" or "automaton"

State 3

Start in state 3

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & \boxed{I_1} \end{bmatrix}$$

Choose an operator

Result: I_1

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Result: I_1

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Choose an operator

Result: I_1

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & \boxed{I_2} \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

State 3

State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

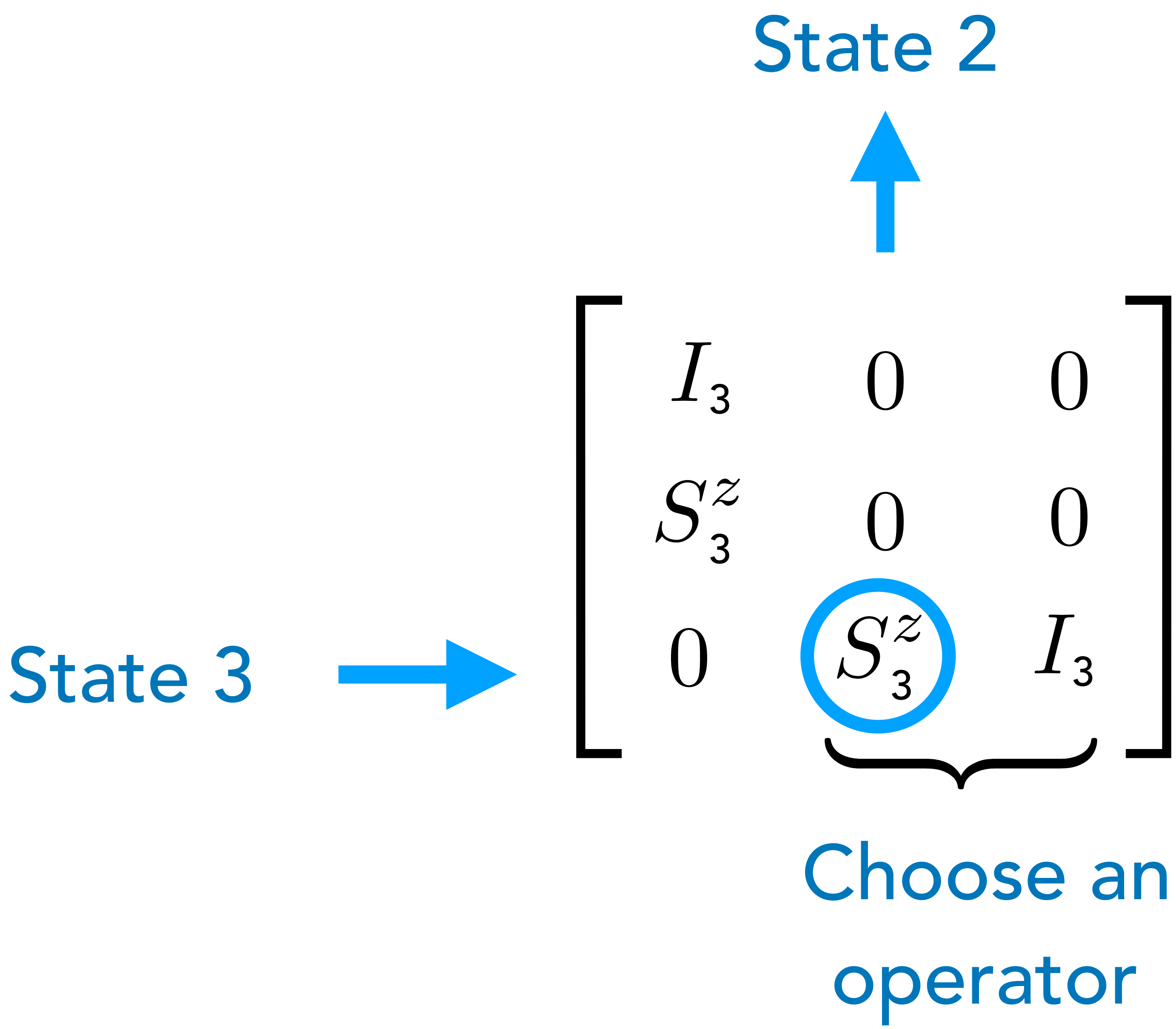
State 3 →

$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2 \quad S^z_3$

View as a "machine" or "automaton"



$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3$

View as a "machine" or "automaton"

State 2 → $$\begin{bmatrix} I_4 & 0 & 0 \\ S^z_4 & 0 & 0 \\ 0 & S^z_4 & I_4 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3$

View as a "machine" or "automaton"

State 2 →

$$\begin{bmatrix} I_4 & 0 & 0 \\ S^z_4 & 0 & 0 \\ 0 & S^z_4 & I_4 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1

State 2

$$\begin{bmatrix} I_4 & 0 & 0 \\ S^z_4 & 0 & 0 \\ 0 & S^z_4 & I_4 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1 →

$$\begin{bmatrix} I_5 & 0 & 0 \\ S^z_5 & 0 & 0 \\ 0 & S^z_5 & I_5 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1

State 1

$$\begin{bmatrix} I_5 & 0 & 0 \\ S^z_5 & 0 & 0 \\ 0 & S^z_5 & I_5 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1

State 1

$$\begin{bmatrix} I_5 & 0 & 0 \\ S^z_5 & 0 & 0 \\ 0 & S^z_5 & I_5 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4 \quad I_5$

Familiar 1D Hamiltonians as MPOs

Transverse-field Ising model

$$\begin{bmatrix} I_j & & \\ \sigma^z_j & & \\ -h\sigma^x_j & \sigma^z_j & I_j \end{bmatrix}$$

$$H = \sum_j \sigma^z_j \sigma^z_{j+1} - h\sigma^x_j$$

Heisenberg model

$$\begin{bmatrix} I_j & & & & \\ S^+_j & & & & \\ S^-_j & & & & \\ S^z_j & & & & \\ 0 & \frac{1}{2}S^-_j & \frac{1}{2}S^+_j & S^z_j & I_j \end{bmatrix}$$

$$H = \sum_j \mathbf{S}_j \cdot \mathbf{S}_{j+1}$$

MPOs can even capture "long range" interactions

$$\begin{bmatrix} I_j & & \\ \sigma^z_j & \lambda I_j & \\ & \lambda\sigma^z_j & I_j \end{bmatrix}$$

$$H = \sum_{i<j} \lambda^{j-i} \, \sigma^z_i \sigma^z_j$$

MPOs can even capture "long range" interactions

$$\begin{bmatrix} I_j & & & \\ \sigma_j^z & \lambda_1 I_j & & \\ \sigma_j^z & & \lambda_2 I_j & \\ & \lambda_1 \sigma_j^z & \lambda_2 \sigma_j^z & I_j \end{bmatrix}$$

$$H = \sum_{i<j} \left(\lambda_1^{j-i} + \lambda_2^{j-i} \right) \sigma_i^z \sigma_j^z$$