IJGTA **International Journal of Graph Theory and its Applications**
Volume 1 Issue 2 (2015) Pages 83–96

Sums of permanental minors using Grassmann algebra

P. Butera[1,*], M. Pernici[2]

[1]Dipartimento di Fisica Universita' di Milano-Bicocca and Istituto Nazionale di Fisica Nucleare, Sezione di Milano-Bicocca, 3 Piazza della Scienza, 20126 Milano, Italy
email: paolo.butera@mib.infn.it

[2]Istituto Nazionale di Fisica Nucleare, Sezione di Milano, 16 Via Celoria, 20133 Milano, Italy
email: mario.pernici@mi.infn.it

Received 19 June 2014; Revised 5 November 2014; Published online 28 December 2015
Corresponding editor: Yongtang Shi

Abstract We show that a formalism proposed by Creutz to evaluate Grassmann integrals provides an algorithm of complexity $O(2^n n^3)$ to compute the generating function for the sum of the permanental minors of a matrix of order n. This algorithm improves the Brualdi-Ryser formula, whose complexity is at least $O(2^{5n/2})$. In the case of a banded matrix with band-width w and rank n, the complexity is $O(2^{\min(2w,n)}(w+1)n^2)$.

Related algorithms for the matching and independence polynomials of graphs are presented.

Keywords Dimer problem
AMS subject classifications 05C69; 05C70; 15A15

1 Introduction

Let $G=(V,E)$ be an undirected graph with $v=|V|$ and $e=|E|$. A matching of G is a set of pairwise disjoint edges. The matching generating polynomial of G, i.e. the generating

*Corresponding author.

function of the number $N(i)$ of different matchings of G containing i edges, defined by

$$M(t) = \sum_{i=0}^{[v/2]} N(i) t^i \tag{1}$$

first appeared in combinatorics as the "rook" polynomial [17]. It was then introduced in statistical physics [9, 10] for the study of the monomer-dimer system on a lattice and in theoretical chemistry [11] to compute the "Hosoya index" $Z(G) = M(1)$, i.e. the total number of matchings of G.

$M(-1)$ on a lattice graph can be interpreted as the Witten index of a supersymmetric dimer model defined on that lattice [4].

For a generic graph, the best current algorithms for computing the matching generating polynomial are based on recurrence relations [7]. Alternatively, for a bipartite graph the coefficients of the matching polynomial can be computed as the sum of the permanental minors of the reduced adjacency matrix of the graph (defined as the submatrix of the adjacency matrix from the even to the odd vertices). The best current algorithm for computing the sum of the permanental minors is the Brualdi-Ryser formula [2]. Notice that even in the case of the permanent of banded matrices the complexity of currently used algorithm is in general exponential. In [24] it was shown that for banded matrices which are block factorizable the permanent can be computed in polynomial time.

The study of graph matchings can be naturally formalized by introducing anticommuting variables [5, 13, 20], so that the edges in a matching cannot overlap due to the Pauli exclusion principle. Creutz introduced an efficient algorithm for Grassmann integration [3]. In [6] the Creutz algorithm is applied to a graph coloring problem.

In this paper, we shall present a simplified form of the Creutz algorithm, in which Grassmann integration reduces to simple polynomial manipulations.

A hard object is represented as a product of even elements η_i of a Grassmann algebra, associated to the nodes i of the graph on which the objects lie. The η_i-elements are commuting and nilpotent and are represented as products $\eta_i = \bar{\theta}_i \theta_i$ of anticommuting variables $\theta_i, \bar{\theta}_i$. In the case of dimer systems, this notation was introduced [5, 8, 13] as a starting point to deduce the free-fermion interpretation of the close-packed dimer model on planar lattices.

The generating function that counts the hard objects is a Grassmann integral of a product on these objects.

Our algorithm to compute the generating function of the sums of the permanental minors of a matrix of order n has time complexity $O(2^n n^3)$, while for the Brualdi-Ryser formula the complexity is larger than $O(2^{5n/2})$. In the case of banded matrices with a fixed band-width w the former algorithm has quadratic complexity in n, $O(2^{2w}(w+1)n^2)$, the

latter exponential.

In the case of dimers, the η-elements are associated to the end-point of the dimer and we obtain efficient prescriptions to compute the matching polynomial for both bipartite and non-bipartite graphs.

Another important graph polynomial associated to G, the independence polynomial, $I(t) = \sum_{i\geq 0} a(i)t^i$, with $a(i)$ the number of independent subsets of i vertices in V, can be similarly derived. In this case the hard object is represented by η-elements associated to the edges adjacent to a vertex.

Appendix A tabulates the values of the Witten indices in the cases of square and hexagonal lattices with relatively large size, for some of which we disagree with the results of the calculation in [25].

We provide an implementation of these algorithms in Python; examples of its usage are given in Appendix B.

2 An algebraic formalism for counting hard objects on a graph

Define a "hard object" a on a graph G with vertex set V and edge set E as a subset V_a of the vertices in V. This is a generalization of the notion of dimer. Configurations of two or more hard objects onto G are admissible provided they have no common vertices (i.e. the vertex subsets of the various objects are "independent").

Let us associate to each object a the expression

$$O_a = 1 + w_a \prod_{i\in V_a} \eta_i \tag{2}$$

where w_a is a weight factor, and the product runs on a set of elements $\eta_i = \bar{\theta}_i\theta_i$, where θ_i, $\bar{\theta}_i$ are Grassmann anticommuting variables.

Define

$$\langle A\rangle = \int \prod_{i=1}^{v} \mathrm{d}\theta_i\, \mathrm{d}\bar{\theta}_i \exp\big(\sum \bar{\theta}_i\theta_i\big)A \tag{3}$$

where the Berezin integration [1] over anticommuting variables is used.

The η-elements satisfy the rules

$$\eta_i^2 = 0 \tag{4}$$

$$\eta_i\eta_j = \eta_j\eta_i \tag{5}$$

$$\langle \eta_{i_1}\cdots\eta_{i_k}\rangle = 1 \tag{6}$$

when $i_1,\dots,i_k$ are all distinct. Consider now the product of all admissible objects onto the graph G

$$Z_G = \left\langle \prod_a O_a \right\rangle. \tag{7}$$

One can write

$$Z_G = \left\langle \prod_a \left(1 + w_a \prod_{i\in V_a} \eta_i\right)\right\rangle = \int \mathrm{d}\theta\, \mathrm{d}\bar\theta\, \exp(S) \tag{8}$$

with

$$S = \sum_{i\in V} \bar\theta_i\theta_i + \sum_a w_a \prod_{i\in V_a} \bar\theta_i\theta_i. \tag{9}$$

If $w_a = t$ for all a, where t is a variable, $Z_G(t)$ is the generating function of the number of ways to settle the hard objects onto the graph.

Observe that after performing a partial product $\prod_b O_b$, if an element η_i does not occur in the remaining products in Z_G, then one can replace η_i with 1. This reduces the number of possible monomials, thus simplifying the product. To save memory and improve performance, the product should be ordered in such a way that only few elements η_i are present for any partial product. This algorithm is a simplified version of the Creutz algorithm [3]; since the Grassmann variables θ_i and $\bar\theta_i$ appear only in η_i, we can avoid introducing the Fock space for fermionic operators and use only simple polynomial manipulations.

2.1 Sums of permanental minors

The permanent of a $n\times n$ matrix A [15] is the coefficient of the $x_1\cdots x_n$ monomial in

$$\prod_{i=1}^n \sum_{j=1}^n A_{ij}x_j, \tag{10}$$

so obviously

$$\mathrm{perm}(A) = \left\langle \left(\sum_{i_1} A_{1,i_1}\eta_{i_1}\right)\left(\sum_{i_2} A_{2,i_2}\eta_{i_2}\right)\cdots \right\rangle. \tag{11}$$

The sum of permanental k-minors of a square matrix of size n is

$$p_k(A) = \sum \mathrm{perm}(A_{r,s}) \tag{12}$$

where r,s are all the $(n-k)$-subsets of $\{1,\dots,n\}$ and $A_{r,s}$ is the minor obtained by eliminating the rows r and the columns s. Using directly this formula [2], since the complexity for computing the permanent of A using the Ryser algorithm [19] is $O(2^n n)$, the complexity for computing the case $k = n/2$ for n even is $\binom{n}{n/2}^2 O(2^{n/2}n) \simeq O(2^{5n/2})$.

The generating function of the sums of permanental minors of the $m \times n$ matrix A is

$$\sum_k p_k(A)t^k = \left\langle \left(1+t\sum_{i_1=1}^{n} A_{1,i_1}\eta_{i_1}\right)\left(1+t\sum_{i_2=1}^{n} A_{2,i_2}\eta_{i_2}\right)\cdots\left(1+t\sum_{i_m=1}^{n} A_{m,i_m}\eta_{i_m}\right)\right\rangle. \quad (13)$$

To compute (13), after evaluating the i-th partial product, there are 2^n monomials in η, each of them multiplied by a polynomial of degree at most i in t. Multiplying by $(1+t\sum_{j=1}^{n} A_{i,j}\eta_j)$ and expanding the product one gets $i2^n n$ terms, so that the complexity of computing the generating function for the sum of permanental minors is $O(2^n m^2 n)$. In all the estimates of the time complexity, we have neglected the contribution of number multiplication.

In the case of a square banded matrix of size n with entries $M_{ij} = 0$ for $|i-j| > w$, at the end of the i-th partial product, the number of η-elements is $\nu = \min(2w, n)$, so that there are $2^{\min(2w,n)}$ monomials in η, each multiplied by a polynomial of degree i; the computational complexity is $O(2^{\min(2w,n)}(w+1)n^2)$, i.e. for fixed w, it is polynomial in n, while the algorithm in [2] is exponential.

If one is interested only in computing the permanent using the Ryser algorithm [19], the complexity is $O(n2^n)$, even in the case of banded matrices, while with our algorithm it is $O(n^2)$.

If the matrix is "almost banded", i.e. it has h non-zero elements outside the band, with h small, we have to replace $2w$ with $2w+h$ in the above estimate of the complexity.

2.2 Matching generating polynomial

The matching generating polynomial of G, i.e. the generating function of the number $N(i)$ of different matchings of G containing i edges, is defined by

$$M(t) = \sum_{i=0}^{[v/2]} N(i)t^i. \quad (14)$$

(Equivalently one defines the matching polynomial

$$\mu(t) = \sum_{i=0}^{[v/2]} (-1)^i N(i)t^{v-2i} \quad (15)$$

related to the former by $\mu(x) = x^v M(-x^{-2})$).

Consider now (7) in the case of dimers:

$$Z_G = \left\langle \prod_{\langle i,j\rangle} (1+w_{i,j}\eta_i\eta_j) \right\rangle. \quad (16)$$

The matching generating polynomial $M(t)$ is obtained from Z_G, setting $w_{i,j} = t$.

To make clear by an example the algebraic manipulations used, consider the graph A in Figure 1.

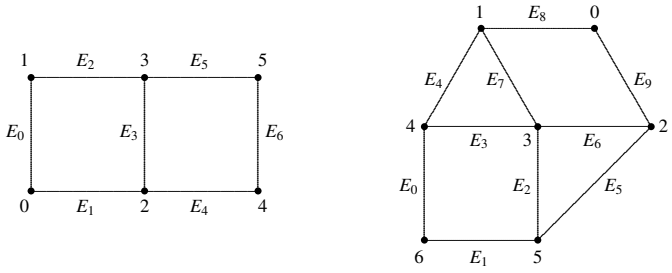

Figure 1. The graphs A and B.

Evaluate the partial product O_0O_1 (O_0 is associated to the edge E_0 etc.): one has

$$M_G = \langle (1 + t\eta_0\eta_1 + t\eta_0\eta_2) O_2O_3O_4O_5O_6 \rangle = \langle (1 + t\eta_1 + t\eta_2) O_2O_3O_4O_5O_6 \rangle.$$

In the last step η_0 has been replaced with 1 since it does not occur in the remaining terms $O_2, \ldots, O_6$. Similarly in the next step, after expanding the partial product $O_0O_1O_2$ we can set $\eta_1 = 1$,

$$M_G = \langle (1 + t + t\eta_2 + t\eta_3 + t^2\eta_2\eta_3) O_3O_4O_5O_6 \rangle.$$

In the partial product $O_0O_1O_2O_3$, a term $(t + t^2)\eta_2\eta_3$ is added. After expanding the partial product $O_0O_1O_2O_3O_4$, we can set $\eta_2 = 1$.

$$M_G = \langle (1 + 2t + (2t + 2t^2)\eta_3 + (t + t^2)\eta_4 + t^2\eta_3\eta_4) O_5O_6 \rangle.$$

In the partial product $O_0O_1O_2O_3O_4O_5$, we can set $\eta_3 = 1$:

$$M_G = \langle (1 + 4t + 2t^2 + (t + 2t^2)\eta_4 + (t + 2t^2)\eta_5 + (t^2 + t^3)\eta_4\eta_5) O_6 \rangle.$$

Finally

$$M_G = 1 + 7t + 11t^2 + 3t^3. \tag{17}$$

In each step there are at most two η-elements.

Given a graph G with v vertices and e edges, start with the empty graph G_0 on v vertices, add an edge to get G_1; then continue to add edges, until $G_e = G$. For a graph G_i in this sequence, an "active node" is by definition a node which is incident with at least one edge, and has a degree less than the degree that the node has in G. The active node number ν is

the maximum number of active nodes in the sequence $G_0, \ldots, G_e$. In general the size of the computer memory used by the algorithm grows with a factor 2^ν.

The graph A in the above example has active node number $\nu = 2$.

The time complexity for computing the matching polynomial for graphs with small active node number is $O(2^\nu v^3)$, analogous to the case of the sums of permanental minors for banded matrices; the space complexity is $O(2^\nu v^2)$. For fixed ν, the complexity is polynomial in v. Therefore one can deal with large graphs, provided ν is small.

We have not yet devised a general prescription to determine an ordering for which ν is close to minimum. A simple greedy procedure to get a sequence with small (but generally non-optimal) ν is the following: as long as it is possible, add an edge at the time without increasing the value of ν; otherwise add an edge of one among the shortest paths in $G - \{\text{non-active vertices}\}$, which join active vertices.

As an example of a sequence of random graphs with fixed ν, take a sequence of regular bipartite graphs constructed in the following way. Let G_0 be a cycle with k vertices, with k even. Add another cycle with k vertices; the odd (even) vertices of this cycle are linked respectively to the even (odd) vertices of the previous cycle in a random way, obtaining G_1; continue adding cycles in this way, obtaining the sequence G_i. The sequence of the corresponding regular bipartite graphs is obtained by linking the vertices of the first and last cycle. Since the active node number is $\nu = 2k$ (the vertices on the first and the last cycle are active), computing the matching polynomials for a sequence of N cycle graphs takes $O(N^2)$. The code for computing a sequence with $k = 6$ is included in the examples reported in [16].

In [3] an ordering of fermionic variables, with the insertion of a projector excluding a fermionic operator, is similarly chosen. Since this formalism is applied to fermions on a finite square lattice (a grid), there is a natural way to establish a longitudinal direction and a transverse direction. Only fermionic modes on the transverse direction appear in the computation, so one can deal with long grids with few modes in the transverse direction.

Observe that from (16), distributing a term $(1 + w_{k,l}\eta_k\eta_l)$ we get

$$Z_G(t) = \left\langle \prod_{\langle i,j\rangle \neq \langle k,l\rangle} (1 + w_{i,j}\eta_i\eta_j) \right\rangle + w_{k,l} \left\langle \prod_{\langle i,j\rangle : i\neq k,l;\, j\neq k,l} (1 + w_{i,j}\eta_i\eta_j) \right\rangle \tag{18}$$

which gives a recursion relation for matching generating polynomials [7]

$$Z_G(t) = M_{G-\langle k,l\rangle}(t) + tM_{G-k-l}(t). \tag{19}$$

If the graph G is bipartite, let us indicate by y_i the elements associated to the even sites, with η_i those associated to the odd sites; then

$$Z_G = \left\langle \left(1+\sum_{i_1} y_1 w_{1,i_1}\eta_{i_1}\right)\left(1+\sum_{i_2} y_2 w_{2,i_2}\eta_{i_2}\right)\cdots\right\rangle. \tag{20}$$

Since the y_j element occurs only in the j-th term of the product, it can be set equal to unity, so that

$$Z_G = \left\langle \left(1+\sum_{i_1} w_{1,i_1}\eta_{i_1}\right)\left(1+\sum_{i_2} w_{2,i_2}\eta_{i_2}\right)\cdots\right\rangle \tag{21}$$

which gives (13) in the case $w_{i,j} = tA_{i,j}$.

As an application, we have computed $M(-1)$ for some periodical square lattices and for some hexagonal lattices in the brick-wall representation considered in [25]; we disagree with [25] in some cases, see Appendix A.

2.3 Independence polynomial

The independence polynomial $I_G(t) = \sum_{i\geq 0} a(i)t^i$ is the generating function for the number $a(i)$ of ways of choosing i independent vertices on G. A hard object is made by associating to a vertex the product of the η-elements on the edges incident with that vertex. The greedy algorithm for ordering the product consists in choosing a short path in $G - \{\text{non-active vertices}\}$.

The matching generating polynomial of a graph G is the independence polynomial of the *line graph* of G.

As an example, consider the graph A, whose line graph is B (see Figure 1). Evaluate the partial product O_0O_1, set $\eta_8 = 1$,

$$I = I(L(A)) = I(B) = \langle O_0\cdots O_6\rangle = \langle (1+t\eta_9+t\eta_4\eta_7)O_2\cdots O_6\rangle. \tag{22}$$

Evaluate the partial product $O_0O_1O_2$, set $\eta_9 = 1$,

$$I = \langle (1+t+t\eta_4\eta_7+t\eta_5\eta_6+t^2\eta_4\eta_5\eta_6\eta_7)O_3\cdots O_6\rangle.$$

Evaluate the partial product $O_0O_1O_2O_3$, set $\eta_6 = \eta_7 = 1$,

$$I = \langle (1+t+t\eta_4+t\eta_5+(t+t^2)\eta_2\eta_3+t^2\eta_4\eta_5)O_4\cdots O_6\rangle.$$

Evaluate the partial product $O_0O_1O_2O_3O_4$, set $\eta_3 = \eta_4 = 1$,

$$I = \langle (1+2t+(t^2+t)\eta_0+(t+t^2)\eta_2+(t+t^2)\eta_5+t^2\eta_0\eta_5)O_5O_6\rangle.$$

Evaluate the partial product $O_0O_1O_2O_3O_4O_5$, set $\eta_2 = \eta_5 = 1$,

$$I = \langle (1+4t+2t^2+(t+2t^2)\eta_0+(t+2t^2)\eta_1+(t^2+t^3)\eta_0\eta_1)O_6\rangle.$$

Finally one gets the same as in (17).

As a check, we computed $I(-1)$ for the hexagonal lattices in the brick-wall representation considered in Table VII of [25], which can be interpreted as the Witten index of the quantum hexagonal model.

As another application, we computed $I(1)$ for square grids of size up to 35×35. The results agree with [21] where results are reported up to the size 33×33.

For the 34×34 square grid, we get

$$\begin{aligned} I_{34\times 34}(1) = {} & 38789112893323488901952524504879848981849788177663 45 \\ & 15543429025520634672163871702025048010830489308788 29 \\ & 13564262766592538500796108515840997971525487730656 07 \\ & 50525066858787608415249512675048159456458202982728 2. \end{aligned}$$

In the 35×35 case, we get

$$\begin{aligned} I_{35\times 35}(1) = {} & 72124294712717214286776360359845549941067616972563 90 \\ & 20463162637577536768438282480330361488529451859080 35 \\ & 25311051635720880907913188131152164648956900394960 48 \\ & 22376420723536367577998661984811651073683532087579 77 \\ & 68521522195. \end{aligned}$$

In theoretical chemistry $I(1)$ is called the Merrifield-Simmons index [14]. In Appendix B we have computed the Merrifield-Simmons index of the Buckminster fullerene.

3 Conclusions

We have shown that a simplified version of the Creutz algorithm can be used to compute sums of permanental minors, matching and independence polynomials. In the case of the sums of permanental minors, we have shown that this algorithm has lower complexity than using the Brualdi-Ryser formula. The algorithms are in general exponential, but they can become polynomial in particular cases. For example, sums of permanental minors have polynomial complexity if the matrix is banded. It is then important to be able to recognize whether a matrix can be brought to banded form by permuting its rows and its columns. A similar ordering problem is met when computing the matching and independence polynomials. We did not address the problem of finding an optimal ordering: presumably it is related to the tree decomposition of graphs [18].

Appendix A: The Witten index for rectangular and hexagonal periodic lattices

In [25] the Witten index $W = \sum(-1)^i N(i)$ is evaluated for the supersymmetric dimer model. For the largest lattices we agree with these results only modulo 2^{32}. We think it is likely that in [25] the large integer arithmetic was inadequately managed. We checked only the cases $m \times n$ for $m,n \geq 4$ and both even. The disagreeing values are listed in Tables 1 and 3.

We have computed the index $W(G)$ also for the larger lattices indicated in Table 2. The quantity $|W|^{(1/(mn))}$ should be compared with the expression $W = 2r'^{mn}\cos(mn\theta + \theta_0)$ of [25], where $r = 1.33 \pm 0.01$.

We have compared our evaluations of the Witten indices with the results in [25] also in the case of hexagonal lattices. Our results agree only modulo 2^{32} with [25] (Table VIII in that reference); the disagreeing values are shown in Table 3.

We have computed the index W also for the larger lattice indicated in Table 4. For $10 \leq m,n \leq 16$ the Witten index per site $|W|^{(1/(mn))}$ is between 1.156 and 1.192 with an average value 1.18, which lies below the interval $r = 1.4 \pm 0.1$ reported in [25].

m	n	W	$\lvert W\rvert^{(1/(mn))}$	W in [25]	$\lvert W\rvert^{(1/(mn))}$ in [25]
10	8	-14550253471	1.340	-1665351583	1.304
10	10	3235851927936	1.334	1741554048	1.237

Table 1. Comparison of the values of the Witten index $W(G)$ for a square grid G of size $m \times n$ obtained by the algorithm introduced in this paper with the disagreeing results in Table III in [25].

m	n	W	$\lvert W\rvert^{(1/(mn))}$
12	10	-139080563404700	1.312
12	12	988571682202805376	1.333

Table 2. The values of the Witten index W for a square grid of size $m \times n$ larger than those considered in [25].

m	n	W	W in [25]
10	14	7711439360	−878495232
10	16	−655517342208	1612654080
12	12	94909515776	420235264
12	14	6459966411264	335598080
12	16	100182729294336	−1677852160
14	10	11948085184	−936816704
14	12	6736033699456	1524979328
14	14	742553681809408	1080971264
14	16	−1384901745575424	1869085184

Table 3. The values of the Witten index W for hexagonal grids of size $m \times n$, disagreeing with those considered in [25].

m	n	W
16	14	−119633551609600
16	16	−6060918132969537536

Table 4. The values of the Witten index W for hexagonal grids of size $m \times n$ larger than those considered in [25].

Appendix B: Usage of the Python module "hobj"

The module "hobj" can be downloaded from [16]. It can be used without any dependence; the code for univariate polynomials, represented as arrays, is adapted from SymPy [23]. Here is the example for the graph A in Figure 1.

```
>>> from hobj import dup_matching_generating_poly
>>> d = {0:[1,2], 1:[0,3], 2:[0,3,4], 3:[1,2,5], 4:[2,5], 5:[3,4]}
>>> dup_matching_generating_poly(d)
[3, 11, 7, 1]
```

Since it is a bipartite graph, one can compute it also using the reduced adjacency matrix.

```
>>> from hobj import dup_permanental_minor_poly
>>> from domains import ZZ
>>> m = [[1,1,0],[1,1,1],[0,1,1]]
>>> dup_permanental_minor_poly(m, ZZ)
[3, 11, 7, 1]
```

In the case of bipartite graphs the second way is often faster.

The following examples take a fraction of a second on a current personal computer. Compute the sum of the permanental minors of a banded matrix.

```
>>> from hobj import dup_permanental_minor_poly
>>> from domains import ZZ
>>> m = [[i*j if abs(i-j) < 6 else 0 for i in range(20)] for j in range(20)]
>>> sum(dup_permanental_minor_poly(m, ZZ))
11936810897247956264161397956481650508142206788L
>>> dup_permanental_minor_poly(m, ZZ, 1)
11936810897247956264161397956481650508142206788L
```

The second way is faster and uses less memory because it avoids constructing the polynomial. Similarly in the following examples.

Let us call "hobj" from Sage [22] and compute the sum of the coefficients of the matching polynomial for the Buckminster fullerene C_{60} (truncated icosahedron) computed first in [12].

```
sage: from hobj import dup_matching_generating_poly
sage: d = graphs.BuckyBall().to_dictionary()
sage: sum(dup_matching_generating_poly(d))
1417036634543488
sage: dup_matching_generating_poly(d, val=1)
1417036634543488
```

Same for the independence polynomial.

```
sage: from hobj import dup_independence_poly
sage: sum(dup_independence_poly(d))
217727997152
sage: dup_independence_poly(d, val=1)
217727997152
```

References

[1] F. A. Berezin, The Method of Second Quantization, Academic Press, New York, 1966.

[2] R. A. Brualdi, H. J. Ryser, Combinatorial Matrix Theory, Cambridge University Press, Cambridge, 1991.

[3] M. Creutz, Evaluating Grassmann integrals, *Phys. Rev. Lett.* 81 (1998) 3555–3558.

[4] P. Fendley, K. Schoutens, H. van Eerten, Hard squares with negative activity, *J. Phys. A: Math. Gen.* 38 (2005) 315–322.

[5] M. E. Fisher, Statistical mechanics of dimers on a plane lattice, *Phys. Rev.* 124 (1961) 1664–1672.

[6] J. O. Fjærestad, Dimer and fermionic formulations of a class of colouring problems, *J. Phys. A: Math. Theor.* 45 (2012) 075001.

[7] C. D. Godsil, Algebraic Combinatorics, Chapman and Hall, London, 1993.

[8] R. Hayn, V. N. Plechko, Grassmann variable analysis for dimer problems in two dimensions, *J. Phys. A: Math. Gen.* 27 (1994) 4753–4760.

[9] O. J. Heilman, E. H. Lieb, Monomers and dimers, *Phys. Rev. Lett.* 24 (1970) 1412–1414.

[10] O. J. Heilman, E. H. Lieb, Theory of monomer-dimer systems, *Commun. Math. Phys.* 25 (1972) 190–232.

[11] H. Hosoya, Topological Index. A newly proposed quantity characterizing the topological nature of structural isomers of saturated hydrocarbons, *Bull. Chem. Soc. Jpn.* 44 (1971) 2332–2339.

[12] H. Hosoya, Matching and symmetry of graphs, *Comp. Math. Appl.* 12B (1986) 271–290.

[13] C. A. Hurst, H. S. Green, New solution of the Ising problem for a rectangular lattice, *J. Chem. Phys.* 33 (1960) 1059–1062.

[14] R. E. Merrifield, H. E. Simmons, Enumeration of structure-sensitive graphical subsets: Theory, *Proc. Natl. Acad. Sci. USA* 78 (1981) 692–695.

[15] J. K. Percus, Combinatorial Methods, Applied Mathematical Sciences 4, Springer-Verlag, New York, 1971.

[16] M. Pernici, "A Python library for computing matching and independence polynomials".
`https://github.com/pernici/hobj`

[17] J. Riordan, An Introduction to Combinatorial Analysis, Wiley, New York, 1958.

[18] N. Robertson, P. D. Seymour, Graph minors. III. Planar tree-width, *J. Combin. Theory Ser. B* 36 (1984) 49–64.

[19] H. J. Ryser, Combinatorial Mathematics, Carus Mathematical Monograph 14, Mathematical Association of America, Washington D.C., 1963.

[20] S. Samuel, The use of anticommuting variable integrals in statistical mechanics III: Unsolved models, *J. Math. Phys.* 21 (1980) 2820–2833.

[21] N. J. Sloane, "The On-Line Encyclopedia of Integer Sequences". https://oeis.org/search?q=A006506

[22] William A. Stein et al., Sage Mathematics Software (Version 5.7). http://www.sagemath.org

[23] SymPy Development Team, "SymPy: Python library for symbolic mathematics". http://www.sympy.org

[24] K. Temme, P. Wocjan, Efficient computation of the permanent of block factorizable matrices, preprint arXiv:1208.6589v1.

[25] H. van Eerten, Extensive ground state entropy in supersymmetric lattice models, *J. Math. Phys.* 46 (2005) 123302.