

---

# SOLVING THE MAXIMUM INDEPENDANT SET PROBLEM WITH TENSOR NETWORKS

**Jin-Guo Liu**

Harvard University  
jinguoliu@g.harvard.edu

**Xun Gao**

Harvard University  
xungao@g.harvard.edu

## ABSTRACT

Solving the maximum independent set size problem: the maximum independent set size, the independence polynomial and the optimal configuration.

## 1 TECHNICAL GUIDE

**OMEinsum** a package for einsum,

**OMEinsumContractionOrders** a package for finding the optimal contraction order for einsum  
<https://github.com/Happy-Diode/OMEinsumContractionOrders.jl>,

**TropicalGEMM** a package for efficient tropical matrix multiplication (compatible with OMEinsum),

**TropicalNumbers** a package providing tropical number types and tropical algebra, one o the dependency of TropicalGEMM,

**LightGraphs** a package providing graph utilities, like random regular graph generator,

**Polynomials** a package providing polynomial algebra and polynomial fitting,

**Mods and Primes** packages providing finite field algebra and prime number generators.

One can install these packages by opening a julia REPL, type `]` to enter the `pkg>` mode and type, e.g.

```
pkg> add OMEinsum
```

An example of computing the maximum independant set size

```
julia> using OMEinsum, LightGraphs, TropicalNumbers

julia> n, k = 6, 3
(6, 3)

julia> g = LightGraphs.random_regular_graph(n, k)
{6, 9} undirected simple Int64 graph

julia> ixs = [minmax(e.src,e.dst) for e in LightGraphs.edges(g)]
9-element Vector{Tuple{Int64, Int64}}:
 (1, 2)
 (1, 4)
 (1, 5)
 (2, 3)
 (2, 6)
 (3, 5)
 (3, 6)
 (4, 5)
 (4, 6)
```

```

julia> code = EinCode((ixs..., [(i,) for i in LightGraphs.vertices(g)]...), ())
12, 14, 15, 23, 26, 35, 36, 45, 46, 1, 2, 3, 4, 5, 6 ->

julia> optimized_code = optimize_greedy(code, Dict{<int, int>{i=2 for i=1:6}})
46, 46 ->
├─ 46, 6 -> 46
│ └─ 6
│   └─ 46
└─ 254, 526 -> 46
   └─ 256, 62 -> 526
      └─ 26, 2 -> 62
         └─ 2
           └─ 26
              └─ 23, 563 -> 256
                 └─ 35, 63 -> 563
                    └─ 36, 3 -> 63
                       └─ 35, 5 -> 35
                          └─ 23
                             └─ 241, 145 -> 254
                                └─ 15, 54 -> 145
                                   └─ 45, 4 -> 54
                                      └─ 4
                                         └─ 45
                                            └─ 15
                                               └─ 21, 14 -> 241
                                                  └─ 14
                                                     └─ 12, 1 -> 21
                                                        └─ 1
                                                           └─ 12

julia> function mis_contract(code::OMEinsum.NestedEinsum, x::T) where T
    tensors = map(OMEinsum.getixs(Iterators.flatten(code))) do ix
        @assert length(ix) == 1 || length(ix) == 2
        length(ix) == 1 ? [one(T), x] : [one(T) one(T); one(T) zero(T)]
    end
    code(tensors...)
end
mis_contract (generic function with 1 method)

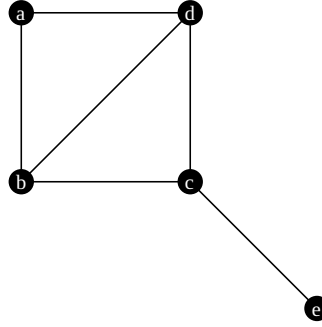
julia> result = mis_contract(optimized_code, Tropical{1.0})
0-dimensional Array{TropicalF64, 0}:
2.0,

```

For larger graph, we hight recommend using two additional packages, TropicalGEMM for BLAS speed tropical matrix multiplication and OMEinsumContractionOrders for hyper optimized contraction order (the KaHyPar approach) (??).

## 2 COMPUTING DEGENERACY

Consider the following 5 node graph.

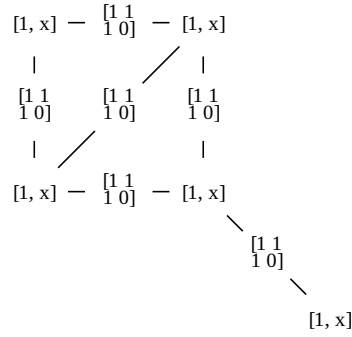


If put a vector of size 2 at each vertex

$$\begin{pmatrix} 1 \\ x \end{pmatrix}, \quad (1)$$

where  $x$  is a variable, and a matrix of size  $2 \times 2$  at each edge

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}. \quad (2)$$



Then we contract this graph with  $\text{einsum } a, b, c, d, ab, ad, bc, bd, cd, ce \rightarrow$ . The result is a scalar function of  $x$ , it corresponds to the independence polynomial  $I(G, x)$ .

$$I(G, x) = \sum_{k=1}^{\alpha(G)} a_k x^k, \quad (3)$$

where  $a_k$  is the degeneracy for independent set size  $k$ .

## 2.1 SYMBOLIC COMPUTING

The most straight forward approach is the symbolic computing, where we store a polynomial of  $x$  as a vector of factors. We define the algebra between these vectors of factors and insert them to the original tensor network contraction algorithm. However, this method suffers from a space overhead that proportional to the maximum independent set size.

## 2.2 POLYNOMIAL FITTING

Let  $m$  be the maximum independent set size and  $X$  be a set of  $m + 1$  random real numbers, e.g.  $\{0, 1, 2, \dots, m\}$ . We compute the einsum contraction for each  $x \in X$  and obtain the following relations

$$a_0 + a_1 x_1 + a_1 x_1^2 + \dots + a_m x_1^m = y_0 \quad (4)$$

$$a_0 + a_1 x_2 + a_2 x_2^2 + \dots + a_m x_2^m = y_1 \quad (5)$$

$$\dots \quad (6)$$

$$a_0 + a_1 x_m + a_2 x_m^2 + \dots + a_m x_m^m = y_m \quad (7)$$

The polynomial fitting between  $X$  and  $Y = \{y_0, y_1, \dots, y_m\}$  gives us the factors.

## 2.3 FOURIER TRANSFORMATION

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{pmatrix} \quad (8)$$

$$\begin{pmatrix} 1 & r\omega & r^2\omega^2 & \dots & r^m\omega^m \\ 1 & r\omega^2 & r^2\omega^4 & \dots & r^m\omega^{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r\omega^m & r^2\omega^{2m} & \dots & r^m\omega^{m^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{pmatrix} \quad (9)$$

When  $r = 1$ , the left side is a DFT matrix. We can obtain the factors using the relation  $\vec{d} = \text{FFT}^{-1}(\omega) \cdot \vec{y}$ . In the special case that  $\omega = e^{-2\pi i/(m+1)}$ , it is directly solvable with inverse fast fourier transformation algorithm in package FFTW.

$$\text{FFT}(\omega) \cdot \vec{d}_r = \vec{y} \quad (10)$$

where  $(\vec{d}_r)_k = a_k r^k$ , by choosing different  $r$ , we can obtain better precision in low independent set size region ( $\omega < 1$ ) and high independent set size region ( $\omega > 1$ ).

## 2.4 FINITE FIELD ALGEBRA

## 3 UTILIZING THE SPARSITY

Tensor network compression is an important tool to utilize sparsity.

We contract the tensors in a subregion  $R \subseteq G$  of a graph  $G$ , and obtain a resulting tensor  $A$  of rank  $|C|$ , where  $C$  is the set of vertex tensors at the cut. The maximum independent set size in this region with boundary configuration  $\sigma \in \{0, 1\} \otimes |C|$  is  $A_\sigma$ . We say an entry  $A_{\sigma_a}$  is “better” than  $A_{\sigma_b}$  if

$$(\sigma_a \wedge \sigma_b = \sigma_a) \wedge (A_{\sigma_a} \geq A_{\sigma_b}), \quad (11)$$

where  $\wedge$  is a bitwise and operations. The first term means that whenever a bit in  $\sigma_a$  has boolean value 1, the corresponding bit in  $\sigma_b$  is also 1. While the second term means the maximum independent set size with boundary configuration fixed to  $\sigma_a$  is not less than that fixed to  $\sigma_b$ . The word “better” means the best solution with boundary configuration  $\sigma_a$  is never worse than that with  $\sigma_b$ . When Eq. ?? holds, It is easy to see that if  $\sigma_b \cup \overline{\sigma_b}$  is one of the solutions for maximum independent sets in  $G$ ,  $\sigma_a \cup \overline{\sigma_b}$  is also a solution.

### 3.1 THE TENSOR NETWORK COMPRESSION DETECTS BRANCHING RULES AUTOMATICALLY

*max-size rule* The resulting tensor of Tropical tensor contraction stores only the local independent set size that maximizes the internal degrees on freedom. Where the internal degree of freedom are edges that does not connect to tensors not contracted.

*max-interior rule* The resulting tensor of Tropical tensor contraction stores only the local independent set size that maximizes the internal degrees on freedom.

We are going to verify the Lemmas used in branching algorithm in book (?).

**Rule 1.** *If a vertex  $v$  is in an independent set  $I$ , then none of its neighbors can be in  $I$ . On the other hand, if  $I$  is a maximum (and thus maximal) independent set, and thus if  $v$  is not in  $I$  then at least one of its neighbors is in  $I$ .*

Contract  $N[v]$ , each entry of the resulting tensor of rank  $|N(v)|$  corresponds to a locally maximized independent set size with fixed boundary configuration. If the boundary configuration is a bit string of 0s,  $\sigma_v$  will take value 1 to maximize the local independent set size.

**Rule 2.** *Let  $G = (V, E)$  be a graph, let  $v$  and  $w$  be adjacent vertices of  $G$  such that  $N[v] \subseteq N[w]$ . Then*

$$\alpha(G) = \alpha(G \setminus w). \quad (12)$$

Contract  $N[w]$ , both  $\{v, w\}$  disappear from the tensor indices because they are inner degrees of freedom. If  $w$  is one, then  $N[v]$  are all zeros, the resulting tensor element can not be larger than setting  $v = 1$  and  $w = 0$ . By the maximization rule, the local tensor does not change if we remove  $w$ .

**Rule 3.** *Let  $G = (V, E)$  be a graph and let  $v$  be a vertex of  $G$ . If no maximum independent set of  $G$  contains  $v$  then every maximum independent set of  $G$  contains at least two vertices of  $N(v)$ .*

**Rule 4.** *Let  $G = (V, E)$  be a graph and  $v$  a vertex of  $G$ . Then*

$$\alpha(G) = \max(1 + \alpha(G \setminus N[v]), \alpha(G \setminus (M(v) \cup \{v\}))). \quad (13)$$

Here,  $M(v)$  is the set of mirrors of  $v$  in  $G$ . A vertex  $w \in N^2(v)$  is called a mirror of  $v$  if  $N(v) \setminus N(w)$  is a clique. This rule states that if  $v$  is not in  $M$ , there exists an MIS  $I$  that  $M(v) \notin I$ . otherwise, there must be one of  $N(v)$  in the MIS (maximization rule). If  $w$  is in  $I$ , then none of  $N(v) \cap N(w)$  is in  $I$ , then there must be one of node in the clique  $N(v) \setminus N(w)$  in  $I$  (maximization rule), since clique has at most one node in the MIS, the tensor compression will eliminate this solution by moving the occupied node to the interior. Hence, the tensor compression rule do captures the mirror rule.

**Rule 5.** *Let  $G = (V, E)$  be a graph and  $v$  be a vertex of  $G$  such that  $N[v]$  is a clique. Then*

$$\alpha(G) = 1 + \alpha(G \setminus N[v]). \quad (14)$$

**Rule 6.** *Let  $G$  be a graph, let  $S$  be a separator of  $G$  and let  $I(S)$  be the set of all subsets of  $S$  being an independent set of  $G$ . Then*

$$\alpha(G) = \max_{A \in I(S)} |A| + \alpha(G \setminus (S \cup N[A])). \quad (15)$$

**Rule 7.** *Let  $G = (V, E)$  be a disconnected graph and  $C \subseteq V$  a connected component of  $G$ . Then*

$$\alpha(G) = \alpha(G[C]) + \alpha(G \setminus C). \quad (16)$$

$$if |V| = 0 then return 0 if  $\exists v \in V$  with  $d(v) \leq 1$  then return  $1 + \max_{A \in I(S)} |A| + \alpha(G \setminus (S \cup N[A]))$  if  $\exists v \in V$  with  $d(v) = 2$  then let  $u_1$  and  $u_2$  be the neighbors of  $v$  and let  $C_1$  and  $C_2$  be the components of  $G \setminus \{v\}$  containing  $u_1$  and  $u_2$  respectively. Then return  $\max(\alpha(G[C_1]) + \alpha(G[C_2]), \alpha(G \setminus \{v\}))$  (17)$$