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

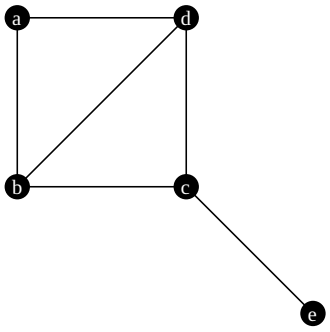
```
julia> using OMEinsum, LightGraphs, TropicalNumbers

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)]...), ())
1∘2, 1∘4, 1∘5, 2∘3, 2∘6, 3∘5, 3∘6, 4∘5, 4∘6, 1, 2, 3, 4, 5, 6 ->

julia> optimized_code = optimize_greedy(code, Dict([i=>2 for i=1:6]))
4∘6, 4∘6 ->
├─ 4∘6, 6 -> 4∘6
│  ├─ 6
│  └─ 4∘6
└─ 2∘5∘4, 5∘2∘6 -> 4∘6
   ├─ 2∘5∘6, 6∘2 -> 5∘2∘6
   │  ├─ 2∘6, 2 -> 6∘2
   │  │  ├─ 2
   │  │  └─ 2∘6
   │  └─ 2∘3, 5∘6∘3 -> 2∘5∘6
   │     ├─ 3∘5, 6∘3 -> 5∘6∘3
   │     │  ├─ 3∘6, 3 -> 6∘3
   │     │  │  ┆
   │     │  │
   │     │  └─ 3∘5, 5 -> 3∘5
   │     │     ┆
   │     │
   │     └─ 2∘3
   └─ 2∘4∘1, 1∘4∘5 -> 2∘5∘4
      ├─ 1∘5, 5∘4 -> 1∘4∘5
      │  ├─ 4∘5, 4 -> 5∘4
      │  │  ├─ 4
      │  │  └─ 4∘5
      │  └─ 1∘5
      └─ 2∘1, 1∘4 -> 2∘4∘1
         ├─ 1∘4
         └─ 1∘2, 1 -> 2∘1
            ├─ 1
            └─ 1∘2


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}:
2.0ₜ
```

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 (Schlag, 2020) approach) (Gray & Kourtis, 2021; Pan & Zhang, 2021).

2 Computing degeneracy

Consider the following 5 node graph.

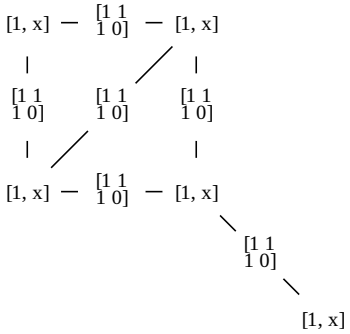


If put a vector of size 2 at each vertex

$$\begin{pmatrix}1\\x\end{pmatrix}, \tag{1}$$

were x is a variable, and a matrix of size 2×2 at each edge

$$\begin{pmatrix}1 & 1\\1 & 0\end{pmatrix}. \tag{2}$$



Then we contract this graph with einsum `a,b,c,d,ab,ad,bc,bd,cd,ce->`. The result is a scalar function of x, it corresponds to the independence polynomial (Ferrin, 2014).

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

where a_k is the the degeneracy for independant 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 propotional to the maximum independant 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. $\{0, 1, 2, \ldots, m\}$. We compute the einsum contraction for each $x \in X$ and obtain the following relations

$$a_0 + a_1 x_1 + a_1 x_1^2 + \ldots + a_m x_1^m = y_0 \tag{4}$$
$$a_0 + a_1 x_2 + a_2 x_2^2 + \ldots + a_m x_2^m = y_1 \tag{5}$$
$$\ldots \tag{6}$$
$$a_0 + a_1 x_m + a_2 x_m^2 + \ldots + a_m x_m^m = y_m \tag{7}$$

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

2.3 Fourier transformation

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

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

When $r = 1$, the left side is a DFT matrix. We can obtain the factors using the relation $\vec{a} = \mathrm{FFT}^{-1}(\omega) \cdot \vec{y}$. 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`.

$$\mathrm{FFT}(\omega) \cdot \vec{a}_r = \vec{y} \tag{10}$$

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

2.4 Finite field algebra

It is possible to compute the independence polynomials exactly using basic integer types only, even when the degeneracy is bigger than that can be represented by 64 bit integers. What we need is computing with the finite field algebra. In a finite field algebra $GF(p)$, we have the following observations

1. One can still use Gaussian elimination (Golub & Van Loan, 2013) to solve the linear equations in a finite field, since the multiplicative inverse is properly defined in a finite field,
2. Let remainders of an integer x over a set of coprime integers $\{p_1, p_2, \ldots, p_n\}$ being $\{x \pmod{p_1}, x \pmod{p_2}, \ldots, x \pmod{p_n}\}$, then $\{x \pmod{p_1 \times p_2 \times \ldots \times p_n}\}$ can be obtained using the chinese remainder theorem.

With these observations, we use Algorithm 1 to compute degeneracies. In the algorithm, except the computation of chinese remainder theorem, all computations are done with integers with fixed width.

3 Utilizing the sparsity

Tensor network compression is an important tool to utilize sparsity.

Algorithm 1: Compute independence polynomial exactly without integer overflow

Let $P = 1$, $X = 0, 1, 2, \ldots, m$ and $\hat{X}_{ij} = X_i^j$, where $i, j = \{0, 1, \ldots m\}$;
while *true* **do**
  compute the largest prime 64 bit integer p that $\gcd(p, P) = 1$;
  compute the tensor network contraction on $GF(p)$ and obtain $Y \pmod p = \{y_0 \pmod p, y_1 \pmod p, \ldots, y_m \pmod p\}$;
  $A \pmod p = \{a_0 \pmod p, a_1 \pmod p, \ldots, a_m \pmod p\}$ = gaussian_elimination($\hat{X}, Y \pmod p$);
  $A \pmod{P \times p}$ = chinese_remainder($A \pmod P, A \pmod p$);
  if $A \pmod P = A \pmod{P \times p}$ **then**
    return $A \pmod P$;
  end
  $P = P \times p$;
end

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 vertice tensors at the cut. The maximum independant set size in this region with boundary configuration $\sigma \in \{0, 1\} \otimes |C|$ is A_σ. We say an entry A_{σ_a} is "better" than A_{σ_b} if

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

where $\wedge$ is a bitwise and operations. The first term means that whenever a bit in σ_a has boolean value 1, the corresponding bit in σ_b is also 1. While the second term means the maximum independant set size with boundary configuration fixed to σ_a is not less than that fixed to σ_b. The word "better" means the best solution with boundary configuration σ_a is never worse than that with σ_b. When Eq. 11 holds, It is easy to see that if $\sigma_b \cup \overline{\sigma_b}$ is one of the solutions for maximum independant 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 independant 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 independant set size that maximizes the internal degrees on freedom.

We are going the verify the Lemmas used in branching algorithm in book (Fomin & Kaski, 2013).

***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 independant set size with fixed boundary configuration. If the boundary configuration is a bit string of 0s, σ_v will takes value 1 to maximize the local independant 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*

$$\alpha(G) = \alpha(G \backslash w). \tag{12}$$

Contract $N[w]$, both $\{v, w\}$ disapear 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*

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

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 $N(v)\backslash N(w)$ 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 $N(v)\backslash N(w)$ in I (maximization rule), since clique has at most one node in the MIS, the tensor compression will eliminate this solution by moving the occuppied 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*

$$\alpha(G) = 1 + \alpha(G\backslash N[v]). \tag{14}$$

***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*

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

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

$$\alpha(G) = \alpha(G[C]) + \alpha(G\backslash C)). \tag{16}$$

$$if|V| = 0thenreturn0if\exists v \in Vwithd(v) \leq 1thenreturn1 + mis2(G\backslash N[v])if\exists v \in Vwithd(v) = 2then(letu1andu2betheneighborsc \tag{17}$$

References

Gregory Matthew Ferrin. Independence polynomials. 2014.

Fedor V Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56 (3):80–88, 2013.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2013.

Johnnie Gray and Stefanos Kourtis. Hyper-optimized tensor network contraction. *Quantum*, 5:410, Mar 2021. ISSN 2521-327X. doi: 10.22331/q-2021-03-15-410. URL http://dx.doi.org/10.22331/q-2021-03-15-410.

Feng Pan and Pan Zhang. Simulating the sycamore quantum supremacy circuits, 2021.

Sebastian Schlag. *High-Quality Hypergraph Partitioning*. PhD thesis, Karlsruhe Institute of Technology, Germany, 2020.