Fixed-Parameter Tractability of the Weighted Edge Clique Partition Problem

Andreas Emil Feldmann
Department of Applied Mathematics, Charles University, Prague, Czech Republic

Davis Issac
Hasso Plattner Institute, Potsdam, Germany

Ashutosh Rai
Department of Applied Mathematics, Charles University, Prague, Czech Republic

Abstract

We develop an FPT algorithm and a compression for the Weighted Edge Clique Partition (WECP) problem, where a graph with n vertices and integer edge weights is given together with an integer k, and the aim is to find k cliques, such that every edge appears in exactly as many cliques as its weight. The problem has been previously only studied in the unweighted version called Edge Clique Partition (ECP), where the edges need to be partitioned into k cliques. It was shown that ECP admits a kernel with k^2 vertices [Mujuni and Rosamond, 2008], but this kernel does not extend to WECP. The previously fastest algorithm known for ECP has a runtime of $2^{\mathcal{O}(k^2)}n^{O(1)}$ [Issac, 2019]. For WECP we develop a compression (to a slightly more general problem) with 4^k vertices, and an algorithm with runtime $2^{\mathcal{O}(k^{3/2}w^{1/2}\log(k/w))}n^{O(1)}$, where w is the maximum edge weight. The latter in particular improves the runtime for ECP to $2^{\mathcal{O}(k^{3/2}\log k)}n^{O(1)}$.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Edge Clique Partition, fixed-parameter tractability, kernelization

Digital Object Identifier 10.4230/LIPIcs.IPEC.2020.17

Related Version A full version of the paper is available at https://arxiv.org/abs/2002.07761.

Funding All the three authors were supported by Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004).

1 Introduction

Problems that aim to cover a graph by a small number of cliques have a long history and have been studied extensively in the past (see e.g. [2, 3, 5, 10, 16, 18, 7, 8]). For these types of problems we are given a graph G and an integer k, and the tasks include to either cover or partition the edges or the vertices of G using at most k cliques or bicliques (i.e., complete bipartite graphs). Plenty of applications exist in both theory [22] and practice, e.g., in computational biology [1, 6], compiler optimization [21], language theory [11], and database tiling [9]. In this paper, we study the variant called the Edge Clique Partition (ECP) problem, defined as follows.

ECP (Edge Clique Partition)
Input: a graph G on n vertices, a positive integer k
Output: a partition of the edges of G into k cliques (if it exists, otherwise output NO)

ECP is known to be NP-hard even in K_4-free graphs and chordal graphs [16], and together with [14], the reductions of [16] imply APX-hardness. To circumvent these hardness results, we focus on *parameterized algorithms* (see [4] for the basics). More specifically, we focus on FPT algorithms for the natural parameter k, i.e., the number of cliques. Fleischer et al. [7] show that on planar graphs, ECP can be solved in $\mathcal{O}^*(2^{96\sqrt{k}})$ time[1]. They also

[1] The $\mathcal{O}^*$-notation hides polynomial factors in input size.

15th International Symposium on Parameterized and Exact Computation (IPEC 2020).
Editors: Yixin Cao and Marcin Pilipczuk; Article No. 17; pp. 17:1–17:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

generalized the result to d-degenerate graphs, giving an algorithm with $\mathcal{O}^*(2^{dk})$ runtime, which has a linear exponent for bounded-degeneracy graphs. For K_4-free graphs, Mujuni and Rosamond [18] gave an algorithm with a runtime[2] of $\mathcal{O}^*((\frac{k+3}{2})^k) = \mathcal{O}^*(2^{\mathcal{O}(k \log k)})$, which was improved by Fleischer et al. [7] to $\mathcal{O}^*((\sqrt{k}/3)^k)$ and even $\mathcal{O}^*((64c)^k)$ for some large (unspecified) constant c. Hence, also for these graphs an exponent linear in k is possible, albeit with a very large base. On the other hand, the algorithm of Mujuni and Rosamond [18] for K_4-free graphs has been empirically shown [24] to be rather efficient, even though it "only" comes with a near-linear exponent of $\mathcal{O}(k \log k)$.

Mujuni and Rosamond [18] showed that ECP is FPT in k for general graphs, by giving a *kernel* (see [4] for definition) of size k^2. However, no algorithms with (near-)linear dependence on k in the exponent are known for ECP. The fastest algorithm so far is given by Issac [12, Theorem 3.10] and runs in $\mathcal{O}^*(2^{2k^2+k\log_2 k+k})$ time, i.e., the exponent is quadratic in k. This algorithm is an adaptation of an algorithm by Chandran et al. [3] for the Biclique Partition problem (where we want to partition the edges into k bicliques) in bipartite graphs. In contrast, the best runtime lower bound known for ECP only excludes a sub-linear dependence on k in the exponent: if n denotes the number of vertices of the input graph, there is no $2^{o(k)}n^{O(1)}$ time algorithm for ECP assuming the Exponential Time Hypothesis (ETH). This follows due to a $2^{o(n)}$ lower bound for 3-Dimensional Matching [13] under ETH, and a reduction from Exact 3-Cover (which is a generalization of 3-Dimensional Matching) to ECP by Ma et al. [16]. An obvious open problem arising here is to close the gap between the upper and lower bounds on the runtime for ECP. Our main contribution is to show that for general graphs the exponent of the runtime for ECP can be significantly lowered from $\mathcal{O}(k^2)$ to $\mathcal{O}(k^{3/2}\log k)$.

▶ **Theorem 1.** *ECP has an algorithm running in* $\mathcal{O}\left((2e\sqrt{k})^{k^{3/2}+k} \cdot k^2 2^{5k} + n^2 \log n\right)$ *time.*

In fact, our algorithm solves a more general problem that we call the *Weighted Edge Clique Partition* (WECP) problem defined as follows:

WECP (Weighted Edge Clique Partition)
Input: a graph G on n vertices, edge weights $w_e : E(G) \to \mathbb{N}$, and a positive integer k
Output: a multiset of at most k cliques such that each edge appears in exactly as many cliques as its weight (if it exists, otherwise output NO)

Note that WECP is equivalent to ECP on a multigraph, by taking the weights as the edge multiplicities, which however increases the encoding length.

WECP can be thought of as a *clustering* of vertices where the clusters are allowed to overlap and the weight of an edge denotes the number of clusters in which the endpoints appear together. Such clustering problems appear naturally in computational biology, e.g., in the inference of gene pathways from gene co-expression data [20], where the clusters correspond to pathways and vertices correspond to genes. Thus developing efficient algorithms for WECP is of practical relevance.

WECP has not been studied previously and the known FPT algorithms for ECP do not extend to WECP. In particular, the techniques from the k^2-kernel for ECP by Mujuni and Rosamond [18] does not extend to WECP. Also, a 3^k-kernel for the very similar Biclique Partition problem by Fleischer et al. [8] just uses twin-reduction rule but this does not work for WECP. We first show a compression (see preliminaries for definition) with 4^k vertices for WECP that can be computed in polynomial time. The compression is into an even more general (auxiliary) problem that we call the *Annotated Weighted Edge Clique Partition* (AWECP) problem, defined as follows.

[2] In [18] the runtime was mistakenly reported as $\mathcal{O}^*(k^{(k+3)/2})$, cf. [7].

AWECP (Annotated Weighted Edge Clique Partition)
Input: a graph G on n vertices, edge-weights $w_e : E(G) \to \mathbb{N}$, a special set of vertices $W \subseteq V(G)$, vertex weights $w_v : W \to \mathbb{N}$, and a positive integer k
Output: a multiset of at most k cliques such that each edge e appears in exactly as many cliques as its edge-weight, and each vertex in W appears in exactly as many cliques as its vertex-weight (if such k cliques exist, otherwise output NO)

Note that WECP is exactly the special case of AWECP when W is empty. We give a kernel for AWECP that implies the compression for WECP into AWECP.

▶ **Theorem 2.** *AWECP has a kernelization algorithm that runs in $\mathcal{O}(n^2 \log n)$ time and outputs a kernel having at most 4^k vertices and encoding length $\mathcal{O}(16^k \log k)$ bits.*

▶ **Corollary 3.** *WECP has a compression into an AWECP instance having at most 4^k vertices and encoding length $\mathcal{O}(16^k \log k)$ bits. The compression can be found in $\mathcal{O}(n^2 \log n)$ time.*

Then we proceed to give the first FPT algorithm for WECP, which also implies the improved algorithm for ECP.

▶ **Theorem 4.** *WECP with the edge weights upper bounded by some value w has an algorithm running in $\mathcal{O}\left((2e\sqrt{k/w})^{k^{3/2}w^{1/2}+k} \cdot k^2 2^{5k} + n^2 \log n\right)$ time.*

Note that Theorem 4 implies an FPT algorithm for WECP when parameterized by k as $w \leq k$ for any YES-instance. Also, Theorem 1 follows from Theorem 4 by setting $w = 1$.

1.1 Our techniques

Our approach is based on the work of Chandran et al. [3], who solve the Bipartite Biclique Partition problem using linear algebraic techniques: we express AWECP as a low-rank matrix decomposition problem. For this we allow matrices to have wildcard entries in the diagonal that will be denoted by $\star$. We define $\mathbb{Z}^\star := (\mathbb{Z}_{\geq 0} \cup \{\star\})$. For $x, y \in \mathbb{Z}^\star$, we write $x \stackrel{\star}{=} y$ if and only if either $x = y$, or at least one of x and y is $\star$. For two matrices X and Y in $(\mathbb{Z}^\star)^{m\times n}$, we write $X \stackrel{\star}{=} Y$ if and only if $X_{i,j} \stackrel{\star}{=} Y_{i,j}$ for all i, j. We say that a *binary matrix* B (not containing wildcards) is a *Binary Symmetric Decomposition (BSD)* of a matrix $A \in (\mathbb{Z}^\star)^{n\times n}$ if $BB^T \stackrel{\star}{=} A$. The matrix B is called a width-k BSD of A if it is a BSD of A and has at most k columns. We define the *Binary Symmetric Decomposition with Diagonal Wildcards* (BSD-DW) problem as follows

BSD-DW (Binary Symmetric Decomposition with Diagonal Wildcards)
Input: an integer non-negative symmetric matrix $A \in (\mathbb{Z}^\star)^{n\times n}$ such that the wildcards $\star$ appear only in the diagonal, and an integer k
Output: a *width-k BSD* of A (if it exists, otherwise output NO)

We prove (in Lemma 6) that AWECP and BSD-DW are equivalent. Moreover, each column of B (solution to BSD-DW) corresponds to a clique (in the solution to AWECP), i.e. the rows that have a 1 in the j-th column correspond to the vertices that are in the j-th clique. Due to this, we will index the rows and columns of A with vertices, the rows of B with vertices and the columns of B with integers from $[k]$, that correspond to the k cliques. Moreover, we will be fluently switching between the contexts of edge partition of graphs (AWECP), and matrix decomposition (BSD-DW).

In Section 2 we prove that there is a kernel for AWECP with 4^k vertices. For this, we define the notion of $\star$-twins where two vertices u and v are said to be $\star$-twins, if the rows A_u and A_v are equal under $\stackrel{\star}{=}$. We group the vertices into equivalence classes (that we call *blocks*) of $\star$-twins. If a block has size more than 2^k, we show that they can be reduced and represented by one vertex. For this reduction rule, we need to specify how often the representative vertex needs to be covered by cliques. Thus, even if the input is an instance of WECP, the kernel we compute will be annotated, i.e., it will be an instance of AWECP. The 4^k bound on the kernel size follows then by giving a 2^k upper bound on the number of blocks for a YES instance. Since the edge weights and vertex weights for vertices in W cannot exceed k if there is a solution with at most k cliques, a kernel with at most 4^k vertices can be encoded using $\mathcal{O}(\binom{4^k}{2}\log k)$ bits, and so Theorem 2 follows.

To obtain Theorem 4, we first compute a kernel using Theorem 2 as the first step of the algorithm. Our algorithm will solve the more general AWECP problem. As in the algorithm of Chandran et al. [3] (where a different low-rank matrix decomposition problem is solved), the main idea of our algorithm is to guess a row basis for a width-k BSD B, and then fill the remaining rows of B one by one independent of each other. However we need to refine the techniques of Chandran et al. [3] in order to obtain our runtime improvement. In particular, there are two reasons why the algorithm in [3] has a quadratic dependence on k in the exponent: first, to guess a basis of rank k, they need to guess k binary vectors of length k each, which takes $\mathcal{O}(2^{k^2})$ time. But also, they need to guess the k row basis indices of B, for which there are $\binom{m}{k}$ possibilities if the matrix has m rows. Since for Bipartite Biclique Partition there is a kernel where $m \le 2^k$ [8], this adds another factor of $\mathcal{O}(2^{k^2})$ to the runtime.

To circumvent these two runtime bottlenecks, in Section 3 we devise an algorithm that gets around guessing the row indices of the basis of the solution matrix B. Instead of guessing the whole basis, we add a row to the basis only when the current basis cannot *take care* of that row. While this makes our algorithm more involved than the one by Chandran et al. [3], it means that the only bottleneck left is guessing the basis entries. For BSD-DW we can show that a basis with only $k^{3/2}w^{1/2} + k$ ones exists, which follows from the well-studied Zarankiewicz problem [19]. This bound on the structure of the basis then implies Theorem 4.

Since the only bottleneck, which prevents our algorithm from having near-linear dependence on k in the exponent of the runtime, is the step that guesses the entries of the basis for the solution matrix B, a natural question is whether our upper bound of $k^{3/2}w^{1/2} + k$ of the number of ones is (asymptotically) tight. In Section 4 we show that this is indeed tight (at least for the unweighted case) by proving the following theorem:

▶ **Theorem 5.** *For every prime power N and $k = N^2 + N$, there is a matrix $A \in \{0,1\}^{(k+1)\times(k+1)}$ such that there is a width-k BSD for A and every row basis of every width-k BSD of A has $\Theta(k^{3/2})$ ones.*

While this does not give a runtime lower bound in general, it implies that in order to speed up our algorithm for ECP using a better enumeration of the potential basis matrices, one needs to use some property other than a bound on the number of ones. The tight instances are obtained via the well-known Finite Projective Planes.

1.2 Related results

We now survey some results for ECP and related problems, apart from those mentioned above. For ECP, it is also known that the problem is solvable in polynomial time on cubic graphs [7]. The problem of partitioning the vertices instead of the edges into k cliques is equivalent to k-coloring on the complement graph, which is well-known to be NP-hard even for $k = 3$. Similarly, when the vertices need to be partitioned into bicliques or covered by bicliques, Fleischer et al. [8] proved NP-hardness for any constant $k \ge 3$.

Covering the edges of a graph by cliques or bicliques turns out to be generally harder than *partitioning* the edges. For the Edge Clique Cover problem, a kernel with 2^k vertices was shown by Gramm et al. [10], which results in a double-exponential time FPT algorithm when solving the kernel by brute-force. Cygan et al. [5] showed that this is essentially best possible, as under ETH no $2^{2^{o(k)}} n^{O(1)}$ time algorithm exists for Edge Clique Cover and no kernel of size $2^{o(k)}$ exists unless $\mathsf{P} = \mathsf{NP}$. Similarly, for the Biclique Cover problem, where edges of a general graph need to be covered by bicliques, Fleischner et al. [8] gave a kernel with 3^k vertices, and for the Bipartite Biclique Cover problem they gave a kernel with 2^k vertices in each bipartition. These kernels naturally imply double-exponential time algorithms. Chandran et al. [3] proved that for Bipartite Biclique Cover, under ETH no $2^{2^{o(k)}} n^{O(1)}$ time algorithm exists, and unless $\mathsf{P} = \mathsf{NP}$ no kernel of size $2^{o(k)}$ exists.

Chalermsook et al. [2] showed that for the Biclique Cover problem, it is NP-hard to compute an $n^{1-\varepsilon}$-approximation for any $\varepsilon > 0$ [3]. Edge Clique Cover is hard to approximate within $n^{0.5-\varepsilon}$ due to a reduction by Kou et al. [15]. In contrast, a PTAS exists for Edge Clique Cover on planar graphs [1].

1.3 Preliminaries

A problem P_1 parameterized by k is said to admit a compression into problem P_2 if there is an algorithm that takes as input an instance I_1 of P_1, runs in time polynomial in the encoding length of I_1, and outputs an instance I_2 of P_2 that is equivalent to I_1 such that the encoding length of I_2 is at most $f(k)$ for some computable function $f : \mathbb{N} \to \mathbb{N}$. In particular, the size of I_2 depends only on the parameter k and not on the size of I_1.

For an $m \times n$ matrix A, we use $A_{i,j}$ to denote the entry of A at row i and column j. We use A_i to denote the row-vector given by the i-th row of A. For some $I \subseteq [m]$ and $J \subseteq [n]$, we use $A_{I,J}$ to denote the sub-matrix of A when restricted to rows with indices in I and columns with indices in J. Also, we use A_I to denote a sub-matrix of A when restricted to rows with indices in I. We call such a sub-matrix where only rows are restricted as row sub-matrix. A *row-basis* (or just *basis* for brevity) B of A is any row sub-matrix of A such that every row of A can be expressed as a linear combination of rows of B, and the rows of B are linearly independent with each other.

▶ **Lemma 6.** *Given an instance (G, w_e, W, w_v, k) of AWECP we can find an equivalent instance (A, k) of BSD-DW in $\mathcal{O}(|V(G)|^2)$ time. Similarly given an instance (A, k) of BSD-DW, we can find an equivalent instance of AWECP in $\mathcal{O}(n^2)$ time, where n is the number of rows (or columns) in A.*

Proof. Given an instance (G, w_e, W, w_v, k) of AWECP, we can construct an instance of (A, k) of BSD-DW as follows. Let $V(G) = \{1, \dots, n\}$; take the non-diagonal entries of A as the corresponding entries of the *weighted adjacency matrix* of G, i.e., if there is an edge between two vertices u and v, the entry $A_{u,v}$ is equal to $w_e(uv)$ and if u and v do not have an edge between them then $A_{u,v} = 0$; for every vertex $v \in W$, take $A_{v,v}$ as the vertex weight of v; for every vertex $v \in V(G) \setminus W$, take $A_{v,v}$ as the wildcard $\star$. Note that the mapping is invertible, i.e., given a BSD-DW instance (A, k) we get an AWECP instance (G, w_e, W, w_v, k) as follows. Take $V(G) := \{1, 2, \dots, n\}$ where n is the number of rows (and columns) of A. For distinct $u, v \in [n]$, if $A_{u,v}$ is non-zero, put an edge between u and v in G with weight $A_{u,v}$. For each

[3] The paper wrongly claims the same result also for Biclique Partition. The bug is acknowledged here: `https://sites.google.com/site/parinyachalermsook/research?authuser=0`.

$v \in [n]$ such that $A_{v,v}$ is not a wildcard, put v in W and set its vertex weight to $A_{v,v}$. It is clear that this mapping is a bijective mapping between AWECP and BSD-DW instances and can be calculated in both directions in $\mathcal{O}(n^2)$ time. It remains to prove that the instances are equivalent.

Now, we define a bijective mapping between candidate solutions of the two problems. Naturally, a candidate solution of AWECP is a multiset of k cliques and a candidate solution of BSD-DW is an $n \times k$ matrix. Consider a candidate solution $\mathcal{C} := \{C_1, C_2, \dots C_k\}$ of an AWECP instance (G, w_e, W, w_v, k). We map it to a candidate solution $B \in \{0,1\}^{n\times k}$ of a BSD-DW instance (A, k) as follows. Take the row B_u as the characteristic vector of u in the k cliques, i.e., $B_{u,j} := 1$ if $u \in C_j$, and $B_{u,j} := 0$ otherwise. The inverse mapping then turns out to be as follows. Given a candidate solution $B \in \{0,1\}^{n\times k}$ of instance (A, k) construct k cliques where the j-th clique is $C_j := \{u \mid B_{u,j} = 1\}$. To see that C_j is indeed a clique, consider any two vertices $u, v \in C_j$: since $B_{u,j} = B_{v,j} = 1$, we know that $A_{u,v} = B_u B_v^T \geq 1$, which implies that there is an edge between u and v in G.

First, we prove that if $\mathcal{C}$ is a solution of AWECP(G, w_e, W, w_v, k), then B is a solution of BSD-DW(A, k). It is clear that B has only k columns by construction. So, it only remains to prove that for all pairs $u, v \in [n]$, $B_u B_v^T \stackrel{\star}{=} A_{u,v}$. First consider the case when u and v are distinct. Let J denote the set of all j such that both u and v appear together in C_j. Since $\mathcal{C}$ is a solution of AWECP(G, w_e, W, w_v, k), we have that $|J| = A_{u,v}$. By construction of B, we have that J is exactly the set of indices j where $B_{u,j} = B_{v,j} = 1$. Thus $B_u B_v^T = |J| = A_{u,v}$. Now consider the case when $u = v$. If $A_{u,u}$ is a $\star$ then clearly $B_u B_u^T \stackrel{\star}{=} \star = A_{u,u}$. So, suppose $A_{u,u} \neq \star$. This means $u \in W$ implying that u appears in exactly $A_{u,u}$ many cliques in $\mathcal{C}$. Thus $B_u B_u^T = A_{u,u}$.

We now prove the reverse direction, i.e., we prove that if B is a solution of BSD-DW(A, k), then $\mathcal{C}$ is a solution of AWECP(G, w_e, W, w_v, k). By construction, $\mathcal{C}$ has at most k cliques. Thus, it is sufficient to prove the following two statements: (1) every pair $u, v \in V(G)$ appears together in exactly $A_{u,v}$ many cliques in $\mathcal{C}$ (2) each vertex $v \in W$ appears in $A_{v,v}$ many cliques in $\mathcal{C}$. First we prove (1). We know $B_u B_v^T = A_{u,v}$. Since B is binary, this means that there are exactly $A_{u,v}$ many indices j such that $B_{u,j}$ and $B_{v,j}$ are both 1. Let J be the set of those indices. Observe that the set of cliques where both u and v appear together are exactly $\{C_j : j \in J\}$. Thus, the edge uv is in $|J| = A_{u,v}$ many cliques. Now we prove (2). Consider a vertex $v \in W$. We know $B_v B_v^T = A_{v,v}$. Since B is binary, this means that there are exactly $A_{v,v}$ many ones in B_v. Thus, the vertex v is in $A_{v,v}$ many cliques. ◀

2 Kernel

We will now give a kernel for AWECP and BSD-DW, thereby proving Theorem 2. Let (G, w_e, W, w_v, k) be an instance of AWECP and (A, k) be the corresponding instance of BSD-DW obtained by the transformation as in the proof of Lemma 6. We may move seamlessly between the graph and matrix terminologies as both problems are equivalent. Whenever we say a solution in this section, we mean the solution to the BSD-DW instance i.e., a width-k BSD of A. We say two distinct vertices u and v in G are **$\star$-twins** if they are adjacent and satisfy $A_u \stackrel{\star}{=} A_v$. We now prove the following easy property of $\star$-twins.

▶ **Lemma 7.** *For distinct vertices u, v and w in G, suppose u and v are $\star$-twins and v and w are $\star$-twins. Then:*

1. *u and w are $\star$-twins, and*
2. *all the entries of the submatrix $A_{\{u,v,w\},\{u,v,w\}}$ are the same except for wildcards.*

Proof. First, let us prove the second statement. Let $A_{u,v} = \alpha$. Then we know $A_{u,w} = \alpha$ as v and w are $\star$-twins. Then $A_{v,w} = \alpha$ as u and v are $\star$-twins. Thus all the non-diagonal elements of $A_{\{u,v,w\}\{u,v,w\}}$ are equal to α. If $A_{u,u} \neq \star$ then $A_{u,u} = A_{v,u} = \alpha$ as u and v are $\star$-twins. Similarly, if $A_{v,v} \neq \star$ then $A_{v,v} = A_{v,u} = \alpha$ as u and v are $\star$-twins. And, if $A_{w,w} \neq \star$ then $A_{w,w} = A_{v,w} = \alpha$ as v and w are $\star$-twins.

Now, for the first statement to hold, we only need to show that $A_{u,z} = A_{w,z}$ for all $z \notin \{u, v, w\}$. Indeed, $A_{u,z} = A_{v,z} = A_{w,z}$ where the first equality is because u and v are $\star$-twins and the second is because v and w are $\star$-twins. ◀

Thus we have that the relation $\star$-*twins* is transitive. It is also symmetric, as easily seen from the definition. Note that $\star$-twins are required to be adjacent, and thus the relation is not reflexive. But to make it reflexive, we simply define a vertex to be a $\star$-twin of itself. Thus, we can group the vertices into equivalence classes of $\star$-twins. We call each equivalence class a **block**. Note that there can be blocks containing only a single vertex. The following lemma is a direct consequence of Lemma 7.

▶ **Lemma 8.** *For a block D, the entries of the sub-matrix $A_{D,D}$ are all same except for wildcards.*

▶ **Fact 9.** *For values a, b and c, if $a \overset{\star}{=} b$ and $b \overset{\star}{=} c$, and $b \neq \star$ then $a \overset{\star}{=} c$.*

▶ **Lemma 10.** *Suppose we have a YES instance of AWECP without isolated vertices. Then there can be at most 2^k blocks.*

Proof. Let B be a width-k BSD of A. Note that B exists as we have a YES instance. In order to prove the lemma, it is sufficient to show that if u and v are in different blocks, then B_u and B_v are distinct, because then there can only be 2^k distinct rows of B, as there are only k columns in B and B is binary. Assume for the sake of contradiction that $B_u = B_v$ and u and v are in different blocks, i.e., they are not $\star$-twins. Let $b := B_u B^T = B_v B^T$. We have $A_u \overset{\star}{=} B_u B^T = b$ and $A_v \overset{\star}{=} B_v B^T = b$. This implies $A_u \overset{\star}{=} A_v$ using Fact 9, as the vector b contains no wildcards. Then, for u and v to be not $\star$-twins, it should be the case that u and v are not adjacent, i.e, $A_{u,v} = 0$. But then, $B_u B_v^T = 0$. Since $B_u = B_v$ by assumption, we have that $B_u = B_v = \mathbf{0}$ and hence $A_u = A_v = \mathbf{0}$. This means that u and v are isolated vertices, which is a contradiction. ◀

The above lemma shows the soundness of our first reduction rule that is as follows.

▶ **Reduction rule 1.** *If the number of blocks is more than 2^k, output that the instance is a NO instance.*

Next, we prove the following lemma about $\star$-twins that helps us to come up with a reduction rule that bounds the size of each block.

▶ **Lemma 11.** *Let $D := \{v_1, v_2, \ldots, v_t\}$ be a block of $\star$-twins. For a YES instance, there exists a solution B such that the rows $B_{v_1}, B_{v_2}, \ldots, B_{v_t}$ are either all pairwise distinct, or all same.*

Proof. It is sufficient to prove the following statement: if there is a solution B such that $B_{v_1} = B_{v_2}$, then there is also a solution C such that $C_{v_1} = C_{v_2} = \cdots = C_{v_t}$. So, assume that $B_{v_1} = B_{v_2}$. Let C be the matrix defined as $C_v := B_v$ for all $v \notin D$, and $C_v := B_{v_1} = B_{v_2}$ for all $v \in D$. We will prove that C is also a solution. For this, it is sufficient to prove that $C_u C_v^T = A_{u,v}$ for all $u, v \in V$ such that $A_{u,v} \neq \star$. If both u and v are not in D, then $C_u C_v^T = B_u B_v^T = A_{u,v}$. So, without loss of generality assume that $u \in D$. We distinguish the following cases.

1. If $v \in V \setminus D$, then $C_u C_v^T = B_{v_1} B_v^T = A_{v_1,v} = A_{u,v}$, where the last equality follows as v_1 and u are $\star$-twins.
2. If $v \in D \setminus \{u\}$, then $C_u C_v^T = B_{v_1} B_{v_2}^T = A_{v_1,v_2} = A_{u,v}$, where the last equality follows from Lemma 8.
3. If $v = u$: if $A_{u,u} = \star$ then there is nothing to prove, so assume $A_{u,u} \neq \star$. Then $A_{u,u} = A_{v_1,v_2}$ by Lemma 8. Hence we get $C_u C_u^T = B_{v_1} B_{v_2}^T = A_{v_1,v_2} = A_{u,u}$. ◀

Since there are only 2^k possible distinct rows for a solution B, Lemma 11 has the following consequence.

▶ **Lemma 12.** *Let $D := \{v_1, v_2, \ldots, v_t\}$ be a block of $\star$-twins such that $t > 2^k$. For a YES instance, there exists a solution B such that the rows $B_{v_1}, B_{v_2}, \ldots B_{v_t}$ are all same.*

The above lemma suggests that for a block D of size more than 2^k, we only need to keep one representative vertex for all the vertices in D. This leads us to our second reduction rule.

▶ **Reduction rule 2.** *Suppose there is a block D with more than 2^k vertices. Pick any two arbitrary vertices $u, v \in D$. We reduce our instance to an instance A' of AWECP (simultaneously to an instance G' of BSD-DW) as follows: let $G' := G \setminus (D \setminus \{v\})$; for every pair $(v_1, v_2) \neq (v, v)$ in $V(G') \times V(G')$, let $A'_{v_1,v_2} := A_{v_1,v_2}$; let $A'_{v,v} := A_{u,v}$.*

Once we have a solution B' to the reduced instance A' then we construct a solution B to the original instance A as follows: for all $x \in D$, let $B_x := B'_v$; for all $x \in V(G) \setminus D$, let $B_x := B'_x$.

Now, we prove that the above reduction rule is safe.

▶ **Lemma 13.** *Let A', G', B', B be as defined in Reduction rule 2.*
1. *If B' is a width-k BSD of A', then B is a width-k BSD of A.*
2. *Conversely, if A has a width-k BSD then so does A'.*

Proof. 1. It is clear that B has only k columns. So, it only remains to prove that B is a BSD of A, for which it is sufficient to prove that $B_{v_1} B_{v_2}^T \stackrel{\star}{=} A_{v_1,v_2}$ for all $v_1, v_2 \in V(G)$. For $v_1, v_2 \in V(G) \setminus D$, we have

$$B_{v_1} B_{v_2}^T = B'_{v_1} B'^T_{v_2} \stackrel{\star}{=} A'_{v_1,v_2} = A_{v_1,v_2}.$$

For $v_1 \in V(G) \setminus D$ and $v_2 \in D$, we have

$$B_{v_1} B_{v_2}^T = B'_{v_1} B'^T_{v} = A'_{v_1,v} = A_{v_1,v} = A_{v_1 v_2},$$

where the last equality follows as v and v_2 are $\star$-twins.
For $v_1, v_2 \in D$, we have

$$B_{v_1} B_{v_2}^T = B'_v B'^T_v = A'_{v,v} = A_{u,v} = A_{v_1,v_2},$$

where the last equality follows from Lemma 8.

2. By Lemma 12 we know that there exists a width-k BSD of A such that $B_{v_1} = B_{v_2}$ for all $v_1, v_2 \in D$. In particular $B_u = B_v$. Let B' be defined as $B'_x := B_x$ for all $x \in V(G')$. We show that B' is a width-k BSD of A'. Since B' has only k columns, it only remains to prove that B' is a BSD of A', which we do as follows. For $(v_1, v_2) \in (V(G') \times V(G')) \setminus (v, v)$, we have

$$B'_{v_1} B'^T_{v_2} = B_{v_1} B_{v_2}^T \stackrel{\star}{=} A_{v_1,v_2} = A'_{v_1,v_2},$$

and

$$B'_v B'^T_v = B_v B_v^T = B_u B_v^T = A_{u,v} = A'_{v,v}.$$ ◀

After the above rules are exhaustively applied, each block has size at most 2^k and the number of blocks is at most 2^k. Thus we have the required kernel of size 4^k.

The time required for computing the kernel can be shown to be $\mathcal{O}(n^2 \log n)$. This is because the blocks of $\star$-twins can be found by sorting the rows in lexicographic order. Since each comparison takes $\mathcal{O}(n)$ time the sorting can be done in $\mathcal{O}(n^2 \log n)$ time. Also, we need to compute the blocks only once as the reduction rules does not change the blocks.

Since the edge weights and vertex weights for vertices in W cannot exceed k if there is a solution with at most k cliques, a kernel with at most 4^k vertices can be encoded using $\mathcal{O}(\binom{4^k}{2} \log k)$ bits, and so Theorem 2 follows.

3 Algorithm

Here we give an algorithm for the BSD-DW problem. The algorithm also solves AWECP due to the equivalence from Lemma 6. In particular, it solves WECP thereby proving Theorem 4.

We now give a description of the algorithm. Pseudocode is given in Algorithm 1. Our input is a symmetric matrix $A \in (\mathbb{Z}_{\geq 0} \cup \{\star\})^{n\times n}$ where wildcards $\star$ appear only on the diagonal. First we guess a matrix $P \in \{0,1\}^{k\times k}$ such that for some $r \leq k$, $P_{[r],[k]}$ is a row basis of solution B. We show that for this, it is sufficient to enumerate $k \times k$ binary matrices that satisfy a specific property defined as follows. Let w be the largest integer entry of A. We call a matrix **w-limited** if the dot-product of each *distinct* pair of its rows is at most w. The following fact shows that we only need to enumerate w-limited matrices in $\{0,1\}^{k\times k}$ to guess P.

▶ **Fact 14.** *If B is a BSD of matrix A and w is the largest integer entry of A, then any submatrix of B (including B) is w-limited.*

Proof. Since B only has non-negative entries, if B is w-limited, then so are all the submatrices. Suppose the property does not hold for B. Then there exist two rows B_u and B_v such that $B_u B_v^T > w$. But $B_u B_v^T \stackrel{\star}{=} A_{uv}$ and hence $A_{uv} > w$ (note that A_{uv} is not $\star$ as it is not a diagonal-entry). Thus we have a contradiction. ◀

Note that guessing P is done in Loop 1 of Algorithm 1. We will later give a bound on the number of w-limited matrices in $\{0,1\}^{k\times k}$ during the runtime analysis in Section 3.2, thereby bounding the number of iterations of Loop 1.

We maintain partially filled matrices during the algorithm, i.e., we allow matrices to have *null rows* (this is different from wildcards). Think of the null rows as the rows that have not been filled yet. If each row of a matrix is either a binary row or a null row, we call it a *binary matrix with possibly null rows.* We denote by $\mathbb{B}^{n\times k}$, the set of all $n \times k$ binary matrices with possibly null rows.

We maintain a matrix $\tilde{B} \in \mathbb{B}^{n\times k}$ as a potential basis for our solution B. In Line 8, we call `CompleteBasis` that checks whether the current $\tilde{B}$ can be extended to a full solution B. Note that `CompleteBasis` does not try all possibilities to fill the remaining rows. It fills a row with the first binary vector that is compatible with the rows so far, where compatibility is defined as follows. For a matrix $B \in \mathbb{B}^{n\times k}$, we say that a vector $v \in \{0,1\}^k$ is **i-compatible** for B if $v^T v \stackrel{\star}{=} A_{i,i}$ and for all $j \neq i$ such that B_j is not a null row, $v^T B_j^T = A_{i,j}$. If `CompleteBasis` is able to fill all the rows with i-compatible binary vectors, then we are done and we return the resulting matrix (in Line 9). If not, we claim that the row for which we are not able to fill can be added to the basis (in Claim 16). So we add one more row to the basis by copying the next row from P (in Line 7). Thus we increase the number of non-null rows in the basis $\tilde{B}$ by one and repeat. Since the basis can be at most of size k, we need to repeat this at most k times.

Algorithm 1 Algorithm for BSD-DW.

Input : An $n \times n$ symmetric integer diagonal-wildcard matrix A
Output : If A has a width-k BSD then output a width-k BSD B of A; otherwise report that A has no width-k BSD

1 $w \leftarrow$ largest integer weight in A
2 **foreach** *w-limited* $P \in \{0,1\}^{k \times k}$ **do** `// Loop 1`
3   Initialize $\tilde{B}$ to be an $n \times k$ matrix with all null rows
4     $b \leftarrow 1$
5     $i \leftarrow 1$
6   **while** $b \leq k$ *and* P_b *is i-compatible with* $\tilde{B}$ **do** `// Loop 2`
7         $\tilde{B}_i \leftarrow P_b$
8         $(B, i) \leftarrow$ `CompleteBasis`$(A, \tilde{B})$
9   **if** $i = n + 1$ **then** **output** B and **terminate** the algorithm
10         $b \leftarrow b + 1$
11 **output** that A has no width-k BSD and **terminate** the algorithm

Function `CompleteBasis`(A,$\tilde{B}$):
12     $B \leftarrow \tilde{B}$
13   **for** *each null row i in B in increasing order* **do** `// Loop 3`
14   **if** *there is a* $v \in \{0,1\}^k$ *such that v is i-compatible with B* **then**
15             $B_i \leftarrow v$
16   **else return** (B, i)
17   **return** $(B, n + 1)$

3.1 Correctness of the algorithm

The algorithm outputs either through Line 9 or through Line 11. In the former case, we prove the following claim.

▷ Claim 15. If output occurs through Line 9, then the matrix B that is output, is a width-k BSD of A.

Proof. If Line 9 is executed, then this means that the preceding `CompleteBasis` call on Line 8 returned $i = n + 1$. This implies that the return from `CompleteBasis` happened on Line 17. This in turn means that Loop 3 was exited after completing all iterations, implying that the matrix B did not have any null rows at the time of return. Thus $B \in \{0,1\}^{n \times k}$. The rows of B were each filled either in Line 7 (when it was $\tilde{B}$ before being passed to `CompleteBasis`) or in Line 15. In both places, we filled each row i with a vector that was i-compatible at the time of filling. From the definition of i-compatibility, it follows that $BB^T \stackrel{\star}{=} A$, and hence B is a width-k BSD of A. ◁

Consider a NO instance first. From Claim 15 it follows that the output does not occur through Line 9. Thus the output has to occur through Line 11 and hence we correctly output that A does not have a width-k BSD. So it only remains to prove the correctness when A is a YES instance, i.e., when A has a width-k BSD, which is the case we consider for the remainder of the proof. Let B^* be any fixed width-k BSD of A.

Observe that $\tilde{B}$ changes as follows during each iteration of Loop 1: it is initialized to all null rows and each time the algorithm encounters Line 7 a null row is replaced with a binary row vector. We say that a matrix B is **consistent** with B^* if $B_j = B^*_j$ for each j such that B_j is a non-null row.

▷ Claim 16. Consider a matrix $\tilde{B} \in \mathbb{B}^{n\times k}$ that is consistent with B^*. If `CompleteBasis`$(A, \tilde{B})$ returns $i \in [n]$ then B^*_i is linearly independent from the non-null rows of $\tilde{B}$.

Proof. For a matrix $M \in \mathbb{B}^{n\times k}$, we denote by $R(M)$ the set of indices of the non-null rows of M. Suppose for the sake of contradiction that `CompleteBasis`$(A, \tilde{B})$ returns $i \in [n]$ and B^*_i is linearly dependent on the non-null rows of $\tilde{B}$. Then, we have $B^*_i = \Sigma_{\ell\in R(\tilde{B})}\lambda_\ell \tilde{B}_\ell$ for some $\lambda_1, \lambda_2, \cdots, \lambda_\ell \in \mathbb{R}$. Since $\tilde{B}$ is consistent with B^*, we can write $B^*_i = \Sigma_{\ell\in R(\tilde{B})}\lambda_\ell B^*_\ell$.

As `CompleteBasis` returned i, we know that during that iteration of Loop 3 in which row i was considered, no vector $v \in \{0,1\}^k$ was i-compatible with B (here B is the matrix maintained by `CompleteBasis` that was initialized to $\tilde{B}$ on Line 12). In particular, $B^*_i \in \{0,1\}^k$ was not i-compatible with B. Therefore either there was some $j \in R(B)$ such that $B^*_i B^T_j \neq A_{i,j}$, or $B^*_i(B^*_i)^T \not\stackrel{*}{=} A_{i,i}$. The latter cannot be true as B^* is a width-k BSD of A. So there was a $j \in R(B)$ such that $B^*_i B^T_j \neq A_{i,j}$.

We branch into two cases: case 1 when $j \in R(\tilde{B})$ and case 2 when $j \in R(B) \setminus R(\tilde{B})$. In case 1, we have $B_j = \tilde{B}_j = B^*_j$ where the second equality is because $\tilde{B}$ and B^* are consistent. Thus $B^*_i B^T_j = B^*_i(B^*_j)^T = A_{i,j}$, giving a contradiction.

In case 2, B_j was added in Line 15 and hence B_j was j-compatible with B at this time, implying that $B_\ell B^T_j = A_{\ell,j}$ for all $\ell \in R(\tilde{B})$. Since $B_\ell = \tilde{B}_\ell = B^*_\ell$ for $\ell \in R(\tilde{B})$, we have that $B^*_\ell B^T_j = A_{\ell,j}$ for all $\ell \in R(\tilde{B})$. Then, we have a contradiction as follows:

$$\begin{aligned}
B^*_i B^T_j &= \Sigma_{\ell\in R(\tilde{B})}\lambda_\ell B^*_\ell B^T_j \\
&= \Sigma_{\ell\in R(\tilde{B})}\lambda_\ell A_{\ell,j} \\
&= \Sigma_{\ell\in R(\tilde{B})}\lambda_\ell B^*_\ell (B^*_j)^T \\
&= B^*_i (B^*_j)^T \\
&= A_{i,j}
\end{aligned}$$ ◁

For a matrix $X \in \{0,1\}^{k\times k}$, we say we are in iteration (X,t) of the algorithm if we are in the iteration of Loop 1 with $P = X$ and the iteration of Loop 2 with $b = t$. We use $\tilde{B}(X,t)$ to denote the value of $\tilde{B}$ after the execution of Line 7 during iteration (X,t).

▷ Claim 17. At any step of the algorithm, if $\tilde{B}$ is consistent with B^* then the non-null rows of $\tilde{B}$ are linearly independent.

Proof. Consider the first time this is violated during the algorithm. This has to be during the addition of a new non-null row at Line 7. Let (X,t) be the iteration in which this happens. Let p be the index of the row that was added. Observe that $\tilde{B}(X,t)$ has only one additional non-null row compared to $\tilde{B}(X,t-1)$. Also, this additional non-null row is equal to B^*_p as $\tilde{B}(X,t)$ is consistent with B^*. We know the rows of $\tilde{B}(X,t-1)$ are linearly independent as we assumed that the first violation of lemma happens in iteration (X,t). Also, during iteration $(X,t-1)$, i was returned with value p (as the insertion happens in Line 7 in iteration (X,t)). This implies that B^*_p is linearly independent from the non-null rows of $\tilde{B}(X,t-1)$ due to Claim 16. Hence the rows of $\tilde{B}(X,t)$ are linearly independent. ◁

▷ Claim 18. If the iteration (X,k) occurs during the algorithm for some $X \in \{0,1\}^{k\times k}$ such that $\tilde{B}(X,k)$ is consistent with B^* then the algorithm outputs through Line 9 in iteration (X,k).

Proof. Consider the i returned by CompleteBasis$(A, \tilde{B}(X,k))$. It is sufficient to prove that the condition $i = n+1$ in Line 9 is satisfied. Suppose otherwise. Then $i \in [n]$ and by Claim 16, B^*_i is linearly independent from the non-null rows of $\tilde{B}(X,k)$. But by Claim 17, we have that the non-null rows of $\tilde{B}(X,k)$ are linearly independent and hence span the whole space, thus giving a contradiction. ◁

▷ Claim 19. Assume that the output of the algorithm does not occur through Line 9. If for some $Y \in \{0,1\}^{k\times k}$ and $t \le k-1$, iteration (Y,t) occurs and $\tilde{B}(Y,t)$ is consistent with B^*, then there exists some $Z \in \{0,1\}^{k\times k}$ such that iteration $(Z, t+1)$ occurs and $\tilde{B}(Z,t+1)$ is consistent with B^*.

Proof. Since $\tilde{B}(Y,t)$ is consistent with B^*, we know that $Y_{[t]}$ is a sub-matrix of B^*. As the condition in Line 9 is false, we know that an $i \in [n]$ was returned in Line 8 in iteration (Y,t). It is clear from the algorithm that i is a null-row in $\tilde{B}(Y,t)$. Let $Z \in \{0,1\}^{k\times k}$ be such that $Z_{[t]} := Y_{[t]}$, $Z_{t+1} := B^*_i$, and $Z_q := \mathbf{0}$ for all $q \ge t+1$. Observe that $Z_{[t+1]}$ is a submatrix of B^* and hence is w-limited by Fact 14. Since adding zeroes does not destroy w-limitedness, we have that Z is a w-limited $n \times k$ matrix. Thus there is some iteration of Loop 1 with $P = Z$. In this iteration the algorithm behaves similarly to the iteration with $P = Y$ for the first t iterations of Loop 2 as the algorithm has seen only the first t rows of P up to then. Thus $\tilde{B}(Z,t) = \tilde{B}(Y,t)$ and i is returned by Line 8 in iteration (Z,t). Now in Line 7 of iteration $(Z,t+1)$, $\tilde{B}_i$ is assigned Z_{t+1}. Note that $Z_{t+1} = B^*_i$ is indeed i-compatible with $\tilde{B}(Z,t)$ (as $\tilde{B}(Z,t) = \tilde{B}(Y,t)$ and $\tilde{B}(Y,t)$ is consistent with B^*) and that $t+1 \le k$. Hence the loop condition of Loop 2 is true in iteration $(Z,t+1)$. Thus, we have $(\tilde{B}(Z,t+1))_i = Z_{t+1} = B^*_i$ and for all $j \ne i$, we have $(\tilde{B}(Z,t+1))_j = (\tilde{B}(Y,t))_j$. Since $\tilde{B}(Y,t)$ is consistent with B^*, it follows that $\tilde{B}(Z,t+1)$ is consistent with B^*. ◁

Let t be the largest number for which there exists a $P \in \{0,1\}^{k\times k}$ such that iteration (P,t) happens and $\tilde{B}(P,t)$ is consistent with B^*. Due to Claim 19, we know that $t = k$. Then the algorithm outputs through Line 9 according to Claim 18. Thus the algorithm outputs a correct solution B due to Claim 15.

3.2 Runtime analysis

First, let us bound the number of iterations of Loop 1. For this it is sufficient to bound the number of w-limited matrices in $\{0,1\}^{k\times k}$.

▶ **Lemma 20.** *The number of binary w-limited $k \times k$ matrices is at most* $(2e\sqrt{k/w})^{k^{3/2}w^{1/2}+k}$.

Proof. Note that no w-limited matrix can have a $2 \times (w+1)$-sub-matrix having all ones. The number of ones in such a matrix is a special case of the well-studied Zarankiewicz problem and is known [19] to be at most $k^{3/2}w^{1/2} + k$. Hence it follows that the number of binary w-limited $k \times k$ matrices is at most $2^{k^{3/2}w^{1/2}+k} \cdot \binom{k^2}{k^{3/2}w^{1/2}+k}$ by choosing the positions of the at most $k^{3/2}w^{1/2} + k$ potential ones in the matrix and then choosing which of them are actually ones. The bound follows easily by using that $\binom{n}{k} \le \left(\frac{ne}{k}\right)^k$. ◀

Next, let us analyze the runtime of the function CompleteBasis. Loop 3 has at most n iterations. In Line 14, we need to check at most 2^k vectors $v \in \{0,1\}^k$. The checking for i-compatibility of each vector takes $\mathcal{O}(nk)$ time. Hence CompleteBasis takes $\mathcal{O}(k2^kn^2)$ time.

Now, we are ready to calculate the total run time. Due to Lemma 20, Loop 1 has at most $(2e\sqrt{k/w})^{k^{3/2}w^{1/2}+k}$ iterations. Line 3 takes $\mathcal{O}(nk)$ time. Loop 2 has at most k iterations. Line 7 takes at most $\mathcal{O}(k)$ time. The call to CompleteBasis in Line 8 takes at most $\mathcal{O}(k2^k n^2)$ time as we already calculated. Any other step takes only constant time. Thus the total running time is bounded by $\mathcal{O}\left(\left((2e\sqrt{k/w})^{k^{3/2}w^{1/2}+k}\right)\left(nk + k(k + k2^k n^2)\right)\right) = \mathcal{O}\left((2e\sqrt{k/w})^{k^{3/2}w^{1/2}+k} \cdot k^2 2^k n^2\right)$. We may run our algorithm on the kernel provided by Theorem 2, which means we may set $n = 4^k$ in the above expression. Thus the total running time is $\mathcal{O}\left((2e\sqrt{k/w})^{k^{3/2}w^{1/2}+k} \cdot k^2 2^{5k} + n^2 \log n\right)$. This proves Theorem 4.

4 Lower bound for number of ones in the basis matrix

In this section we construct binary matrices for which there is a width-k BSD and every basis of every width-k BSD has $\Omega(k^{3/2})$ ones, thereby proving Theorem 5. We obtain such instances via Finite Projective Planes (FPPs), which are defined as a set system $\mathcal{S}$ over a universe U of elements such that:

1. for each $e, e' \in U$ there is exactly one $S \in \mathcal{S}$ containing both of them,
2. for each $S, S' \in \mathcal{S}$ there is exactly one $e \in U$ such that $e \in S \cap S'$, and
3. there is a set of 4 elements in U such that no three of them are in any $S \in \mathcal{S}$.

It is known [17] that for any FPP, both the number of elements and the number of sets are equal to $N^2 + N + 1$ for some $N \geq 2$. Here N is called the *order* of the FPP. It also follows that for an FPP of order N, each set has exactly $N + 1$ elements and each element is contained in exactly $N + 1$ sets. It is also known that FPPs of order N exist for every prime power N [17]. Given an FPP of order N, in the following we will denote the characteristic incidence matrix of elements and sets by $F \in \{0,1\}^{(N^2+N+1)\times(N^2+N+1)}$, where rows are elements and columns are sets.

We now give a reduction from FPPs to ECP. For this, consider a vertex set V with $N^2 + N + 1$ vertices. Let I be a subset of $N + 1$ vertices in V. Let G_N be the graph defined as the clique over V minus the clique over I, i.e., every pair of vertices in V is adjacent except when both are from I. In other words, if $X := V \setminus I$, then G_N is a split graph with X as the clique and I as the independent set, where all the adjacencies are present between X and I. In Lemmas 21 and 23, we show that G_N has a small ECP if and only if an FPP of order N exists.

▶ **Lemma 21.** *If a finite projective plane $\mathcal{S}$ of order N exists, then G_N has a clique partition $\mathcal{C}$ into $|\mathcal{C}| \leq N^2 + N$ cliques.*

Proof. Let $\mathcal{S}$ be an FPP of order N over a universe U, and fix one of its sets $S \in \mathcal{S}$. We identify this set with the independent set of G_N, i.e., $S = I$. After fixing the elements of S, all other elements in $U \setminus S$ are arbitrarily identified with the other vertices in X. We claim that the remaining sets in $\mathcal{S} \setminus \{S\}$ form a clique partition, i.e., if $C_{S'} = \{uv \in E(G_N) \mid u, v \in S'\}$ then the set $\mathcal{C} = \{C_{S'} \mid S' \in \mathcal{S} \setminus \{S\}\}$ partitions the edge set of G_N into cliques. From Property 1 of an FPP, for any edge uv (i.e., at least one of u and v is in X) there is exactly one set $S' \in \mathcal{S} \setminus \{S\}$ such that $u, v \in S'$. This means that the subgraphs in $\mathcal{C}$ partition the edge set. Furthermore, by Property 2 no $S' \in \mathcal{S} \setminus \{S\}$ intersects in more than one vertex with the independent set I. Thus every subgraph of $\mathcal{C}$ is a clique. Moreover, any FPP of order N has exactly $N^2 + N + 1$ sets, and so there are $N^2 + N$ cliques in $\mathcal{C}$. ◀

▶ **Lemma 22.** *If $\mathcal{C}$ is a set of cliques that partition the edges of G_N and $|\mathcal{C}| \leq N^2 + N$, then for each $C \in \mathcal{C}$, $|V(C)| = N + 1$.*

Proof. First let us prove that $|V(C)| \leq N+1$. Suppose for the sake of contradiction that $|V(C)| \geq N+2$. Note that C contains at most one vertex from I, as a clique and independent set can intersect on at most one vertex. Let $C' := V(C) \setminus I$ and $I' := I \setminus V(C)$. Clearly $|C'| \geq N+1$ and $|I'| \geq N$ (recall that $|I| = N+1$). Note that every edge in $C' \times I'$ has to be covered by a distinct clique in $\mathcal{C} \setminus \{C\}$: any two edges that have different endpoints in I cannot be in the same clique, since there is no edge between these endpoints, while any two edges with different endpoints in C cannot be in the same clique, since the only edge between these endpoints is already covered by C. But there are $|C'||I'| \geq N^2+N$ such edges whereas there are only N^2+N-1 cliques in $\mathcal{C} \setminus \{C\}$. Thus we have a contradiction.

Hence we established $|V(C)| \leq N+1$. Now suppose for the sake of contradiction $|V(C)| < N+1$. Using the fact that every clique of $\mathcal{C}$ has size at most $N+1$, the total number of edges covered by $\mathcal{C}$ is strictly less than $|\mathcal{C}|\binom{N+1}{2} \leq (N^2+N)\binom{N+1}{2} = N^2(N+1)^2/2$. However, since $|I| = N+1$ and consequently $|X| = N^2$, the total number of edges of G_N is $\binom{N^2}{2} + N^2 \cdot (N+1) = N^2(N+1)^2/2$. Thus, we have a contradiction. ◀

▶ **Lemma 23.** *Let $N \geq 2$. If $\mathcal{C}$ is a set of cliques that partition the edges of G_N such that $|\mathcal{C}| \leq N^2+N$, then $\mathcal{S} = \{V(C) \mid C \in \mathcal{C}\} \cup \{I\}$ is an FPP of order N over V. Moreover, the incidence matrix F of $\mathcal{S}$ with the column for I removed from it, is the BSD of the adjacency matrix of G_N that corresponds to $\mathcal{C}$.*

Proof. We will prove that $\mathcal{S} = \{V(C) \mid C \in \mathcal{C}\} \cup \{I\}$ satisfies the three properties in the definition of an FPP, which then has order N by Lemma 22 above. Property 1 follows easily from the definition of an edge clique partition: for each pair of adjacent vertices there is exactly one clique covering their edge, while any pair of non-adjacent vertices only appear in I.

Let us now prove Property 2. For any $S, S' \in \mathcal{S}$, it follows easily from the definition of an edge clique partition that $|S \cap S'| \leq 1$ (otherwise some edge is contained in two cliques). Also, for any $S \in \mathcal{S}$, it is true that $|S \cap I| \leq 1$ (otherwise some clique would contain a non-edge). Assume there are $S, S' \in \mathcal{S}$ with $S \cap S' = \emptyset$. By Lemma 22, we have $|S| = |S'| = N+1$, and so all the $(N+1)^2$ edges of $S \times S'$ have to be covered by distinct cliques (otherwise some clique would contain an edge already covered by one of the cliques induced by S or S'). But we do not have so many cliques as $|\mathcal{C}| \leq N^2+N$. Thus we have $|S \cap S'| = 1$ for any $S, S' \in \mathcal{S}$, and so Property 2 is satisfied.

Let us now prove Property 3. Consider any arbitrary clique $C \in \mathcal{C}$. Pick two vertices from $V(C) \setminus I$ and two vertices from $I \setminus V(C)$. Note that $|V(C) \setminus I| = |I \setminus V(C)| \geq N+1-1 = N \geq 2$, and hence two vertices can be picked from the sets. It is easy to see that out of these four vertices at most two are in any set in $\mathcal{S}$.

It is easy to see that the incidence matrix F of $\mathcal{S}$ minus the column for I is the BSD of the adjacency matrix of G_N that corresponds to the clique partition $\mathcal{C}$. ◀

By using Lemmas 21 and 23 and the fact that the element-set incidence matrix of an FPP has full rank [23], we prove Theorem 5, thereby giving the required lower bound on the number of ones in the basis matrix.

▶ **Fact 24.** *The element-set incidence matrix of any FPP has full rank [23].*

Proof of Theorem 5. Let N be a prime power and $k := N^2+N$. We will show that the adjacency matrix A of G_N has a width-k BSD and every basis of every width-k BSD of A has $\Theta(k^{3/2})$ ones. Note that A is a $(k+1) \times (k+1)$ binary matrix as stated in the theorem.

Since N is prime, there is an FPP of order N [17]. Then by Lemma 21, there is an edge clique partition of G_N with at most $k = N^2 + N$ cliques. Thus, the adjacency matrix A of G_N has a width-k BSD, by using the equivalence in Lemma 6.

Now, consider any width-k BSD B of A and $\tilde{B}$ be any basis of B. Then, by Lemma 6, there is an edge clique partition of G_N with at most k cliques. By Lemma 23, $\mathcal{S} = \{V(C) \mid C \in \mathcal{C}\} \cup \{I\}$ is an FPP of order N. Let F be the element-set incidence matrix of $\mathcal{S}$. By Lemma 23, B is equal to F minus the column in F corresponding to I. By Fact 24, F has full rank, i.e. it has rank $N^2 + N + 1 = k + 1$. This implies B has rank k, and hence has at least k columns. Since B is a width-k BSD, this means it has exactly k columns, and hence is a $(k+1) \times k$ matrix. Since B has rank k, we have that $\tilde{B}$ has k rows and k columns. Thus, $\tilde{B}$ is B minus some row of B. Since each column of B corresponds to a clique of $\mathcal{C}$ containing $N+1$ vertices by Lemma 22, we have that B has $k(N+1)$ ones. Hence the number of ones in $\tilde{B}$ is at least $k(N+1) - k = \Theta(k\sqrt{k})$. ◀

5 Conclusion and Open Problems

We showed that AWECP admits a kernel with 4^k vertices, and an algorithm with a runtime of $2^{O(k^{3/2} w^{1/2} \log(k/w))} n^{O(1)}$, which implies that ECP can be solved in $2^{O(k^{3/2} \log k)} n^{O(1)}$ time. We think the following are the most interesting related open questions.

- Close the gap further between the upper and lower bounds on the running time for ECP that are currently $2^{\mathcal{O}(k^{3/2} \log k)} n^{\mathcal{O}(1)}$ and $2^{\Omega(k)} n^{\mathcal{O}(1)}$ respectively.
- Does WECP admit a polynomial-sized kernel like ECP?
- Can we show a tightness of analysis of our algorithm for WECP as we showed for ECP in Section 4, i.e., can we construct positive integer matrices with largest weight w that has a width-k BSD and every basis of every width-k BSD have $\Omega(k^{3/2} w^{1/2})$ ones?
- The algorithm of Chandran et al. [3] for Bipartite Biclique Partition with runtime $2^{O(k^2)} n^{O(1)}$ is also based on guessing the basis of a binary decomposition $A = BC$, and is currently the fastest FPT algorithm for the problem. If we can show that in any solution at least one of B and C has a row basis (column basis in case of C) with at most $g(k)$ ones, then we get a running time $2^{O(g(k) \log k)} n^{O(1)}$ using a similar algorithm as we gave for ECP. What is the minimum value of $g(k)$ possible?

References

1 Mathieu Blanchette, Ethan Kim, and Adrian Vetta. Clique cover on sparse networks. In *Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 93–102, 2012.

2 Parinya Chalermsook, Sandy Heydrich, Eugenia Holm, and Andreas Karrenbauer. Nearly tight approximability results for minimum biclique cover and partition. In *European Symposium on Algorithms*, pages 235–246. Springer, 2014.

3 L Sunil Chandran, Davis Issac, and Andreas Karrenbauer. On the parameterized complexity of biclique cover and partition. In *11th International Symposium on Parameterized and Exact Computation*, pages 1–13. Schloss Dagstuhl, 2017.

4 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4 (8). Springer, 2015.

5 Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. Known algorithms for edge clique cover are probably optimal. *SIAM Journal on Computing*, 45(1):67–83, 2016.

6 Andres Figueroa, James Borneman, and Tao Jiang. Clustering binary fingerprint vectors with missing values for dna array data analysis. *Journal of Computational biology*, 11(5):887–901, 2004.

7 Rudolf Fleischer and Xiaotian Wu. Edge Clique Partition of K4-Free and Planar Graphs. In *International Conference on Computational Geometry, Graphs and Applications*, pages 84–95. Springer, 2010.

8 Herbert Fleischner, Egbert Mujuni, Daniël Paulusma, and Stefan Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21-23):2045–2053, 2009.

9 Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *International conference on discovery science*, pages 278–289. Springer, 2004.

10 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction, exact, and heuristic algorithms for clique cover. In *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 86–94. SIAM, 2006.

11 Hermann Gruber and Markus Holzer. Inapproximability of nondeterministic state and transition complexity assuming p$\neq$ np. In *International Conference on Developments in Language Theory*, pages 205–216. Springer, 2007.

12 Davis Issac. *On some covering, partition and connectivity problems in graphs*. PhD thesis, Saarland University, Saarbrücken, Germany, 2019. doi:10.22028/D291-29620.

13 Klaus Jansen, Felix Land, and Kati Land. Bounding the running time of algorithms for scheduling and packing problems. *SIAM Journal on Discrete Mathematics*, 30(1):343–366, 2016.

14 Viggo Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Information Processing Letters*, 37(1):27–35, 1991.

15 L. T. Kou, L. J. Stockmeyer, and C. K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Commun. ACM*, 21(2):135–139, 1978. doi:10.1145/359340.359346.

16 SH Ma, WD Wallis, and JL Wu. The complexity of the clique partition number problem. *Congr. Numer*, 67:59–66, 1988.

17 J Matoušek and J Nešetřil. *Invitation to Discrete Mathematics*. Oxford University Press, 2009.

18 Egbert Mujuni and Frances Rosamond. Parameterized complexity of the clique partition problem. In *Proceedings of the fourteenth symposium on Computing: the Australasian theory-Volume 77*, pages 75–78. Australian Computer Society, Inc., 2008.

19 Vladimir Nikiforov. A contribution to the zarankiewicz problem. *Linear algebra and its applications*, 432(6):1405–1411, 2010.

20 Blair Sullivan (University of Utah). Personal communication.

21 Subramanian Rajagopalan, Manish Vachharajani, and Sharad Malik. Handling irregular ilp within conventional vliw schedulers using artificial resource constraints. In *Proceedings of the 2000 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 157–164, 2000.

22 Fred S Roberts. Applications of edge coverings by cliques. *Discrete applied mathematics*, 10(1):93–109, 1985.

23 Howard Sachar. The f_p span of the incidence matrix of a finite projective plane. *Geometriae Dedicata*, 8(4):407–415, 1979.

24 Xiao-tian Wu, Yu-Hao Lin, and R Fleischer. Research of fixed parameter algorithm for clique partition problem. *Computer Engineering*, 37(11):92–93, 2011.