Tensor models I

Subject: Graphs, tensor models, combinatorics, recursion, backtracking.

Introduction

Various interesting problems in combinatorics and graph theory arise when studying Majorana fermion quantum mechanical models having melonic large-N limits, [1]. In this problem we will use these models as an excuse to explore `Mathematica` graph-theory functionality, as well as to work on recursive backtracking and prunning techniques.

The field content of higher-rank Majorana fermion quantum mechanical models is just a single rank-r tensor of (real) fermionic degrees of freedom, $\psi_{i_1 i_2 \cdots i_r}$ with all indices taking values $1, 2, \ldots, N$. Assuming no symmetries among the different indices, each of them can be assigned a colour and only indices of the same colour should ever be contracted.

An interaction "bubble" is a graph where each vertex represents the field ψ, and all indices have been contracted. Lagrangians of specific models are constructed by adding one or more of these bubbles to the standard kinetic term. For example, for $r = 3$ we could have the bubbles represented in Fig. 1, as well as many others.

While the number of possible interaction bubbles grows very rapidly with the rank and number of vertices (can you think of other quartic interaction bubbles in rank-3 models?), only some of them lead to physically interesting models. Indeed, one may prove that a rank-r model has a melonic large-N limit when its interaction bubble is a "maximally single-trace" (MST) complete graph of $r + 1$ vertices[1]. The complete graph on $r + 1$ vertices is

[1] In graph-theory language, maximally single-trace complete interaction bubbles correspond to perfect 1-factorizations of the complete graph on $n = r + 1$ vertices, K_n.

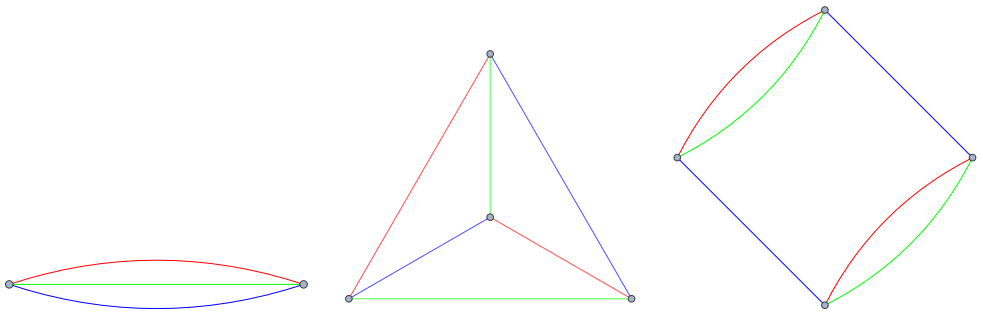

Figure 1: Examples of interaction bubbles for rank-3 models. (a) The quadratic interaction $\psi_{ijk}\psi_{ijk}$; (b) the quartic interaction $\psi_{i_1j_1k_1}\psi_{i_1j_2k_2}\psi_{i_2j_1k_2}\psi_{i_2j_2k_1}$; (c) the quartic interaction $\psi_{i_1j_1k_1}\psi_{i_1j_1k_2}\psi_{i_2j_2k_1}\psi_{i_2j_2k_2}$. The interaction bubble depicted in (b) is that of the Klebanov-Tarnopolski model [2].

obtained whenever any pair of fields (vertices) have exactly one index contraction (edge) between them. The MST property is achieved if, considering each pair of index positions (colors), their contractions are single-trace in matrix-model terminology (*i.e.* the graph is connected when we only look at the edges with the corresponding colors).

For example, Fig. 1a Fig. 1b represent MST interaction bubbles whereas Fig. 1c does not, and since Fig. 1b also corresponds to the complete graph on 4 vertices, this is the only MST complete interaction bubble depicted in Fig. 1. An example of a MST complete interaction bubble for rank-5 models is depicted in Fig. 2.

Of course, since the vertices of a bubble all represent the same field ψ, the nodes in the corresponding graph are unlabelled (*i.e.* indistinguishable). Moreover, the index positions (associated to colors) are not physically relevant, so that there is a residual color-permutation symmetry S_r. One may therefore consider interaction bubbles up to an arbitrary permutation of their colors[2].

Your task in this problem is to write `Mathematica` code to generate inequivalent MST complete interaction bubbles of various ranks.

[2]The residual S_r symmetry may be broken if each index is given its own range, $i_j = 1, 2, \ldots, N_j$ for $j = 1, 2, \ldots, r$, but we will not investigate this possibility any further.

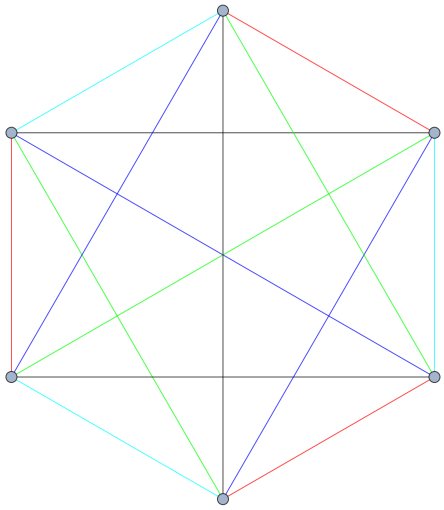

Figure 2: A MST complete interaction bubble for rank-5 tensors, corresponding to $\psi_{i_1 j_1 k_1 l_1 m_1}\psi_{i_1 j_2 k_2 l_2 m_2}\psi_{i_2 j_1 k_3 l_3 m_2}\psi_{i_2 j_3 k_1 l_2 m_3}\psi_{i_3 j_2 k_3 l_1 m_3}\psi_{i_3 j_3 k_2 l_3 m_1}$.

Problem statement

Before beginning, try to estimate the number of graphs you need to consider, *i.e.* the number of colorings of the complete graph K_n with $n-1$ colors which are candidates for MST interaction bubbles. Note that depending on how you pose the combinatorial problem, you may end up with very different estimations, all of which should however display the exponential blowup typical of graph combinatorial problems. This immediately leads us to two conclusions:

- There is no hope to find all MST complete interaction bubbles for n sufficiently large;
- The efficiency of your code will directly depend both on how you construct the candidate graphs, as well as on how you "filter" them to retain only the MST ones.

After constructing MST complete interaction bubbles, we are interested in keeping only the inequivalent ones w.r.t. the node-relabelling and color-shuffling isomorphisms described in the previous section. Graph isomorphism is a problem not known to be either in P or NP space, but in any case very

efficient implementations are available in practice. `Mathematica` provides this functionality through `IsomorphicGraphQ`, but unfortunately this function lacks support for multi-edges and edge-colored graphs, which makes it essentially useless for our purposes. You will therefore have to choose between implementing your own graph isomorphism function (easy but very inefficient) or interfacing with a more powerful library such as `nauty` (harder but providing state-of-the-art efficientcy), see http://pallini.di.uniroma1.it/ and [3].

Generating all MST complete interaction bubbles

Among the many construction approaches one could think of, here we will adopt the following. We add colours one-by-one (instead of, say, single edges), keeping after each step only the graphs for which the MST condition hasn't been brooken. Thus, for $i = 1, 2, \ldots, n-1$ in the i-th step we add colour i in all possible ways to each of the graphs we have generated with $i-1$ colours. Then for each resulting graph with i colors we check if all the colour pairs (i, j) with $j = 1, 2, \ldots, i-1$ satisfy the MST condition. You should implement this process recursively, the base case being that there is a unique graph with n nodes and 0 colors (*i.e.* the graph with no edges).

In this scheme the basic building blocks are sets of edges all having the same colour. In a valid set each node needs to be incident to exactly one edge, so that the set defines a "matching" of the n nodes, whereupon every node gets assigned a unique partner[3]. It should then be clear these sets consist of $n/2$ edges, thus making explicit the fact that MST complete interaction bubbles exist only for n even (*i.e.* odd-rank tensor models).

There are $(n-1)!!$ valid sets of edges (can you see why?), and in order to construct a MST complete interaction bubble we shall need to choose $n-1$ of them, one for each colour. However, not all pairs of sets are compatible with each other: since the complete graph has no repeated edges, we cannot use in a bubble two sets having a common edge. Denoting the sets by f_a for $a = 1, 2, \ldots, (n-1)!!$, we can then define a $(n-1)!! \times (n-1)!!$ compatibility matrix

$$C_{ab} \equiv (f_a \cap f_b = \emptyset) \ . \tag{1}$$

In order to speed-up the algorithm, both the f_a and the compatibility matrix C_{ab} should be pre-computed beforehand.

[3]In graph-theory language, these sets are 1-factors of the complete graph K_n.

The goal of this sub-problem is to implement the alogorithm described above to find non-isomorphic MST complete interaction bubbles with n vertices, for $n = 6$ (easy), $n = 8$ (intermediate) and $n = 10$ (harder). Do they exist for every n? Are they unique?

Generating random MST complete interaction bubbles

Because trying to construct all MST complete interaction bubbles for larger values of n is hopeless, one may resort to trying to construct "some" examples, if not all possible ones. For this purpose, the algorithm described in the previous subsection can immediately be adapted to become a randomized sampling technique. Indeed, at each step we can choose one of the possible (*i.e.* compatible) matchings to add to our current graph construction, instead of trying all possibilities. While doing this, if we ever run into a construction that cannot be extended, or one that breaks the MST condition, we simply restart from the beginning.

Another alternative consists in randomly prunning the search-tree whenever the number of graphs to consider is too high in some intermediate step. In this way, we make sure to check all possibilities both at the begining and at the end of our search, and introduce randomization only where strictly necessary.

Can you generate MST complete interaction bubbles for $n = 12$ in this way? Can you say anything about their uniqueness?

Details

There are various tricks and optimizations one can use to make the algorithms more efficient:

- **Bitmasks:** A bitmask encodes a finite set using an integer number, whose binary digits are 1 or 0 according to whether each element in a finite universe is present in the set or not, respectively. For example, if the universe of all possible elements is $\{a, b, c\}$, the $2^3 = 8$ possible sets in the universe are encoded by numbers $0, 1, \ldots, 7$, so that for example $\{a, c\}$ is encoded by $5 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$. More generally, the 2^K sets in a universe of K elements are encoded by the bitmasks $0, 1, \ldots, 2^K - 1$. The main advantage of this approach is that set oper-

ations such as intersection, union, *etc.* are implemented by elementary bitwise operations, *e.g.* `BitAnd`, `BitOr`, *etc.* in `Mathematica` .

- **Canonical encodings:** In order to remove duplicates from a list of L elements $e_1, \ldots, e_L$ given a boolean comparison function $F(e_i, e_j)$, one may have to execute the comparison $\mathcal{O}(L^2)$ times (in `Mathematica` this is performed by `DeleteDuplicates`). If the computation of F is very time-consuming (*e.g.* if we have implemented a very naive graph isomorphism function), this would be very inefficient. Instead, one may define a canonical representation for each element, say $G(e_i)$, and then delete duplicates by comparing canonical representations $F(e_i, e_j) \leftrightarrow G(e_i) = G(e_j)$. Even if the computation of G is as expensive as that of F, now we only need to run it $\mathcal{O}(L)$ times, so the whole process is much faster assuming the comparison of canonical representations is essentially trivial (in `Mathematica` this is performed by `DeleteDuplicatesBy`).

- **Removing redundancy I:** Note that the initial step of the construction is trivial, since any matching of n vertices is equivalent to every other matching upon relabelling the vertices. Thus, we can fix the matching used for the first color to be any one we want, say f_1 where the f_a have been ordered according to some arbitrary criterion (*e.g.* lexicographically). This simple optimization should greatly reduce the number of graphs generated in successive steps, therefore making the whole construction run much faster.

- **Removing redundancy II:** The results of the algorithms described in the previous section can be encoded as lists of $n-1$ matchings used to build a MST complete interaction bubble. Note that the order in which the matchings appear is associated to their colors, whose permutations are unphysical. Therefore, one may fix the order arbitrarily, say for example by always having $\{f_{a_1}, f_{a_2}, \ldots, f_{a_{n-1}}\}$ with $a_1 < a_2 < \cdots < a_{n-1}$. Enforcing this condition upon construction will again serve to reduce the number of intermediate graphs to consider, thus making the algorithm more efficient. Moreover, there is a neat way to do so: the compatibility matrix C_{ab} of (1) is in principle symmetric, since $f_a \cap f_b = f_b \cap f_a$, but we can break the symmetry by requiring that $b > a$ for f_b to be compatible with f_a. This will immediately result in

the desired S_{n-1} symmetry breaking[4].

Note that many of these ideas can (and should) be pushed further to allow the algorithm to work up to $n = 10, 12$.

Useful functions

You may find it useful to read the `Mathematica` help pages of the following functions:

- **Graphs:** `Graph`, `ConnectedGraphQ`, `AdjacencyMatrix`, ...
- **Others:** `FromDigits`, `IntegerDigits`, `BitAnd`, `BitOr`, `DeleteDuplicates`, `DeleteDuplicatesBy`, ...

References

[1] I. R. Klebanov, P. N. Pallegar and F. K. Popov, "Majorana Fermion Quantum Mechanics for Higher Rank Tensors," arXiv:1905.06264 [hep-th].

[2] I. R. Klebanov and G. Tarnopolsky, "Uncolored random tensors, melon diagrams, and the Sachdev-Ye-Kitaev models," Phys. Rev. D **95** (2017) no.4, 046004 doi:10.1103/PhysRevD.95.046004 [arXiv:1611.08915 [hep-th]].

[3] B.D. McKay and A. Piperno, "Practical Graph Isomorphism, II," Journal of Symbolic Computation, **60** (2014), pp. 94-112 doi:10.1016/j.jsc.2013.09.003 [arXiv:1301.1493 [cs-dm]].

[4] C. Colbourn, "CRC Handbook of Combinatorial Designs. Discrete Mathematics and Its Applications," CRC Press, 2010.

[4] While it may seem that at this point little symmetry is left, this is not quite the case. There is a residual symmetry of $\mathrm{Aut}(f_1)$ not broken by the choice of initial step, and it needs to be combined with S_{n-1} to fully account for bubble isomorphisms.