Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 95-106
www.stacs-conf.org

EXACT COVERS VIA DETERMINANTS

ANDREAS BJÖRKLUND

E-mail address: andreas.bjorklund@yahoo.se

Abstract. Given a k-uniform hypergraph on n vertices, partitioned in k equal parts such that every hyperedge includes one vertex from each part, the k-Dimensional Matching problem asks whether there is a disjoint collection of the hyperedges which covers all vertices. We show it can be solved by a randomized polynomial space algorithm in $O^*(2^{n(k-2)/k})$ time. The $O^*()$ notation hides factors polynomial in n and k.

The general Exact Cover by k-Sets problem asks the same when the partition constraint is dropped and arbitrary hyperedges of cardinality k are permitted. We show it can be solved by a randomized polynomial space algorithm in $O^*(c_k^n)$ time, where $c_3 = 1.496, c_4 = 1.642, c_5 = 1.721$, and provide a general bound for larger k.

Both results substantially improve on the previous best algorithms for these problems, especially for small k. They follow from the new observation that Lovász' perfect matching detection via determinants (Lovász, 1979) admits an embedding in the recently proposed inclusion–exclusion counting scheme for set covers, *despite* its inability to count the perfect matchings.

1. Introduction

The Exact Cover by k-Sets problem (XkC) and its constrained variant k-Dimensional Matching (kDM) are two well-known NP-hard problems. They ask, given a k-uniform hypergraph, if there is a subset of the hyperedges which cover the vertices without overlapping each other. In the kDM problem the vertices are further partitioned in k equal parts and the hyperedges each includes exactly one vertex from each part. While being two of the 21 items of Karp's classic list of NP-complete problems [6] for $k \geq 3$, little is known on their algorithmic side. In this paper, we present stronger worst case time bounds for these problems by combining Lovász' perfect matching detection algorithm via determinants [10] with the inclusion–exclusion counting for set covers [1]. We show

Theorem 1.1. *k-Dimensional Matching on n vertices can be solved by a Monte Carlo algorithm with exponentially low probability of failure in n, using space polynomial in n, running in $O^*(2^{n(k-2)/k})$ time.*

Theorem 1.2. *Exact Cover by k-Sets on n vertices can be solved by a Monte Carlo algorithm with exponentially low probability of failure in n, using space polynomial in n,*

1998 ACM Subject Classification: F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Hypergraphs.

Key words and phrases: Moderately Exponential Time Algorithms, Exact Set Cover, k-Dimensional Matching.

27th Symposium on Theoretical Aspects of Computer Science, Nancy, 2010
Editors: Jean-Yves Marion, Thomas Schwentick
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.STACS.2010.2447

Algorithm $\setminus k$	3	4	5	6	7	8
kDM in [1]	1.587	1.682	1.741	1.782	1.811	1.834
kDM here	1.260	1.414	1.516	1.587	1.641	1.682
XkC in [1]	1.842	1.888	1.913	1.929	1.940	1.948
XkC in [8]	1.769	1.827	1.862	1.885	1.901	1.914
XkC here	1.496	1.642	1.721	1.771	1.806	1.832

Table 1: Comparison of the base c in the $O^*(c^n)$ runtime of previous and the new algorithms.

running in $O^*(c_k^n)$ *time, with* $c_3 = 1.496, c_4 = 1.642, c_5 = 1.721, c_6 = 1.771, c_7 = 1.806$, *and in general* $c_k < 2\left(8.415k^{0.9-k}(k-1)^{0.6}(k-1.5)^{k-1.5}\right)^{-1/k}$

These bounds are large improvements over the previously known ones. In particular, for three dimensional matching our algorithm runs in time asymptotically proportional to the square root of the previous best algorithm's runtime.

We hope the present paper conveys the message that inclusion–exclusion is amendable not only to counting problems, but can at times be used more directly to settle the decision version of a problem.

1.1. Previous Work

Perhaps the most famous algorithmic contribution on the subject of exact covers is Knuth's *Dancing Links* paper [7], which actually just addresses a general implementation issue which saves a small constant factor in the natural backtracking algorithm for the problem. About the backtracking approach on exact cover he writes "Indeed, I can't think of any other reasonable way to do the job in general". While we certainly may agree depending on how much you put in the words "reasonable" and "general", we must point out that the best provable worst case bounds for the problems are obtained by analyzing very different algorithms. Björklund et al. [2] uses inclusion–exclusion and fast zeta transforms on the full subset lattice to show that exact set covers of any n vertex hypergraph can be counted in $O^*(2^n)$ time even when the number of hyperedges to choose from are exponential. Restricted to k-uniform hypergraphs, Koivisto [8] proposes a simple clever dynamic programming over subsets which show that Exact Cover by k-Sets can be solved in $O^*(2^{n(2k-2)/\sqrt{(2k-1)^2-2\ln(2)}})$ time. The algorithm is actually capable of counting the solutions and also works for not necessarily disjoint covers. It does, however, use exponential space. The best previous algorithm for the problem using only polynomial space is given in [1] and has a runtime bound in $O^*((1+k/(k-1))^{n(k-1)/k})$. For k-Dimensional Matching, the best known algorithm as far as we know is an $O^*(2^{n(k-1)/k})$ time algorithm resulting from a generalization of Ryser's inclusion–exclusion counting formula for the permanent [12], presented in [1]. A comparison of the bounds guaranteed by these algorithms and the ones given in this paper is shown in Table 1 for small k.

For $k = 2$ the problems X2C and 2DM are better known as the problems of finding a perfect matching in a general and bipartite graph, respectively. For these problems several polynomial time algorithms are known. We definitely admit that it seems like an obvious idea to try to reduce the $k > 2$ cases to the $k = 2$ case searching for faster algorithms for larger k. Still, we believe that it is far from clear how to achieve this efficiently. In this paper we make such an attempt by reducing the $k > 2$ cases to variants of one of the

first polynomial time algorithms for detecting the existence of perfect matchings: Lovász' algorithm from [10] which evaluates the determinant of the graph's Tutte matrix [13] at a random point.

2. Our Approach

2.1. Preliminaries

We use the terminology of (multi)hypergraphs. A hypergraph $H = (V, E)$ is a set V of n vertices and a *multiset* E of (hyper)edges which are subsets of V. Note in particular that with this definition edges may include only one (or even no) vertex and may appear more than once. In a k-uniform hypergraph each edge $e \in E$ has size $|e| = k$. Given a vertex subset $U \subseteq V$, the *projected hypergraph* of $H = (V, E)$ on U, denoted $H[U] = (U, E[U])$ is a hypergraph on U where there is one edge e_U in $E[U]$ for every $e \in E$, defined by $e_U = e \cap U$, i.e. the projection of e on U.

We study two related problems.

Definition 2.1 (k-Dimensional Matching, kDM)**.**
Input: A k-uniform hypergraph $H = (V_1 \cup V_2 \cup \cdots V_k, E)$, with $E \subseteq V_1 \times V_2 \times \cdots V_k$.
Question: Is there $S \subseteq E$ s.t. $\cup_{s \in S} s = V_1 \cup V_2 \cup \cdots V_k$ and $\forall s_1 \neq s_2 \in S : s_1 \cap s_2 = \emptyset$.

Definition 2.2 (Exact Cover by k-Sets, XkC)**.**
Input: A k-uniform hypergraph $H = (V, E)$.
Question: Is there $S \subseteq E$ s.t. $\cup_{s \in S} s = V$ and $\forall s_1 \neq s_2 \in S : s_1 \cap s_2 = \emptyset$.

For a matrix $\mathbf{A}$ we will by $\mathbf{A}_{i,j}$ denote the entry at row i and column j.

2.2. Determinants

The determinant of an $n \times n$-matrix $\mathbf{A}$ over an arbitrary ring R can be defined by the Leibniz formula

$$\det(\mathbf{A}) = \sum_{\sigma:[n]\to[n]} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} \mathbf{A}_{i,\sigma(i)} \tag{2.1}$$

where the summation is over all permutations of n elements, and sgn is a function called the sign of the permutation which assigns either one or minus one to a permutation. In this paper we will restrict ourselves to computing determinants over fields of characteristic two, GF(2^m) for some positive integer m. In such fields every element serves as its own additive inverse, and in particular so does the element one, and the sgn function identically maps one to every permutation. Thus it vanishes from Eq. 2.1 in this case, and the determinant coincides with another matrix quantity, called the *permanent*:

$$\operatorname{per}(\mathbf{A}) = \sum_{\sigma:[n]\to[n]} \prod_{i=1}^{n} \mathbf{A}_{i,\sigma(i)} \tag{2.2}$$

Permanents of 0–1-matrices over the natural numbers are known to count the perfect matchings of the bipartite graph described by the matrix. The reader may subsequently be tempted to think that this identity of determinants and permanents over fields of characteristic two is the property that makes our algorithms work. There is however nothing

magical about these fields in this context. Our reason for working in $\mathrm{GF}(2^m)$ is simply that with this choice of fields we don't even have to define the sign function, making several of the proof arguments later on much easier to digest. In principle though, *any* large enough field will work, with slightly more complicated proofs.

The interesting property of the determinant that we will exploit here is that although it is defined above in Eq. 2.1 as a sum of an exponential number of terms, it admits computation in time polynomial in n. This can be achieved for instance via the so called LU-factorization of the matrix which almost any textbook on linear algebra will tell you. In fact, computing the determinant is no harder than square matrix multiplication, see [3], and hence it can be done in $O(n^\omega)$ field operations where $\omega = 2.376$ is the Coppersmith–Winograd exponent [4].

2.3. Inclusion–Exclusion for Set Covers

Let us review the inclusion–exclusion counting scheme for exact set covers presented by Björklund and Husfeldt in [1]: Given a k-uniform hypergraph $H = (V, E)$ and any subset $U \subseteq V$, we can count the number of Exact Covers by k-Sets, denoted $\#\mathrm{X}k\mathrm{C}(H)$, by the inclusion–exclusion formula

$$\#\mathrm{X}k\mathrm{C}(H) = \sum_{X \subseteq V-U} (-1)^{|X|} W(H, U, X) \tag{2.3}$$

where $W(H, U, X)$ counts the number of ways to exactly cover U with $|V|/k$ edges in $H[U]$ whose corresponding edges in H are disjoint from X. Put differently, $W(H, U, X)$ counts the number of ways to pick $|V|/k$ edges from H, all having an empty intersection with X, which cover U without any overlap. In particular, when $U = \emptyset$ it is straightforward to compute $W(H, \emptyset, X)$ by just counting the number of edges in H disjoint from X, calling this quantity $d(X)$, and then computing the binomial $\binom{d(X)}{m}$. In [1], some examples where this algorithm could be accelerated by choosing a larger U were identified where the speedup was obtained by utilizing U's such that the projected hypergraph on U had low path–width. This enabled efficient counting by dynamic programming over a path decomposition.

2.4. Moving to GF(2^m)

In this paper, we find a new way to allow a large U to expedite the computation of the formula Eq. 2.3 above. We observe that whenever the projected hypergraph contains edges of size at most two, we can use determinants to compute the formula faster. We note that if the problem of counting perfect matching had an efficient algorithm A, we would almost immediately get an $O^*(2^{n(k-2)/k})$ time algorithm for the kDM problem. We would simply let U be any two of the parts in the input partition, and use A to compute $W(H, U, X)$. Unfortunately, counting perfect matchings even in bipartite graphs is #P-complete [14].

The key insight of the present paper circumvents the apparent obstacle formed by the intractability of counting matchings: we only need to be able to efficiently compute *some* fixed weighted sum of the matchings (with no weights set to zero). This is exactly where the determinants come to our rescue. The price we pay is that we have to give up counting the solutions over the natural numbers. Here we demonstrate the result through counting over fields of characteristic two which only allow us to detect if there is a cover at all and gives us little knowledge of their number. Furthermore, to avoid having an even number of solutions cancel we will employ a fingerprint technique, very much in the same spirit as Williams [15]

recently extended the k-path detection algorithm based on an algebraic sieving method of Koutis [9]. The fingerprint idea is to think of the computation as evaluating a polynomial of a degree much smaller than the number of elements of its base field and then computing it at a randomly chosen point. The fact that a polynomial cannot have more roots than its degree assure us that with great probability we discover with this single point probing whether the polynomial is the zero-polynomial or not. We will in fact use the multivariate polynomial analogue, see e.g. [11].

Lemma 2.3 (Schwartz-Zippel)**.** *Let $P(x_1, x_2, ..., x_n)$ be a non-zero n-variate polynomial of degree d over a field F. Pick $r_1, r_2, ..., r_n \in F$ uniformly at random, then*

$$Pr(P(r_1, r_2, ..., r_n) = 0) \leq \frac{d}{|F|}$$

For now, it is sufficient to think of the inclusion–exclusion formula of Eq. 2.3 as evaluating a multivariate polynomial over the base field $\mathrm{GF}(2^m)$ for some m. In what follows we will associate with all edges e in the input hypergraph a variable v_e. Our modified version of Eq. 2.3 reads as follows.

Lemma 2.4. *Given an XkC-instance $H = (V, E)$ and the family of all its solutions $\mathcal{S}$, we have that, for every subset $U \subseteq V$,*

$$\sum_{X \subseteq V-U} W_{2,f}(H, U, X) = \sum_{E' \in \mathcal{S}} \prod_{e \in E'} v_e^{f(e)} \tag{2.4}$$

where the computation is over a multivariate polynomial ring over GF(2^m), f is a function mapping the edges to the positive integers, and

$$W_{2,f}(H, U, X) = \sum_{E''} \prod_{e \in E''} v_e^{f(e)} \tag{2.5}$$

where the summation is over all $E'' \subseteq E$, satisfying four constraints

- *Avoidance,* $\forall e \in E'' : e \cap X = \emptyset$
- *Cardinality,* $|E''| = |V|/k$
- *Coverage,* $U \subseteq \cup_{e \in E''} e$
- *Disjointness,* $\forall e_1 \neq e_2 \in E'' : e_1 \cap e_2 \cap U = \emptyset$

Proof. First, note that every $E' \in \mathcal{S}$ fulfills all four conditions Avoidance, Cardinality, Coverage, and Disjointness for $X = \emptyset$, but violates Avoidance for every other X, irrespective of the choice of U. Thus, the contribution $\prod_{e \in E'} v_e$ of every solution E' is counted precisely once.

Second, a non-solution E'' obeying the three conditions Cardinality, Coverage, and Disjointness, fulfills the Avoidance condition for an *even* number of choices of X irrespective of U, namely for all subsets of the elements of V that the union of the sets in E'' fails to cover. Hence, all of these contributions $\prod_{e \in E''} v_e^{f(e)}$ cancel each other since we are working in a field of characteristic two. ∎

Combining the two Lemmas above 2.3 and 2.4 into an algorithm choosing a random point $r_1, r_2, ..., r_{|E|} \in \mathrm{GF}(2^m)$ and evaluating the left-hand sum of Eq. 2.4 in the straightforward fashion, we get:

Corollary 2.5. *Given an XkC-instance $H = (V, E)$ and a subset $U \subseteq V$, there is a Monte Carlo algorithm which returns "No" whenever there is no cover and returns "Yes" with*

probability at least $1 - \max_{e \in E} f(e)|V|/(k2^m)$ *when there exists at least one, running in time* $O^*(2^{|V|-|U|}\tau(W_{2,f}, U))$*, where* $\tau(W_{2,f}, U)$ *is the time required to evaluate any of the polynomials* $W_{2,f}(H, U, X)$ *for* $X \subseteq V - U$*, in a random point over the base field GF*(2^m)*.*

Note that by letting m be in the order of n, when f is bounded by a constant, we get exponentially low probability of failure in n. Armed with Corollary 2.5, we can start looking for projections U over which the computation of $W_{2,f}(H, U, X)$ is easy. The next two sections will describe two examples of how we can use determinants to accelerate the computation.

3. k-Dimensional Matching

We begin by the easier application, kDM. For this problem we can trivially find a large vertex subset on which the projected instance is a multigraph, and in fact also bipartite: we just use any two of the parts in the vertex partition given as input. Edmonds [5] observed that one could relate a bipartite graphs' perfect matchings to the determinant of a symbolic matrix. A perfect matching is a collection of disjoint edges so that every vertex is covered by precisely one edge. To a given a bipartite graph $G = (U, V, E), n = |U| = |V|$, he associated an $n \times n$-matrix $\mathbf{A}$ with rows representing vertices in U, and columns the vertices of V, and equated $\mathbf{A}_{i,j}$ with a variable v_{ij} if $(i, j) \in E$ and zero otherwise. He showed that the determinant of $\mathbf{A}$ is non-zero iff G has a perfect matching. We will use essentially the same result, with the small exception that we need to deal with multiple edges between a vertex pair, making sure all contributes. Formally

Definition 3.1. Given a hypergraph $H = (V, E)$ and a subset $U \subseteq V$ such that the projected hypergraph $H[U]$ is a bipartite multigraph on two equally sized vertex parts $U_1 \cup U_2 = U$, its Edmonds matrix, denoted $\mathbf{E}(H, U_1, U_2)$, is defined by

$$\mathbf{E}(H, U_1, U_2)_{i,j} = \sum_{\substack{e=(i,j) \in E[U] \\ i \in U_1, j \in U_2}} v_e$$

where again, v_e is a variable associated with the edge e.

We formulate our Lemma in terms of a special case of XkC instead of kDM directly to capture a more general case.

Lemma 3.2. *For a XkC-instance* $H = (V, E)$ *and two equally sized disjoint vertex subsets* $U_1, U_2 \subseteq V$ *such that the projected hypergraph* $H[U_1 \cup U_2]$ *is a bipartite multigraph,*

$$\det(\mathbf{E}(H, U_1, U_2)) = \sum_{M \in \mathcal{M}} \prod_{e \in M} v_e \tag{3.1}$$

where the computation is over a multivariate polynomial ring over GF(2^m) *for some* m *and the summation is over all perfect matchings* $\mathcal{M}$ *in* $H[U_1 \cup U_2]$*.*

Proof. By definition of the determinant 2.1, the summation is over all products of n of the matrix elements in which every row and column are used exactly once. Transferred to the associated bipartite graph, this corresponds to a perfect matching in the graph since rows and columns represent the two vertex sets respectively. Moreover, the converse is also true, i.e. for every perfect matching there is a permutation describing it. Hence the mapping is

one-to-one. The inner product counts all choices of edges producing a matching described by a permutation σ since:

$$\prod_{i=1}^{n} \mathbf{E}_{i,\sigma(i)} = \prod_{i=1}^{n} \sum_{e=(i,\sigma(i))} v_e = \sum_{M \in \mathcal{M}(\sigma)} \prod_{e \in M} v_e \tag{3.2}$$

where $\mathcal{M}(\sigma)$ is the set of all perfect matchings $e_1, e_2, ..., e_n$ such that $e_i = (i, \sigma(i))$. ∎

3.1. The Algorithm

Now we are ready to prove Theorem 1.1. Given an input instance $H = (V_1, V_2, ..., V_k, E)$ to the kDM problem where $V_1, V_2, ..., V_k$ describe the vertex partition of the n vertices, we simply let $U = V_1 \cup V_2$ in the algorithm described by Corollary 2.5, with f mapping one to every edge. To compute $W_{2,f}(H, U, X)$ we construct the Edmonds matrix of the hypergraph H *restricted* to its edges disjoint to X, projected on U, with the variables replaced by the random sample point $(r_1, r_2, ..., r_{|E|})$ chosen. Next we compute its determinant. The correctness follows from Lemma 3.2, after noting that every perfect matching in a projected hypergraph contains n/k disjoint edges. The runtime bound is easily seen to be $O^*(2^{n(k-2)/k})$ since $|U| = |V_1| + |V_2| = 2n/k$.

4. Exact Cover by k-Sets

Next we proceed to the XkC problem. In comparison to the kDM we are faced with a number of additional obstacles on our way to a similar result.

- First, a projection will typically capture edges differently, some will have large projections and some no at all.
- Second, in particular the projected edges will probably not form a multigraph.
- Third, even if they did it may not be a bipartite one.

For the first obstacle, we will prove that it is sufficient to find a projection on which at least one cover's edges all leave projected edges of size two *or less*. This is basically an extension of the idea for the XkC algorithm in proposition 10 in [1]. There, a vertex subset U is picked uniformly at random of a carefully chosen size, and in the projected hypergraph only the edges which leave a projection of size one or less are kept. Then the inclusion–exclusion formula Eq. 2.3 is used after noting that $W(H, U, X)$ is now easy to compute. The process is repeated a number of times dictated by the size of U. The best size to use is a trade-off of the resulting summation runtime and the probability that a cover is projected gracefully in the sense that all its edges are kept after the projection.

For the second obstacle, in addition to handling multiple edges we also need to count perfect matchings in which loops, i.e. edges connecting a vertex to itself, count as covering the vertex of its endpoints. Since this means that not all perfect matchings will involve the same number of edges, we have to take special care to make the determinants useful. We use polynomial interpolation to solve for the contributions of matchings of the same size separately to be able to fulfill the Cardinality constraint for $W_{2,f}$ in Corollary 2.5. To this end we introduce an auxiliary variable s parametrizing the matrices and use several determinant calculations.

For the third obstacle, we will use a variation of a result generalizing Edmonds' due to Tutte [13]. He showed that even for general not necessarily bipartite graphs one can make

a connection between its perfect matchings and the determinant of a symbolic matrix, although twice as large matrices in both directions are required. To a given a graph $G = (V, E), n = |V|$ he associates an $n \times n$-matrix $\mathbf{A}$ with rows and columns representing the vertices, and assigns $\mathbf{A}_{i,j} = v_{i,j}$ for $i < j$ and $\mathbf{A}_{i,j} = -v_{i,j}$ for $i > j$ with $v_{i,j}$ a variable for each edge $(i, j) \in E$. The remaining entries are set to zero. The determinant of $\mathbf{A}$ is non-zero iff G has a perfect matching.

We define matrices similar to Tutte's:

Definition 4.1. Given a hypergraph $H = (V, E)$ and a subset $U \subseteq V$ such that in the projected hypergraph $H[U]$ all edges have size at most two, its Tutte matrix of index s, denoted $\mathbf{T}^{(s)}(H, U)$, is defined by

$$\mathbf{T}^{(s)}(H,U)_{i,j} = \begin{cases} \sum v_e & : e \in E[U], e = (i,j), i \neq j \\ s \sum v_e & : e \in E[U], e = (i,j), i = j \end{cases}$$

Lemma 4.2. *For a XkC-instance $H = (V, E)$ and a vertex subset $U \subseteq V$ such that in the projected hypergraph $H[U]$, every edge has size at most two,*

$$\det(\mathbf{T}^{(s)}(H,U)) = \sum_{M \in \mathcal{M}} s^{\Lambda(M)} \prod_{e \in M} v_e^{p(e)} \tag{4.1}$$

where the computation is over a multivariate polynomial ring over GF(2^m) for some m, the summation is over all perfect matchings $\mathcal{M}$ in $H[U]$, $\Lambda(M)$ is the number of loops in the matching M, and $p(e) = 1$ if e is a loop and $p(e) = 2$ otherwise.

Proof. By definition of the determinant 2.1, the summation is over all products $\prod_{i=1}^{n} \mathbf{T}^{(s)}_{i,\sigma(i)}$ for a permutation σ. Call a permutation σ good if $\forall i : \sigma(\sigma(i)) = i$ holds, and bad otherwise. We will argue that only good permutations contribute to the sum. To see why, consider a bad σ. Then there exists a smallest i such that $\sigma(\sigma(i)) = j \neq i$. Look at the cyclic sequence $\{c_i\}$ where $c_0 = i$ and $c_{k+1} = \sigma(c_k)$ for $k > 0$. Let $L > 2$ be the smallest positive integer such that $c_L = i$ (Note that there must be one and that all c_i in between must by distinct since every element in 1 through n is mapped to exactly once). Next define a cycle reversal operation D mapping bad permutations on bad permutations by letting $D(\sigma)$ be identical to σ except in the points c_1 through c_L, where instead $D(\sigma)(c_i) = c_{i-1}$. Now first observe that the reversal operation is dual in the sense that $D(D(\sigma)) = \sigma$ and that $D(\sigma) \neq \sigma$ since $L > 2$, and hence every bad permutation can be uniquely paired with another bad permutation. Second note that the contribution of a bad permutation is identical to the contribution of its dual, since the Tutte matrices are symmetrical. Thus, since we are counting in a field of characteristic two, they cancel each other.

Next we continue to observe that the good permutations describe precisely the structure of all possible perfect matchings in a multigraph: i's such that $\sigma(i) \neq i$ describe ordinary two-vertex edges in the matching, and i's such that $\sigma(i) = i$ describe loops.

The inner product of Eq. 2.1 reads

$$\prod_{i=1}^{n} \mathbf{T}^{(s)}_{i,\sigma(i)} = \left(\prod_{i, i=\sigma(i)} s \sum_{e=(i,i)} v_e \right) \left(\prod_{i, i \neq \sigma(i)} \sum_{e=(i,\sigma(i))} v_e \right) = \sum_{M \in \mathcal{M}(\sigma)} s^{\Lambda(M)} \prod_{e \in M} v_e \tag{4.2}$$

where $\mathcal{M}(\sigma)$ is the set of all *directed* perfect matchings $e_1, e_2, ..., e_n$ described by the good permutation σ for which $e_i = (i, \sigma(i))$.

Now consider a directed perfect matching $e_1, e_2, ..., e_n$ such that for some j, $e_j \neq e_{\sigma(j)}$, and refer to it as being bad. We will see that all of these cancel in very much the same way

as the bad permutations did. Namely, again find the smallest j for which this is the case, and define a reversal operation R_σ mapping bad directed perfect matchings onto themselves by exchanging e_j and $e_{\sigma(j)}$. Since this operation pairs up the bad directed perfect matchings ($R_\sigma(\{e_i\}) \neq \{e_i\}$ and $R_\sigma(R_\sigma(\{e_i\})) = \{e_i\}$) and we work in a field of characteristic two, their contributions cancel. Thus we are left with only good permutations and good directed perfect matchings. The latter can be thought of as undirected perfect matchings in which every non-loop edge is included twice in the product. ∎

To find the contributions of matchings of the same size separately, think of the matchings partitioned in groups $\mathcal{M}_0, \mathcal{M}_1, ..., \mathcal{M}_n$ according to the number of loops of the matching. We can rewrite the determinant in Lemma 4.2 as

$$\det(\mathbf{T}^{(s)}(H,U)) = \sum_{i=0}^{n} s^i M_i \tag{4.3}$$

where $M_i = \sum_{M\in\mathcal{M}_i} \prod_{e\in M} v_e^{p(e)}$ are the quantities we seek. The right hand side of Eq. 4.3 is a degree n polynomial in s and thus we can solve for $M_0, M_1, ..., M_n$ by computing $\det(\mathbf{T}^{(s)}(H,U))$ in n different choices of s, and use Lagrange's interpolation formula to recover the sought values. In fact, either there are no matchings with an odd number of loops or no matchings with an even number of loops depending on the parity of $|U|$. Consequently, the evaluation of $n/2$ points suffices, but we disregard from this optimization possibility for simplicity. Once we have the M_i's we are close to be able to compute $W_{2,f}$ efficiently according to the following Lemma:

Lemma 4.3. *Given a XkC instance $H = (V, E)$ and a $U \subseteq V$ such that for all edges $e \in E$,$|e \cap U| \leq 2$, $f(e) = 2$ if $|e \cap U| = 2$ and 1 otherwise, and any $X \subseteq V - U$,*

$$W_{2,f}(H,U,X) = \sum_{i=0}^{|U|} Z(\frac{|V|}{k} - \lfloor\frac{|U|+i}{2}\rfloor)M_i \tag{4.4}$$

where $M_i = \sum_{M\in\mathcal{M}_i} \prod_{e\in M} v_e^{p(e)}$ are the contribution of all matchings $\mathcal{M}_i$ containing exactly i loops in the projected hypergraph of H on U restricted to the edges disjoint to X, and

$$Z(i) = \sum_{\substack{E'\subseteq Z \\ |E'|=i}} \prod_{e'\in E'} v_{e'} \tag{4.5}$$

with Z defining the set of edges e disjoint to X also having an empty intersection with U.

Proof. The M_i's count the contribution of all ways to cover U with the edges which leaves a non-empty projection on U and the $Z(i)$'s count the contribution of all ways to choose edges leaving an empty projection. Note that a matching from $\mathcal{M}_i$ involves exactly $\lfloor\frac{|U|+i}{2}\rfloor$ edges if it exists. The right hand side of Eq. 4.4 convolutes over all ways their total number of edges could equal $|V|/k$ in order to meet the Cardinality constraint in Corollary 2.5. ∎

The only piece missing is a simple way to evaluate $Z(i)$, and we note that it can be done by dynamic programming through a simple recursion. Number the edges in Z defined in Lemma 4.3 arbitrarily as $e_1, e_2, ..., e_p$, set $Z_i = \{e_1, e_2, ..., e_i\}$, and define

$$z(i,j) = \sum_{\substack{E'\subseteq Z_j \\ |E'|=i}} \prod_{e'\in E'} v_{e'} \tag{4.6}$$

These can be solved for by

$$z(i,j) = \begin{cases} 1 & : i = j = 0 \\ 0 & : i = 0 \text{ or } j = 0 \\ z(i-1,j-1)v_{e_j} + z(i,j-1) & : \text{otherwise} \end{cases} \tag{4.7}$$

and we finally compute $Z(i)$ through $Z(i) = z(i,p)$.

4.1. The Algorithm

We are ready to prove Theorem 1.2. First we describe the algorithm. Given an input instance $H = (V,E)$ to the XkC problem, we compute two parameters t and I depending on k. These are given by the calculations in the next section 4.2. We repeat the following procedure until we detect a cover, in which case we report so, or have tried unsuccessfully I times, in which case we report that no cover was found:

Algorithm 4.4.

(1) Choose a tn-sized subset $U \subseteq V$ uniformly at random.
(2) Construct $H_U = (V, E_U)$ where $E_U = \{e | e \in E, |e \cap U| \leq 2\}$.
(3) Run the summation algorithm in Corollary 2.5 on H_U, using U, and let $f(e) = 2$ if $|e \cap U| = 2$ and 1 otherwise. Use the method of the previous section 4 to compute $W_{2,f}(H_U, U, X)$, i.e.
   (a) Construct the Tutte matrices $\mathbf{T}^{(s)}$ of $H_U[U]$ restricted to its edges which are disjoint to X for $s = g^i, 0 \leq i \leq |U|$ where g is a generator of the multiplicative group in GF(2^m).
   (b) Compute the determinants of $\mathbf{T}^{(s)}$.
   (c) Use Lagrange interpolation to solve for the M_i's via Eq. 4.3.
   (d) Calculate the $Z(i)$'s by Eq. 4.7.
   (e) Evaluate Eq. 4.4.

Given that the random U is such that all edges in *some* exact cover S are kept in H_U, the previous Section 4 verifies its correctness: Lemmas 4.2 and 4.3 together with the observation that g^i for $1 \leq i \leq |U|$ are all distinct points, assures us that step (3) of the algorithm works. We are left with deciding t and I to make it very likely that some exact solution is kept at least once and tune them to get the best possible runtime.

4.2. Runtime Analysis

Our runtime analysis hinges on the probability that any fixed solution S to the XkC instance $H = (V,E)$ when projected on a subset $U \subseteq V$ chosen uniformly at random from the tn-sized subsets of V for some fraction t of the vertices, gets all its edges to leave a small projection on U, namely $\forall e \in S : |e \cap U| \leq 2$. We denote this event by $\varepsilon(t)$. If we repeat the process I times, the probability that none of the I independent random selections for U is successful in the sense that they retain S after the projection, is at most $(1 - \Pr(\varepsilon(t)))^I < e^{-\Pr(\varepsilon(t))I}$. Consequently, we need $I = \log(\epsilon^{-1})\Pr(\varepsilon(t))^{-1}$ to get probability at least $1 - \epsilon$ for one or more of the I selections to be successful. Thus we may use $\epsilon = c^{-n}$ for some constant $c > 1$ to get an exponentially low probability in n of failure without increasing the number of repetitions I by more than a polynomial factor.

k	τ_{12}	τ_2	t	$I^{1/n}$	c_k
3	0.961	0.679	0.547	1.092	1.496
4	0.936	0.613	0.387	1.073	1.642
5	0.921	0.583	0.301	1.060	1.721
6	0.912	0.565	0.246	1.050	1.771
7	0.905	0.554	0.208	1.043	1.806
8	0.900	0.546	0.181	1.038	1.832

Table 2: Numerically found parameters τ_{12} and τ_2 which approximately minimizes c_k.

To bound the probability of the event, we count the number of good tn-sized subsets of the vertices. This is a binomial sum (actually a trinomial one) over the number of edges in the solution S which gets a projection of size two:

$$\sum_{t_1+2t_2=tn} \binom{n/k}{t_1}\binom{n/k-t_1}{t_2}\binom{k}{2}^{t_2}\binom{k}{1}^{t_1} \tag{4.8}$$

To lower bound this sum of all non-negative terms, we will use just one of them. Let $N = n/k$ and parametrize $tn = \tau_{12}N + \tau_2 N$ where τ_{12} is the fraction of sets in the solution S which gets at least one of its elements chosen, and τ_2 is the fraction of sets that gets two.

Then, we bound our probability as the quotient of the single term lower bound on the number of good sets and the number of all sets $\binom{n}{tn}$ to

$$\Pr(\varepsilon(t)) \geq \frac{\binom{N}{\tau_{12}N}\binom{\tau_{12}N}{\tau_2 N}k^{\tau_{12}N}(k-1)^{\tau_2 N}}{2^{\tau_2 N}\binom{kN}{(\tau_{12}+\tau_2)N}} \tag{4.9}$$

The runtime of Corollary 2.5 is $O^*(2^{n-tn})$ given our polynomial time algorithm for computing $W_{2,f}$. Omitting polynomial factors, Algorithm 4.4 for XkC has to run for $Pr(\varepsilon(t))^{-1}$ different choices of U in the worst case. Let $T_{k,t}$ denote the final runtime, and expand the binomials of Eq. 4.9 to get:

$$T_{k,t} \leq \frac{2^{n-tn}}{\Pr(\varepsilon(t))} \leq \frac{2^{kN-\tau_{12}N}(\tau_2 N)!(N-\tau_{12}N)!(\tau_{12}N-\tau_2 N)!(kN)!}{N!k^{\tau_{12}N}(k-1)^{\tau_2 N}(kN-\tau_{12}N-\tau_2 N)!(\tau_{12}N+\tau_2 N)!} \tag{4.10}$$

If we replace the factorials with Stirling's approximation $n! \in \theta(\sqrt{n}(n/e)^n)$ and divide $(N/e)^{k+1}$ out of both numerator and denominator, we are left with a slightly less intimidating expression

$$T_{k,t} \leq \left(\frac{2^{(k-\tau_{12})}\tau_2^{\tau_2}(\tau_{12}-\tau_2)^{\tau_{12}-\tau_2}(1-\tau_{12})^{1-\tau_{12}}}{k^{\tau_{12}-k}(k-1)^{\tau_2}(k-\tau_{12}-\tau_2)^{k-\tau_{12}-\tau_2}(\tau_{12}+\tau_2)^{\tau_{12}+\tau_2}}\right)^N \tag{4.11}$$

Rewriting this as $T_{k,t} \leq c_k^n$ we see that c_k can be obtained as the k:th root of the expression within the brackets in Eq. 4.11. Solving numerically for the choices of τ_{12} and τ_2 that minimizes c_k we find that the minimum moves slightly with increasing k, see Table 2. The minimum, however, lies in a quite flat neighborhood within a large vicinity of the actual minimum, and comparable bounds not too far from the best possible with our technique are obtained with fixed parameters for all k by, say, $\tau_{12} = 0.9$ and $\tau_2 = 0.6$. With this choice of parameters in Eq. 4.11 we obtain the general bound in Theorem 1.2.

Acknowledgements

This research was supported in part by the Swedish Research Council project "Exact Algorithms".

References

[1] A. Björklund and T. Husfeldt. Exact Algorithms for Exact Satisfiability and Number of Perfect Matchings. Algorithmica 52(2): 226-249, 2008.
[2] A. Björklund, T. Husfeldt, and M. Koivisto. Set Partitioning via inclusion–exclusion. SIAM Journal on Computing Vol.39, No.2: 546-563, 2009.
[3] J. R. Bunch and J. E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication, Mathematics of Computation, 28: 231236, 1974.
[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation, 9:251-280, 1990.
[5] J. Edmonds. Systems of distinct representatives and linear algebra. Journal of Research of the Natural Bureau of Standards, 71B, 4:241–245, 1967.
[6] R. Karp. Reducibility Among Combinatorial Problems. Complexity of Computer Computations. New York: Plenum. pp. 85-103, 1972.
[7] D. E. Knuth. Dancing Links, arXiv: cs/0011047, 2000.
[8] M. Koivisto. Partitioning into Sets of Bounded Cardinality. Proceedings of the 7th IWPEC, 2009.
[9] I. Koutis. Faster Algebraic Algorithms for Path and Packing Problems. 35th ICALP, pp. 575–586, 2008.
[10] L. Lovász. On determinants, matchings and random algorithms. Fundamentals of Computing Theory. Akademia-Verlag, Berlin, 1979.
[11] R. Motwani and P. Raghavan. Randomized algorithms. Cambridge University Press, 1995.
[12] H. J. Ryser. Combinatorial Mathematics. Carus Math. Monographs, no. 14. Math. Assoc. of America, Washington, DC, 1963.
[13] W. T. Tutte. The factorization of linear graphs. Journal of the London Mathematical Society, 22:107–111, 1947.
[14] L. G. Valiant. The Complexity of Computing the Permanent. Theor. Comput. Sci. 8: 189–201, 1979.
[15] R. Williams. Finding Paths of Length k in $O^*(2^k)$ Time. Information Processing Letters 109(6):315–318, 2009.