CORE Metadata, citation and similar papers at core.ac.uk

Provided by Warwick Research Archives Portal Repository

WARWICK
THE UNIVERSITY OF WARWICK

Manuscript version: Author's Accepted Manuscript
The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/129258

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:
Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

warwick.ac.uk/lib-publications

Parameterized Complexity and Approximability of Directed Odd Cycle Transversal

Daniel Lokshtanov[*] M. S. Ramanujan[†] Saket Saurabh[‡§] Meirav Zehavi[¶]

Abstract

A *directed odd cycle transversal* of a directed graph (digraph) D is a vertex set S that intersects every *odd directed cycle* of D. In the Directed Odd Cycle Transversal (DOCT) problem, the input consists of a digraph D and an integer k. The objective is to determine whether there exists a directed odd cycle transversal of D of size at most k. In this paper, we settle the parameterized complexity of DOCT when parameterized by the solution size k by showing that DOCT does not admit an algorithm with running time $f(k)n^{\mathcal{O}(1)}$ unless $\mathsf{FPT} = \mathrm{W}[1]$. On the positive side, we give a factor 2 fixed-parameter approximation (FPT approximation) algorithm for the problem. More precisely, our algorithm takes as input D and k, runs in time $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$, and either concludes that D does not have a directed odd cycle transversal of size at most k, or produces a solution of size at most $2k$. Finally, assuming $\mathsf{gap\text{-}ETH}$, we show that there exists an $\epsilon > 0$ such that DOCT does not admit a factor $(1+\epsilon)$ FPT-approximation algorithm.

1 Introduction

A *directed odd cycle transversal* of a digraph D is a set S of vertices of D such that deleting S from D results in a graph without any directed odd cycles. In the NP-complete Directed Odd Cycle Transversal (DOCT) problem, the input consists of a digraph D on n vertices and an integer k, and the task is to determine whether D has a directed odd cycle transversal of size at most k.

DOCT generalizes several well studied problems such as Odd Cycle Transversal (OCT) on undirected graphs [1, 14, 28, 45], Directed Feedback Vertex Set (DFVS) [8, 21, 23, 27], and Directed Subset Feedback Vertex Set [12, 21]. In OCT, the input consists of an undirected graph G and integer k, and the task is to determine whether there exists a subset S of vertices such that $G - S$ is bipartite. Notice that OCT reduces to DOCT by replacing every edge by two arcs, one in each direction. In DFVS, the input consists of a digraph D and integer k, and the task is to determine whether there exists a subset S of vertices such that $D - S$ is a directed acyclic graph. Notice that DFVS reduces to DOCT by adding for every arc uv of D a new vertex x as well as the arcs ux and xv.

The existence of fixed-parameter algorithms (FPT algorithms) for OCT and DFVS were considered to be major open problems in parameterized complexity, until FPT algorithms were found for OCT in 2003 by Reed et al. [45], and for DFVS in 2007 by Chen et al. [8]. The algorithms for these two problems have had significant influence on the development of the field, resulting in proliferation of techniques such as *iterative compression* (OCT) and *important separators* [15, 20] (DFVS). Today both algorithms and corresponding methods are considered fundamental textbook material [15].

Once both OCT and DFVS were shown to be FPT, DOCT immediately became the next natural target. The parameterized complexity of DOCT was explicitly stated as an open problem [17] for the first time in 2007, immediately after the announcement of an FPT algorithm for DFVS. Since then the problem has been re-stated several times [9, 11, 39, 40]. In this paper, we settle the parameterized complexity of DOCT, by showing that the problem is W[1]-hard. Our hardness proof also gives a near-tight running time lower bound for DOCT assuming the Exponential Time Hypothesis

[*]Department of Computer Science, University of California, Santa Barbara, USA. daniello@ucsb.edu

[†]University of Warwick, Warwick, United Kingdom. R.Maadapuzhi-Sridharan@warwick.ac.uk

[‡]The Institute of Mathematical Sciences, HBNI, Chennai, India, IRL 2000 ReLaX and University of Bergen, Bergen, Norway. saket@imsc.res.in

[§]University of Bergen, Norway

[¶]Ben-Gurion University, Beersheba, Israel. meiravze@bgu.ac.il

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416 and No. 715744). The third author also acknowledges the support of Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18. The second author acknowledges support from Centre for Discrete Mathematics and its Applications at University of Warwick and *X-Tract*, Austrian Science Fund (FWF, project P26696). The last author acknowledges support from ISF grant no.1176/18.

(ETH). In particular, we prove the following.

Theorem 1.1. DOCT *is* W[1]*-hard. Furthermore, assuming the ETH there is no algorithm for* DOCT *with running time* $f(k)n^{o(k/\log k)}$.

On the one hand, Theorem 1.1 shows that DOCT is intractable from the perspective of parameterized complexity. On the other hand, the problem is known not to admit a constant factor approximation algorithm running in polynomial time, assuming the Unique Games Conjecture [28]. Hence, the next natural question is whether one could get a constant factor approximation algorithm in FPT time. Our second result is an affirmative answer to this question.

Theorem 1.2. *There is a* $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$*-time FPT-approximation algorithm for* DOCT *with approximation ratio* 2.

In light of Theorem 1.2 the next natural question is whether the approximation factor can be made arbitrarily close to 1. Our final contribution is to provide evidence that there exists an $\epsilon > 0$ such that DOCT does not admit a $(1+\epsilon)$ FPT-approximation algorithm. In particular, the proof of Theorem 1.1 can be thought of as a parameterized reduction from the Binary Constraint Satisfaction (BCSP) problem, informally defined as follows. The input consists of two integers n and k specifying that there are k variables, $x_1, \ldots, x_k$, each variable x_i taking a value from $\{1, \ldots, n\}$, together with a list of *constraints*. Each constraint specifies two variables, x_i and x_j, together with a list L of all legal pairs of values that x_i and x_j may take simultaneously. An assignment of values to the variables satisfies the constraint if $(x_i, x_j) \in L$. The task is to find an assignment that satisfies all constraints. It is well known (see e.g. [38]) that BCSP parameterized by the number of variables k is W[1]-complete. We conjecture that not only is it W[1]-hard to find a satisfying assignment to a BCSP instance if there is one, but it is also W[1]-hard to distinguish between instances that have a satisfying assignment from instances where every assignment violates at least an ϵ fraction of the constraints. Formally, for every $\epsilon > 0$, we define the promise problem ϵ-gap-BCSP, as BCSP where the input instance is promised to either be satisfiable, or has the property that every assignment violates at least an ϵ fraction of the constraints. The task is to determine whether the input instance is satisfiable or not.

Hypothesis 1. (**Parameterized Inapproximability Hypothesis (PIH)**) *There exists an* $\epsilon > 0$ *such that* ϵ-gap-BCSP *is* W[1]*-hard.*

PIH is related to the recently introduced Gap-Exponential Time Hypothesis (Gap-ETH) [19, 36], which is a stronger version of the well-known Exponential Time Hypothesis [24, 25]. Moreover, Gap-ETH has been instrumental in establishing several parameterized inapproximability results in recent years [6, 5, 10].

We remark that for purposes of showing hardness of approximation results starting from Gap-ETH, we could just as well have conjectured that there exists an $\epsilon > 0$ such that there is no $f(k)n^{\mathcal{O}(1)}$ time algorithm for ϵ-gap-BCSP. This is because (see Section 4.2 for a simple proof) assuming Gap-ETH, there exists an $\epsilon > 0$ such that there is no $f(k)n^{\mathcal{O}(1)}$ time algorithm for ϵ-gap-BCSP [1]. Thus, (essentially) any hardness of approximation result assuming PIH can be shown assuming Gap-ETH instead. In addition, the recent result of Bhattacharya et al. [5] on the parameterized inapproximability of the famous Even Set problem is based on this assumption which is slightly weaker than that in the statement of PIH above. However, we strongly believe that PIH is true as stated—indeed, we should hardly claim this conjecture as our own, as quite a few researchers in parameterized complexity have stated this conjecture as a natural formulation of a PCP-theorem in the context of parameterized inapproximability.

Our final result is that assuming either PIH or Gap-ETH, there exists an $\epsilon > 0$ such that DOCT does not admit an FPT-approximation algorithm with ratio $1+\epsilon$.

Theorem 1.3. *Assuming* Gap-ETH *or* PIH *and FPT* $\neq$ W[1]*, there exists an* $\epsilon > 0$ *such that* DOCT *does not admit an FPT-approximation algorithm with approximation ratio* $1 + \epsilon$.

Arc-Directed Odd Cycle Transversal. We remark that easy reductions transfer all of our results to Arc-DOCT, the "arc" version of DOCT where the goal is to remove at most k arcs such that the resulting graph does not have any directed odd cycles. To transfer the hardness results we need to reduce DOCT to Arc-DOCT. For this purpose, it is sufficient to subdivide every arc, and then split every original vertex u of the input digraph into two vertices, u_{in} and u_{out}, such that all arcs leading into u lead into u_{in} instead, all arcs leading out of u lead out of u_{out} instead, and adding the arc $u_{in}u_{out}$. To transfer the algorithmic results from DOCT to Arc-DOCT, we need to reduce Arc-DOCT to DOCT. This is achieved by subdividing every arc twice, and then making each original vertex undeletable by adding $k+1$ copies of it.

[1]Bhattacharya et al. have also included a proof of this statement in [4].

We now outline our methodology.

W[1]-hardness. The starting point for both our hardness results as well as our approximation algorithm is a failed attempt at obtaining an FPT algorithm. The root of this attempt was the FPT algorithm for DFVS by Chen et al. [8] (see also the textbook [15] for a more gentle exposition). The key concept in this algorithm is the notion of *important separators*, defined by Marx [37]. Given a digraph D and two vertices u and v, a *u-v-separator* is a vertex set $S \subseteq V(D) \setminus \{u, v\}$ such that there is no directed path from u to v in $D - S$. A u-v-separator S is called a *minimal u-v-separator* if no proper subset of S is also a u-v-separator.

Given a vertex set S such that u is not in S, we define the *reach of u in $D - S$* as the set $R_D(u, S)$ of vertices reachable from u by a directed path in $D - S$. We can now define a partial order on the set of minimal u-v separators as follows. Given two minimal u-v separators S_1 and S_2, we say that S_1 is "at least as good as" S_2 if $|S_1| \leq |S_2|$ and $R_D(u, S_2) \subseteq R_D(u, S_1)$. In plain words, S_1 "costs no more" than S_2 in terms of the number of vertices deleted, and S_1 "is pushed further towards v" than S_2. A minimal u-v separator S is an *important u-v-separator* if no other minimal u-v-separators is at least as good as S. The key insight behind the algorithm for DFVS by Chen et al. [8], as well as algorithms for several other parameterized problems [13, 12, 16, 29, 30, 33, 35, 34, 41], is that for every k, the number of important u-v-separators of size at most k is at most 4^k [7]. We refer the reader to the textbook by Cygan et al. [15] for a more thorough exposition of important separators.

Applying the initial steps of the DFVS algorithm to DOCT (i.e. the methods of iterative compression, and guessing an order on an undeletable solution), one naturally arrives at an extension of the notion of important separators. Let us define the *cleaning cost* of a minimal u-v separator S as $\mathsf{doct}(D[R_D(u, S)])$, where $\mathsf{doct}(D)$ is the minimum size of a directed odd cycle transversal of D. Then, we define a new partial order on minimal u-v separators. Here, given two minimal u-v separators S_1 and S_2, we say that S_1 is "at least as good as" S_2 if $|S_1| \leq |S_2|$, $R_D(u, S_2) \subseteq R_D(u, S_1)$, and the cleaning cost of S_1 is at most the cleaning cost of S_2. In other words, S_1 costs no more than S_2, S_1 is pushed further towards v than S_2, and "cleaning up" the reach of u in $G - S_1$ does not cost more than cleaning up the reach of u in $G - S_2$. We say that a minimal u-v separator S is a DOCT-*important u-v-separator* if no minimal u-v-separators other than S are at least as good as S with respect to this new partial order.

For every digraph D, vertices u and v and integer k, we know that there are at most 4^k important u-v separators of size at most k. For the purposes of an FPT algorithm for DOCT, the pivotal question becomes whether the number of DOCT-important u-v-separators of size at most k_1 and cleaning cost at most k_2 can be upper bounded by a function of k_1 and k_2 only, or if there exist families of graphs where the number of DOCT-important u-v-separators of size at most k_1 and cleaning cost at most k_2 grows with the size of the graphs. Indeed, a constructive upper bound on $f(k_1, k_2)$, the number of DOCT-important u-v-separators of size at most k_1 and cleaning cost at most k_2, would have implied an FPT algorithm for DOCT.

We managed to prove that there exists a function f such that the number of DOCT-important u-v-separators of size at most k and cleaning cost 0 is at most $f(k)$. In a subsequent attempt to similarly upper bound the number of DOCT-important u-v-separators of size at most k and cleaning cost 1, we discovered the *clock gadgets* (see Section 3.2), which are graphs where the number of DOCT-important u-v-separators of size at most 2 and cleaning cost 1 is $\Omega(n)$.

A clock gadget essentially permits us to encode (in the language of DOCT) the choice of one element out of a domain of size n, without it being clear a priori which element(s) should be the best one(s) to select. For many problems, once one has such a selection gadget it is easy to prove W[1]-hardness by reducing from BCSP (or, equivalently, from Multicolored Clique). However, we were able to show that on graphs consisting only of clock gadgets glued together in the most natural way, DOCT is in fact FPT[2]. In particular, clocks do not provide a general way of synchronizing the choices of different elements, making it difficult to encode the constraints of BCSP using DOCT. We were able to engineer such a synchronization gadget by a non-trivial modification of the "grid gadget" used by Pilipczuk and Wahlström [43] to show W[1]-hardness of Directed Multicut with four terminal pairs. At this point one can complete a reduction from BCSP using clocks to encode the selection of a value for each variable and using synchronization gadgets to encode the constraints of the BCSP instance.

FPT-Approximation. The hardness of DOCT comes from the fact that DOCT-important u-v-separators have to do two jobs at the same time. First, they need to disconnect v from u, and second they need to clean the reach of u from directed odd cycles. Our approximation algorithm works by delegating the two jobs to different solutions, and solving each of the jobs separately and optimally.

[2]Because this is such a specialized graph class, we did not include a proof of this fact in the paper.

Just like our W[1]-hardness proof, our FPT-approximation for DOCT builds on the algorithm of Chen et al. [8] for DFVS. The method of iterative compression (see [15, 20]) allows us to reduce the original problem to the setting where we are given a digraph D, an integer k, and a directed odd cycle transversal $\hat{S}$ of size $2k+1$. The task is to either determine that D does not have a directed odd cycle transversal of size at most k, or output a directed odd cycle transversal of size at most $2k$. We now proceed with a sketch of how to solve this task in FPT time.

In order to witness that a digraph D has no directed odd cycles it is sufficient to partition the vertex set of D into sets $Z_1, Z_2, \ldots, Z_\ell$ such that (a) no arc goes from Z_i to Z_j with $j < i$ and (b) for every $i \leq \ell$ the underlying undirected graph of $D[Z_i]$ is bipartite. To certify (b) it is sufficient to provide a coloring of all vertices in D with black or white, such that every arc with both endpoints in Z_i for some i has different colored endpoints. The sets $Z_1, Z_2, \ldots, Z_\ell$ can always be chosen to be the strongly connected components of D, and in this case the ordering $Z_1, Z_2, \ldots, Z_\ell$ can be any topological ordering of the directed acyclic graph obtained from D by collapsing every strongly connected component to a vertex.

Suppose now that D has a directed odd cycle transversal S of size at most k. Let $Z_1, Z_2, \ldots, Z_\ell$ be an ordered partitioning of $V(D-S)$ and $\phi : V(D-S) \to \{\text{black, white}\}$ be a coloring that certifies that $D - S$ does not have directed odd cycles. At the cost of a 3^k overhead in the running time we can guess for each vertex $v \in \hat{S}$ whether it is deleted (i.e put in the directed odd cycle transversal), colored black or colored white. At the cost of an additional $k!$ overhead in the running time we can guess for every pair of vertices u, v in $\hat{S}$ whether they occur in the same strongly connected component Z_i, and if not, which of the two strongly connected components containing u and v respectively occurs first in the ordering $Z_1, Z_2, \ldots, Z_\ell$. Applying these guesses together with some simple reduction rules, we end up in the following setting. The input is a digraph D, an integer k and a set $\hat{S}$ such that $D - \hat{S}$ contains no directed odd cycles, and $D[\hat{S}]$ is an acyclic tournament (that is, there is an arc between every pair of vertices in S). The task is to either find a set $S \subseteq V(D) \setminus \hat{S}$ of size at most $2k$ such that **(a)** S is a directed odd cycle transversal, and **(b)** no strongly connected component of $D - S$ contains more than one vertex of $\hat{S}$, or to conclude that no such set of size at most k exists.

A set S that only satisfies **(b)** is called a *skew separator* for $\hat{S}$, and the main subroutine in the algorithm of Chen et al. [8] for DFVS is an algorithm that given D, $\hat{S}$ and k, runs in time $\mathcal{O}(4^k k^{\mathcal{O}(1)}(n+m))$, and finds a skew separator S for $\hat{S}$ of size at most k if such a skew separator exists. Our approximation algorithm runs this subroutine and either finds a skew separator S of size at most k, or concludes that no set of size at most k can satisfy both **(a)** and **(b)** (since there is no such set that satisfies just **(b)**). It then determines in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(1)}$ whether $D - S$ has a directed odd cycle transversal of size at most k disjoint from $\hat{S}$. If such a set S^* exists, then the algorithm outputs $S \cup S^*$ as a solution of size at most $2k$ that satisfies **(a)** and **(b)**. If no such directed odd cycle transversal S^* exists, then the approximation algorithm concludes that no set of size at most k can satisfy both **(a)** and **(b)** (since there is no such set that satisfies just (a) in a subgraph of D). All that remains is to describe the algorithm for finding in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(1)}$ a directed odd cycle transversal S^* of size at most k disjoint from $\hat{S}$ in $D - S$, or determining that such a set does not exist.

At this point we observe that the problem breaks up into independent sub-problems for each strongly connected component of $D - S$. For each such component C we have that $|C \cap \hat{S}| \leq 1$, because S is a skew separator for $\hat{S}$. Since $\hat{S}$ is a directed odd cycle transversal for D, if $C \cap \hat{S} = \emptyset$ then there can be no directed odd cycles in $D[C]$. Hence we concentrate on the case when $C \cap \hat{S} = \{w\}$ for a vertex w. In other words, we are down to the case where the input is a digraph D, integer k and a vertex w such that $\{w\}$ is a directed odd cycle transversal for D. The task is to find a directed odd cycle transversal S^* of D of size at most k with $w \notin S^*$.

Define the *shadow* of S^* to be the set of all vertices of $D - S^*$ that are not in the strongly connected component of $D - S^*$ containing w. Using the technique of *shadow removal*, introduced by Marx and Razgon [41] (see also [13, 12, 11]) in their FPT algorithm for MULTICUT, we can reduce the problem to the special case where the shadow of S^* is empty, at the cost of a $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(1)}$ overhead in the running time. In this special case $D - S^*$ is strongly connected, and therefore the underlying undirected graph of $D - S^*$ is bipartite. Thus, S^* is an *undirected* odd cycle transversal for the underlying undirected graph of D. Here we can apply any one of the numerous FPT algorithms [26, 31, 44, 45] for OCT. Thus we can find optimal directed odd cycle transversals in FPT time for the case when a single undeletable vertex is a directed odd cycle transversal, and as discussed above, this is sufficient to complete the factor 2 FPT-approximation.

As a subroutine of our FPT-approximation we gave an FPT algorithm for DOCT for the special case where an undeletable directed odd cycle transversal of size 1 is given as input. Our hardness result also holds for the case where an undeletable directed odd cycle transversal of size 3 is given as input (the vertices $\{x, y, z\}$ in the construction). Therefore, the parameterized complexity of the case when one also has an undeletable directed odd cycle transversal of size 2 in the input, is an interesting open problem. It is conceivable that an FPT algorithm for this case could help in obtaining an FPT-approximation for DOCT with a factor better than 2.

FPT-Inapproximability. For every $\epsilon > 0$ there exists a $\delta > 0$ such that our first reduction, which proves the W[1]-hardness of DOCT, also translates the hardness of ϵ-GAP-BCSP into hardness of distinguishing between digraphs D such that $\mathsf{doct}(D) \leq k$ from digraphs D such that $\mathsf{doct}(D) > k(1 + \delta)$. However, the reduction only works if in the instance of ϵ-GAP-BCSP every variable occurs in at most three constraints. To complete the proof of the parameterized inapproximability of DOCT, we need to reduce ϵ-GAP-BCSP to this special case. We achieve this by replacing every "high degree" variable by a group of independent low degree variables, while ensuring that the low degree variables all get the same value by introducing a constant degree expander of equality constraints between them.

2 Preliminaries

We use the notations $[t]$ and $[t]_0$ as shorthands of $\{1, 2, \dots, t\}$ and $\{0, 1, \dots, t\}$, respectively. Given a function $f : A \to \mathbb{R}$ and a subset $A' \subseteq A$, denote $f(A') = \sum_{a \in A'} f(a)$.

Parameterized Complexity. Formally, a *parameterization* of a problem is the assignment of an integer k to each input instance. Here, the goal is to confine the combinatorial explosion in the running time of an algorithm for Π to depend only on k. We say that a parameterized problem Π is *fixed-parameter tractable* (FPT) if there exists an algorithm that solves Π in time $f(k) \cdot |I|^{\mathcal{O}(1)}$, where $|I|$ is the size of the input instance and f is an arbitrary computable function depending only on the parameter k.

On the negative side, parameterized complexity also provides methods to show that a problem is unlikely to be FPT. The main technique is that of parameterized reductions analogous to those employed in classical complexity. Here, the concept of W[1]-hardness replaces that of NP-hardness, and we need not only construct an equivalent instance in FPT time, but also ensure that the size of the parameter in the new instance depends only on the size of the parameter in the original instance. For our purposes, it is sufficient to note that if there exists such a reduction transforming a problem known to be W[1]-hard to another problem Π, then the problem Π is W[1]-hard as well. Central W[1]-hard-problems include, for example, the problem of deciding whether a nondeterministic single-tape Turing machine accepts within k steps, the CLIQUE problem parameterized be solution size, and the INDEPENDENT SET problem parameterized by solution size

In the context of a parameterized minimization problem Π, we say that an algorithm for Π is an α-*approximation algorithm* if it always outputs a solution of size at most αk when there exists a solution of size at most k (in other words, the input instance is a yes-instance), and it always outputs NO when there does not not exist a solution of size at most αk. Additional details can be found in the monographs [22, 42, 20, 15].

Digraphs. We refer to standard terminology from the book of Diestel [18] for those graph-related terms that are not explicitly defined here. We say that X is a *minimal* directed odd cycle transversal of D if no proper subset of D is also a directed odd cycle transversal of D. Finally, we call X a *minimum* directed odd cycle transversal of D if there is no directed odd cycle transversal of D whose size is strictly smaller than the size of X. In the context of DOCT, we use the terms *solution* and α-*approximate solution* to refer to directed odd cycle transversals of sizes at most k and at most αk, respectively.

Given a vertex set $X \subseteq V(D)$, we let $D[X]$ denote the subgraph of D induced by X, and we define $D \setminus X = D[V(D) \setminus X]$. Given an arc $(u, v) \in A(D)$, we refer to u as the *tail* of the arc and to v as the *head* of the arc. Given a vertex set $X \subseteq V(G)$, we use $N^+(X)$ to denote the set of out-neighbors of X and $N^-(X)$ to denote the set of in-neighbors of X. We use $N^i[X]$ to denote the set $X \cup N^i(X)$ where $i \in \{+, -\}$. We denote by $A[X]$ the subset of edges in $A(D)$ with both endpoints in X. A *strongly connected component* of D is a maximal subgraph in which every vertex has a directed path to every other vertex. We say that a strongly connected component is *non-trivial* if it consists of at least two vertices and *trivial* otherwise. For disjoint vertex sets X and Y, the set Y is said to be *reachable from* X if for *every* vertex $y \in Y$, there exists a vertex $x \in X$ such that the D contains a directed path from x to y. For disjoint subsets $X, Y, Z \subseteq V(D)$, we call Z an X-Y *separator* if there is no path from a vertex of X to a vertex of Y in $D - Z$. Our proofs rely on the following well-known proposition (see, e.g., [3]).

PROPOSITION 2.1. (FOLKLORE) *Let D be a strongly connected directed graph that does not contain a directed odd cycle. Then, the underlying undirected graph of D is a bipartite graph.*

3 W[1]-Hardness

In this section, we resolve the question of the parameterized complexity of DOCT. More precisely, we prove Theorem 1.1.

The source of our reduction is the Partitioned Subgraph Isomorphism (PSI) problem. The definition of this problem relies on the notion of a *colorful mapping*. Given undirected graphs H and G where G has maximum degree 3, and a coloring function $col : V(H) \to V(G)$, we say that an injective function $\varphi : V(G') \to V(H)$ is a *colorful mapping of G' into H*, where G' is a subgraph of G, if for every $v \in V(G')$, $col(\varphi(v)) = v$, and for every $\{u, v\} \in E(G')$, $\{\varphi(u), \varphi(v)\} \in E(H)$. Formally, the PSI problem is defined as follows. This input is a pair of undirected graphs H and G, and a coloring function $col : V(H) \to V(G)$. The maximum degree of a vertex of G is 3. The goal is to decide whether there exists a colorful mapping of G into H?

While the PSI problem requires us to map the entire graph G, to prove our inapproximability result we would also be interested in colorful mappings of *subgraphs* of G. In the context of the PSI problem, we rely on a well-known proposition due to Marx [38], where the same problem is called Colored Subgraph Isomorphism.

Proposition 3.1. (Corollary 6.3, [38]) *The* PSI *problem is* W[1]-hard*. Moreover, unless* ETH *fails,* PSI *cannot be solved in time $f(k)n^{o(\frac{k}{\log k})}$ for any function f where $k = |E(G)|$. Here, $n = |V(H)|$.*

We anticipate that the components introduced by our proof will be useful for other reductions that aim to establish the W[1]-hardness of problems involving parities and/or cuts. Hence, we have structured our proof as follows. First, for the sake of clarity of the proof, we integrate arc and vertex annotations into the definition of DOCT. Then, we introduce the concept of a *clock*, which is a gadget that lies at the heart of our reduction. This gadget captures the power of parities in a compact, easy-to-use manner. In particular, it elegantly encodes the selection of two (not necessarily distinct) indices from a set $[n]$ whose sum is upper bounded by $n+1$ (in the case of a *forward clock*) or lower bounded by $n + 1$ (in the case of a *reverse clock*). We remark that the selection is orchestrated by a variable that we call *time*. Next, we "glue" the tips of the *hands* of a forward clock and a reverse clock together as well as attach arcs that connect carefully chosen vertices on these hands to obtain a *double clock*. The double clock is a gadget that both ensures that two clocks show the exact same time and that this time corresponds to the selection of two indices whose sum is *exactly* $n + 1$. Roughly speaking, it is mentally convenient to associate each double clock with a different *time zone* that encodes the selection of *one* element. Here, since our source problem is a graph problem, the natural choice of an element is a vertex. Having established a time zone for each selection of one element, we turn to *synchronize* hands of *different* double clocks. For this purpose, we introduce the *synchronizer*, which is a gadget that resembles a *folded grid*. We remark that this specific gadget is different yet inspired by a folded grid gadget that is the core of the paper [43]. Having double clocks and synchronizers at hand, we are finally able to present the entire reduction in an intuitive (yet precise) manner. Lastly, we prove that our reduction is correct. At this point, having already established key properties of our gadgets, the reverse direction ("solution to DOCT $\to$ solution to PSI") is simple. For the forward direction ("solution to PSI $\to$ solution to DOCT") we exhibit a partition of the vertex set of the output digraph into pairwise-disjoint sets on which we can define a topological order, such that the graph induced by each set can be shown to exclude directed odd cycles.

3.1 Annotations

Let us begin our proof by integrating arc and vertex annotations into the definition of DOCT. More precisely, we generalize DOCT as follows.

In the Annotated DOCT (A-DOCT) problem, we are given a digraph D, a non-negative integer k, a labeling function $\ell : A(D) \to \{0, 1\}$, and a weight function $w : V(D) \to [2k + 1]$. The goal is to decide whether there exists a subset $X \subseteq V(D)$ such that $w(X) \leq k$ and X intersects every directed cycle C of D where $\ell(E(C))$ is odd.

Henceforth, in the context of A-DOCT, the term *directed odd cycle* would refer to a directed cycle such that $\ell(E(C))$ is odd. As we show in this section, it is easy to see that in order to prove Theorems 1.1 and 1.3, we can focus on the A-DOCT problem.

Let us now present our reduction from A-DOCT to DOCT. For this purpose, let (D, k, ℓ, w) be an instance of A-DOCT. Then, we construct an instance $\mathbf{red}(D, k, \ell, w) = (D', k')$ of DOCT as follows. First, set $k' = k$. Let $A_0 = \{a \in A(D) : \ell(a) = 0\}$ and $A_1 = \{a \in A(D) : \ell(a) = 1\}$. Next, define $V(D') = P \cup Q$, where $P = \{p^i_a : i \in [\alpha k + 1], a \in A_0\}$ and $Q = \{q^i_v : i \in [w(v)], v \in V(D)\}$. Finally, we define $A(D') = S \cup T \cup R$, where $S = \{(q^i_v, p^j_a) : q^i_v \in Q, p^j_a \in P, v \text{ is the tail of } a\}$, $T = \{(p^i_a, q^j_v) : p^i_a \in P, q^j_v \in Q, v \text{ is the head of } a\}$ and $R = \{(q^i_u, q^j_v) : (u, v) \in A_1\}$. Clearly, (D', k') can be computed in polynomial time. Notice that this reduction corresponds to first subdividing arcs

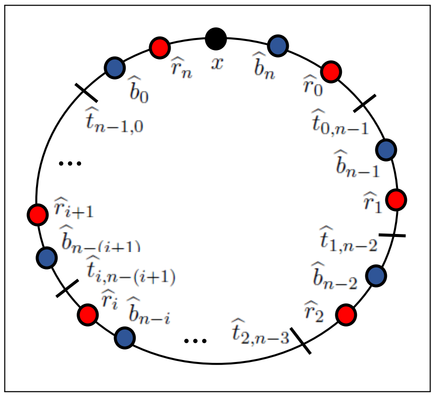

Figure 1: The face of a forward clock. Nodes that have the same shape serve a similar purpose. In particular, some vertices are represented by lines to emphasize that we would sometimes like to delete these vertices, which corresponds to "cutting" the cycle (or path) at this position. The colors distinguish between different types of nodes; red vertices are called r_i, blue vertices are called b_i, etc.

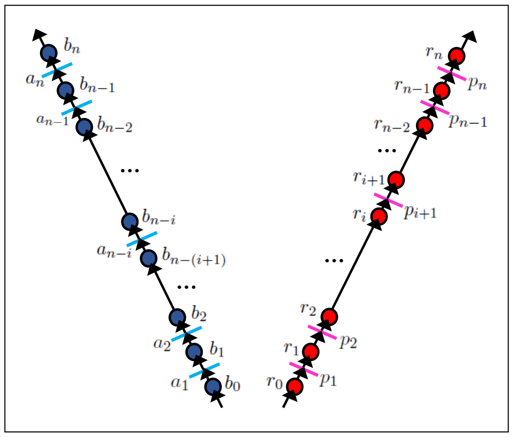

Figure 2: The hands of a forward clock together with the 4 arcs used to attach them. For all $i \in [n]$, $\text{pre}(p_i) = r_{i-1}$ and $\text{post}(p_i) = r_i$, and $\text{pre}(a_i) = b_{i-1}$ and $\text{post}(a_i) = b_i$.

in A_0 once each, then replacing every original vertex with its weight-many copies, and finally replacing every new vertex with $\alpha k + 1$ copies.

LEMMA 3.1. *Let (D, k, ℓ, w) be an instance of* A-DOCT. *If there exists a solution for (D, k, ℓ, w), then there exists a solution for* $\mathbf{red}(D, k, \ell, w) = (D', k')$. *Moreover, if there exists an α-approximate solution for (D', k'), then there exists an α-approximate solution for (D, k, ℓ, w).*

As a corollary to Lemma 3.1, we derive the following result.

COROLLARY 3.1. *For every $\alpha \geq 1$, if there exists an α-approximation algorithm for* DOCT *that runs in time τ, then there exists an α-approximation algorithm for* A-DOCT *that runs in time $\mathcal{O}(\tau + n^{\mathcal{O}(1)})$.*

3.2 The Basic Clock Gadget Let $n, k \in \mathbb{N}$ such that $k \geq 100$. Here, we define an (n, k)*-forward clock* and an (n, k)*-reverse clock*. Since n and k would be clear from context, we simply write *forward clock* and *reverse clock* rather than (n, k)-forward clock and (n, k)-reverse clock, respectively.

3.2.1 Forward Clock Structure. We first define a forward clock C. The *face* of C is an "undirected" cycle, in the sense that consecutive vertices have arcs in both directions. The vertex set of C is the union of four pairwise-disjoint sets $\widehat{R}$ (red), $\widehat{B}$ (blue), $\widehat{T}$ (time) and $\{x\}$. We refer the reader to Fig. 1. We set $\widehat{R} = \{\widehat{r}_i : i \in [n]_0\}$, $\widehat{B} = \{\widehat{b}_i : i \in [n]_0\}$ and $\widehat{T} = \{\widehat{t}_{i,n-i-1} : i \in [n-1]_0\}$. The arc set of the face is the union of the following three pairwise-disjoint sets.

- $\{(x, \widehat{r}_n), (x, \widehat{b}_n), (\widehat{r}_n, x), (\widehat{b}_n, x)\}$.
- $\{(\widehat{b}_i, \widehat{r}_{n-i}) : i \in [n]_0\} \cup \{(\widehat{r}_{n-i}, \widehat{b}_i) : i \in [n]_0\}$.
- $\{(\widehat{r}_i, \widehat{t}_{i,n-i-1}) : i \in [n-1]_0\} \cup \{(\widehat{b}_{n-i-1}, \widehat{t}_{i,n-i-1}) : i \in [n-1]_0\} \cup \{(\widehat{t}_{i,n-i-1}, \widehat{r}_i) : i \in [n-1]_0\} \cup \{(\widehat{t}_{i,n-i-1}, \widehat{b}_{n-i-1}) : i \in [n-1]_0\}$.

The *hands* of C are two directed paths, red and blue (see Fig. 2). The vertex set of the red path is the union of two pairwise disjoint sets, $R = \{r_i : i \in [n]_0\}$ (red) and $P = \{p_i : i \in [n]\}$ (pink). For all $i \in [n]$, we denote $\text{pre}(p_i) = r_{i-1}$ and $\text{post}(p_i) = r_i$. The arc set of the red path is $\{(\text{pre}(p_i), p_i) : i \in [n]\} \cup \{(p_i, \text{post}(p_i)) : i \in [n]\}$. Symmetrically, the blue path is the union of two pairwise disjoint sets, $B = \{b_i : i \in [n]_0\}$ (blue) and $A = \{a_i : i \in [n]\}$ (azure). For all $i \in [n]$, we denote $\text{pre}(a_i) = b_{i-1}$ and $\text{post}(a_i) = b_i$. The arc set of the blue path is $\{(\text{pre}(a_i), a_i) : i \in [n]\} \cup \{(a_i, \text{post}(a_i)) : i \in [n]\}$.

The hands are attached to the face as follows (see Fig. 3). First, we add the arcs (x, r_0) and (x, b_0). Second, for all $i \in [n]_0$, we add the arcs $(r_i, \widehat{r}_i)$ and $(b_i, \widehat{b}_i)$. Then, we "glue" the hands by adding a new vertex, y, and the arcs (r_n, y), (b_n, y) and (y, x).

Finally, let us annotate C (see Fig. 3). The labels of the arcs in the set $\{(x, \widehat{r}_n), (\widehat{r}_n, x), (y, x)\} \cup \{(b_i, \widehat{b}_i) :$

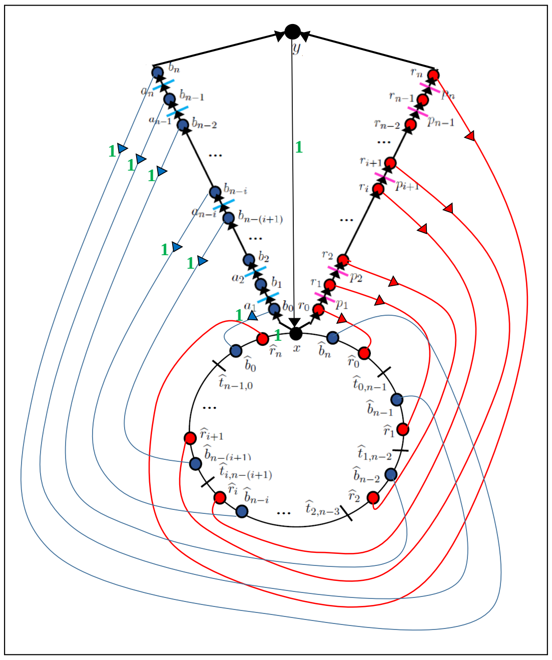

Figure 3: A forward clock. The arcs labeled 1 are marked by a green '1'. The weight of vertices marked by circles is $2k+1$, and the weight of vertices marked by lines is 10.

$i \in [n]_0\}$ are equal to 1, and the labels of all other arcs are equal to 0. Moreover, the weight of the vertices in the set $\widehat{T} \cup P \cup A$ are equal to 10, and the weights of all other vertices are equal to $2k+1$. This completes the description of C. When the clock C is not clear from context, we add the notation (C) to an element (vertex set or vertex) of the clock. For example, we may write $R(C)$ and $x(C)$.

Intuition. It is easy to verify that any solution has to have non-empty intersection with the face of the clock, the cycle starting at x following the blue hand to y and returning to x, and the cycle starting at x following the red hand to y and returning to x. This requires total weight at least 30. The following Lemmata establish the structure of all directed odd cycle transversals of the clock gadget of weight exactly 30, and prove that any directed odd cycle transversal of weight more than 30 must have weight at least 40.

A directed odd cycle transversal of the clock gadget of weight less than 40 must pick one vertex $\hat{t}_{s,n-(s+1)}$ on the face of the clock, one vertex p_i on the red hand, and one vertex a_j on the blue hand. We will show that these three vertices form a directed odd cycle transversal of the clock gadget if and only if $i-1 \leq s$ and $j \leq n-s$. In other words we can think of the variable s as the "time" shown by the clock, and as the time s increases we are allowed to pick the vertex p_i further away from x on the red hand, but at the same time we need to pick the vertex a_j closer to x. This is captured in Definition 3.1 and Lemmata 3.2 and 3.3.

Properties. From the definition of a forward clock, we directly identify which directed odd cycles are present in such a clock.

Observation 3.1. *Let C be a forward clock. The set of directed odd cycles of C is the union of the following sets.*

- **Type 1:** *The set whose only directed odd cycle is the one consisting of the entire red hand and the arc (y, x).*
- **Type 2:** *The set whose only directed odd cycle is the one consisting of the entire blue hand and the arc (y, x).*
- **Type 3:** *For all $i \in [n]_0$, this set contains the directed odd cycle consisting of the directed path from x to r_i on the red hand, the arc $(r_i, \widehat{r}_i)$, and the directed path from $\widehat{r}_i$ to x on the face of the clock that contains the arc $(\widehat{r}_n, x)$.*
- **Type 4:** *For all $i \in [n]_0$, this set contains the directed odd cycle consisting of the directed path from x to b_i on the blue hand, the arc $(b_i, \widehat{b}_i)$, and the directed path from $\widehat{b}_i$ to x on the face of the clock that contains the arc $(\widehat{b}_n, x)$.*
- **Type 5:** *The set whose only directed odd cycle is the face of the clock.*

We proceed to derive properties of "cuts" of a forward clock. To this end, we first need to define the kind of sets using which we would like to "cut" forward clocks.

Definition 3.1. *Let C be a forward clock. We say that a set $X \subseteq V(C)$* cuts C precisely *if there exist $i, j, s \in [n]$ such that $X = \{p_i, a_j, \widehat{t}_{s,n-s-1}\}$, $i-1 \leq s$ and $j \leq n-s$.*

Definition 3.1 directly implies the following observation.

Observation 3.2. *Let C be a forward clock. If $X = \{p_i, a_j, \widehat{t}_{s,n-s-1}\}$ is a set that cuts C precisely, then $i + j \leq n+1$.*

We are now ready to present the desired properties of "cuts" of a forward clock.

Lemma 3.2. *Let C be a forward clock. A set $X \subseteq V(C)$ is a directed odd cycle transversal of C of weight exactly 30 if and only if X cuts C precisely.*

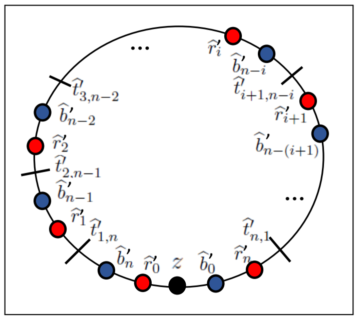

Figure 4: The face of a reverse clock.

Lemma 3.3. *Let C be a forward clock. The weight of a set $X \subseteq V(C)$ that is a directed odd cycle transversal of C but does not cut C precisely is ≥ 40.*

3.2.2 Reverse Clock A reverse clock is simply a forward clock where the directions of *all* arcs have been reversed. For readability reasons it is useful to re-name the vertices (so that, for example, we avoid arcs going from *post* to *pre*), and modify the numbering of the vertices. The proofs in this subsection are identical to the proofs in Subsection 3.2.1, up to permutation of vertex names, changing "from" to "to" and vice versa, and changing the indices. We include the proofs and figures for completeness and ease of reference, since changing the indices slightly changes the equations.

Structure. The *face* of a reverse clock C is an "undirected" cycle whose vertex set is the union of four pairwise-disjoint sets $\widehat{R}'$ (red), $\widehat{B}'$ (blue), $\widehat{T}'$ (time) and $\{z\}$. We refer the reader to Fig. 4. We set $\hat{R}' = \{\widehat{r}'_i : i \in [n]_0\}$, $\hat{B}' = \{\widehat{b}'_i : i \in [n]_0\}$ and $\widehat{T} = \{\widehat{t}'_{i,n-i+1} : i \in [n]\}$. The arc set of the face is the union of the following three pairwise-disjoint sets.

- $\{(z, \widehat{r}'_0), (z, \widehat{b}'_0), (\widehat{r}'_0, z), (\widehat{b}'_0, z)\}$.
- $\{(\widehat{b}'_i, \widehat{r}'_{n-i}) : i \in [n]_0\} \cup \{(\widehat{r}'_{n-i}, \widehat{b}'_i) : i \in [n]_0\}$.
- $\{(\widehat{r}'_i, \widehat{t}'_{i,n-i+1}) : i \in [n]\} \cup \{(\widehat{b}'_{n-i+1}, \widehat{t}'_{i,n-i+1}) : i \in [n]\} \cup \{(\widehat{t}'_{i,n-i+1}, \widehat{r}'_i) : i \in [n]\} \cup \{(\widehat{t}'_{i,n-i+1}, \widehat{b}'_{n-i+1}) : i \in [n]\}$.

The *hands* of C are two directed paths, red and blue. These hands are defined exactly as the hands of a forward clock, except that tags are added to the names of all of their vertices (see Fig. 5). The hands are attached to the face as follows (see Fig. 6). First, we add the arcs (r'_n, z) and (b'_n, z). Second, for all $i \in [n]_0$, we add the arcs $(\widehat{r}'_i, r'_i)$ and $(\widehat{b}'_i, b'_i)$. Then, we "glue" the hands by adding a new vertex, y, and the arcs (y, r'_0), (y, b'_0) and (z, y).

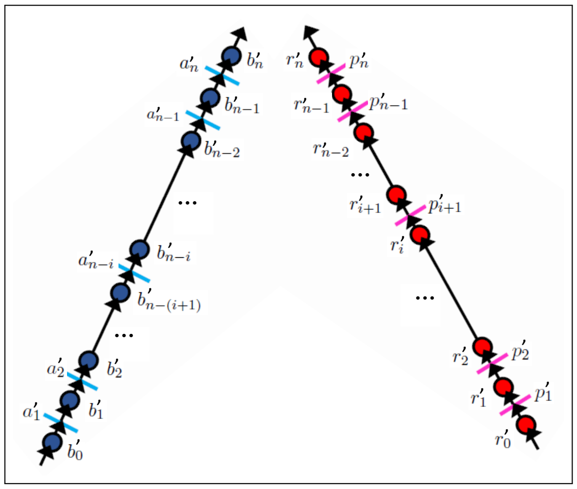

Figure 5: The hands of a reverse clock. For all $i \in [n]$, $\text{pre}(p'_i) = r'_{i-1}$ and $\text{post}(p'_i) = r'_i$, and $\text{pre}(a'_i) = b'_{i-1}$ and $\text{post}(a'_i) = b'_i$.

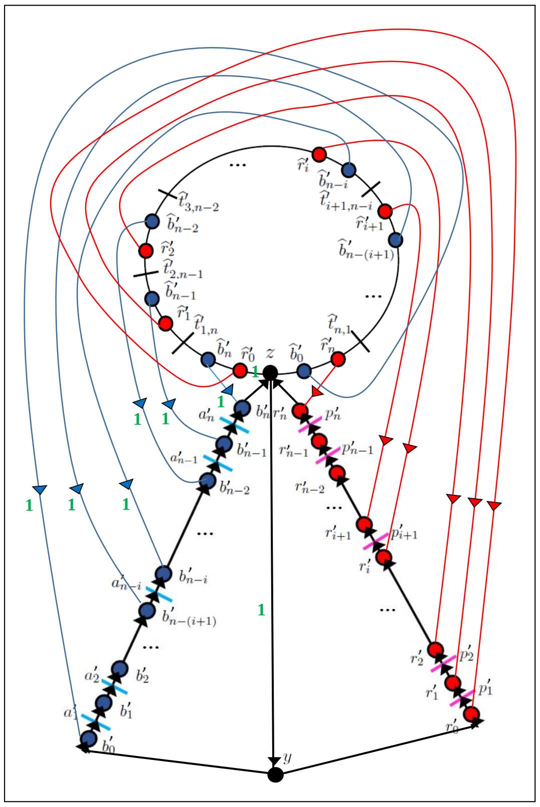

Figure 6: A reverse clock. The arcs labeled 1 are marked by a green '1'. The weight of vertices marked by circles is $2k+1$, and the weight of vertices marked by lines is 10.

Finally, let us annotate C (see Fig. 6). The labels of the arcs in the set $\{(z, \widehat{r}'_0), (\widehat{r}'_0, z), (z, y)\} \cup \{(\widehat{b}'_i, b'_i) :$

$i \in [n]_0\}$ are equal to 1, and the labels of all other arcs are equal to 0. Moreover, the weight of the vertices in the set $\widehat{T}' \cup P' \cup A'$ are equal to 10, and the weights of all other vertices are equal to $2k + 1$. This completes the description of C.

Properties. From the definition of a reverse clock, we directly identify which directed odd cycles are present in such a clock.

Observation 3.3. *Let C be a reverse clock. The set of directed odd cycles of C is the union of the following sets:*

- **Type 1:** *The set whose only directed odd cycle is the one consisting of the entire red hand and the arc (z, y).*
- **Type 2:** *The set whose only directed odd cycle is the one consisting of the entire blue hand and the arc (z, y).*
- **Type 3:** *For all $i \in [n]_0$, this set contains the directed odd cycle consisting of the arc $(\widehat{r}'_i, r'_i)$, the directed path from r'_i to z on the red hand, and the directed path from z to $\widehat{r}'_i$ on the face of the clock that contains the arc $(z, \widehat{r}'_0)$.*
- **Type 4:** *For all $i \in [n]_0$, this set contains the directed odd cycle consisting of the arc $(\widehat{b}'_i, b'_i)$, the directed path from b'_i to z on the blue hand, and the directed path from z to $\widehat{b}'_i$ on the face of the clock that contains the arc $(z, \widehat{b}'_0)$.*
- **Type 5:** *The set whose only directed odd cycle is the face of the clock.*

As in the case of a forward clock, we proceed to derive properties of "cuts" of a reverse clock. To this end, we first need to define the kind of sets using which we would like to "cut" reverse clocks.

Definition 3.2. *Let C be a reverse clock. We say that a set $X \subseteq V(C)$* cuts C precisely *if there exist $i, j, s \in [n]$ such that $X = \{p'_i, a'_j, \widehat{t}'_{s,n-s+1}\}$, $s \leq i$ and $n-s+1 \leq j$.*

Definition 3.2 directly implies the following observation. Note that due to our placement of indices, the inequality is complementary to the one in Observation 3.2.

Observation 3.4. *Let C be a reverse clock. If $X = \{p'_i, a'_j, \widehat{t}'_{s,n-s-1}\}$ is a set that cuts C precisely, then $i + j \geq n + 1$.*

We are now ready to present the desired properties of "cuts" of a reverse clock.

Lemma 3.4. *Let C be a reverse clock. A set $X \subseteq V(C)$ is a directed odd cycle transversal of C of weight exactly 30 if and only if X cuts C precisely.*

Lemma 3.5. *Let C be a reverse clock. The weight of a set $X \subseteq V(C)$ that is a directed odd cycle transversal of C but does not cut C precisely is at least 40.*

3.3 The Double Clock Gadget Structure. Roughly speaking, an (n, k)-*double clock* is the result of gluing the tips of the hands of an (n, k)-forward clock and an (n, k)-reverse clock together as well as adding a 1-labeled arc from every vertex on the hands of the reverse clock to its "twin" on the forward clock. In what follows, since n and k would be clear from context, we omit explicit references to (n, k). Formally, a double clock $\widetilde{C}$ is defined as the digraph obtained as follows. Let C be a forward clock, and let C' be a reverse clock. Identify the vertex y of both of these clocks. All other vertices are distinct. Now, for all $i \in [n]_0$, add the arcs (r'_i, r_i) and (b'_i, b_i), and let the labels of both of these arcs be 1 .

Properties. By the definition of a double clock, we first directly identify which directed odd cycles are present in such a clock.

Observation 3.5. *Let $\widetilde{C}$ be a double clock. The set of directed odd cycles of $\widetilde{C}$ is the union of the following sets.*

- **Forward:** *The set of directed odd cycles completely contained in the forward clock (see Observation 3.1).*
- **Reverse:** *The set of directed odd cycles completely contained in the reverse clock (see Observation 3.3).*
- **Double Red:** *For all $i \in [n]_0$, this set contains the direct odd cycle consisting of the arc (r'_i, r_i), the directed path from r_i to y on the red hand of the forward clock, and the directed path from y to r'_i on the red hand of the reverse clock.*
- **Double Blue:** *For all $i \in [n]_0$, this set contains the direct odd cycle consisting of the arc (b'_i, b_i), the directed path from b_i to y on the blue hand of the forward clock, and the directed path from y to b'_i on the blue hand of the reverse clock.*

We proceed to derive properties of "cuts" of a double clock. To this end, we again first need to define the kind of sets using which we would like to "cut" double clocks.

Definition 3.3. *Let $\widetilde{C}$ be a double clock. We say that a set $X \subseteq V(\widetilde{C})$* cuts *$\widetilde{C}$* precisely *if there exists $i \in [n]$ such that $X = \{p_i, a_{n-i+1}, p'_i, a'_{n-i+1}, \widehat{t}_{i-1,n-i}, \widehat{t'}_{i,n-i+1}\}$.*

We are now ready to present desired properties of "cuts" of a double clock.

Lemma 3.6. *Let $\widetilde{C}$ be a double clock. A set $X \subseteq V(\widetilde{C})$ is a directed odd cycle transversal of $\widetilde{C}$ of weight exactly 60 if and only if X cuts $\widetilde{C}$ precisely.*

Lemma 3.7. *Let $\widetilde{C}$ be a double clock. The weight of a set $X \subseteq V(\widetilde{C})$ that is a directed odd cycle transversal of $\widetilde{C}$ but does not cut $\widetilde{C}$ precisely is at least 70.*

To analyze structures that combine several double clocks, we need to strengthen the reverse direction of Lemma 3.6. More precisely, we need to derive additional properties of a double clock from which we remove a set that cuts it precisely (in addition to the claim that this graph excludes directed odd cycles). For this purpose, we first introduce the following definition, which breaks a double clock into three "pieces".

Definition 3.4. *Let $\widetilde{C}$ be a double clock, and let X be a set that cuts $\widetilde{C}$ precisely. Then, $\widetilde{C}[X, x]$ denotes the subgraph of $\widetilde{C} \setminus X$ induced by the set of vertices that both can reach x and are reachable from x, and $\widetilde{C}[X, z]$ denotes the subgraph of $\widetilde{C} \setminus X$ induced by the set of vertices that both can reach z and are reachable from z. Moreover, $\widetilde{C}[X, y]$ denotes the subgraph of $\widetilde{C} \setminus X$ induced by the set of vertices that belong to neither $\widetilde{C}[X, x]$ nor $\widetilde{C}[X, z]$.*

Notice that for a double clock $\widetilde{C}$, the only two directed paths from x to y are the red and blue hands of the forward clock of $\widetilde{C}$, the only two directed paths from y to z are the red and blue hands of the reverse clock of $\widetilde{C}$, and all of the directed paths from x to z contain the vertex y. Moreover, to every vertex v on a hand of the forward clock, there exists exactly one directed path from x that avoids y. On the other hand, from the vertex v, note that x is reachable by using an arc to the face of the forward clock (in case of a vertex of weight $2k + 1$) or an outgoing arc followed by an arc to the face of the forward clock (in case of a vertex of weight 10), after which we append either of the two paths from the vertex we have reached on the face of the forward clock to x. A symmetric claim holds in the context of z. In light of these observations, we identify each piece of Definition 3.4 as follows.

Observation 3.6. *Let $\widetilde{C}$ be a double clock, and let $X = \{p_i, a_{n-i+1}, p'_i, a'_{n-i+1}, \widehat{t}_{i-1,n-i}, \widehat{t'}_{i,n-i+1}\}$ be a set that cuts $\widetilde{C}$ precisely. Then, the following three conditions are satisfied.*

1. $V(\widetilde{C}[X, x]) = \widehat{R} \cup \widehat{B} \cup (\widehat{T} \setminus \{\widehat{t}_{i-1,n-i}\}) \cup \{x\} \cup \{p_{i'} \in P : i' < i\} \cup \{r_{i'} \in R : i' < i\} \cup \{a_{i'} \in A : i' < n - i + 1\} \cup \{b_{i'} \in B : i' < n - i + 1\}$.
2. $V(\widetilde{C}[X, y]) = \{y\} \cup \{p_{i'} \in P : i' > i\} \cup \{r_{i'} \in R : i' \geq i\} \cup \{a_{i'} \in A : i' > n - i + 1\} \cup \{b_{i'} \in B : i' \geq n - i + 1\} \cup \{p'_{i'} \in P' : i' < i\} \cup \{r'_{i'} \in R' : i' < i\} \cup \{a'_{i'} \in A' : i' < n - i + 1\} \cup \{b'_{i'} \in B' : i' < n - i + 1\}$.
3. $V(\widetilde{C}[X, z]) = \widehat{R'} \cup \widehat{B'} \cup (\widehat{T'} \setminus \{\widehat{t'}_{i,n-i+1}\}) \cup \{z\} \cup \{p'_{i'} \in P' : i' > i\} \cup \{r'_{i'} \in R' : i' \geq i\} \cup \{a'_{i'} \in A' : i' > n - i + 1\} \cup \{b'_{i'} \in B' : i' \geq n - i + 1\}$.

Furthermore, we notice that the three pieces are locally "isolated" in a double clock. Here, isolation means that there does not exist a vertex in one piece and a vertex in another piece such that the first vertex can reach the second one and vice versa. More precisely, since $\widetilde{C}[X, x]$ and $\widetilde{C}[X, z]$ are strongly connected digraphs, while $(z, y), (y, x) \in A(\widetilde{C} \setminus X)$ and all of the directed paths of $\widetilde{C}$ from x to z contain y, we directly derive the following observation.

Observation 3.7. *Let $\widetilde{C}$ be a double clock, and let X be a set that cuts $\widetilde{C}$ precisely. Then, the following two conditions are satisfied: (1) There do not exist vertices $u \in V(\widetilde{C}[X, x])$ and $v \in V(\widetilde{C}[X, y]) \cup V(\widetilde{C}[X, z])$ such that there exists a directed path from u to v in $\widetilde{C} \setminus X$. (2) There do not exist vertices $u \in V(\widetilde{C}[X, y])$ and $v \in V(\widetilde{C}[X, z])$ such that there exists a directed path from u to v in $\widetilde{C} \setminus X$.*

We also need to internally analyze each piece separately. To this end, we introduce one additional definition.

Definition 3.5. *Let D be a directed graph, and let $\ell : A(D) \to \{0, 1\}$. We say that a function $f : V(D) \to \{\mathbf{b}, \mathbf{w}\}$ is* ℓ-consistent *for D if for all $(u, v) \in A(D)$, it holds that $\ell(u, v) = 0$ if and only if $f(u) = f(v)$.*

When the graph D is clear from context, we simply write ℓ-consistent rather than ℓ-consistent for D. First, we note the following simple observation, which hints at the relevance of Definition 3.5 to A-DOCT.

Observation 3.8. *Let D be a directed graph, and let $\ell : A(D) \to \{0, 1\}$. If there exists an ℓ-consistent function for D, then D does not contain a directed odd cycle.*

Let us now derive another simple implication of Definition 3.5.

Lemma 3.8. *Let D be a directed graph, $\ell : A(D) \to \{0,1\}$, and D' be some subgraph of D whose underlying undirected graph is connected and which contains only 0-labeled arcs. Then, if D admits an ℓ-consistent function, then D also admits an ℓ-consistent function f such that for all $v \in V(D')$, it holds that $f(v) = \mathbf{b}$.*

We proceed by showing that for (arc-labeled) strongly connected digraphs, we can easily find a consistent function.

Lemma 3.9. *Let D be a strongly connected directed graph, and let $\ell : A(D) \to \{0,1\}$. If D does not contain a directed odd cycle, then D admits a function f that is ℓ-consistent.*

Finally, we are ready to present the last property of a double clock relevant to our work.

Lemma 3.10. *Let $\widetilde{C}$ be a double clock, and let X be a set that cuts $\widetilde{C}$ precisely. Then, the following three conditions are satisfied.*

1. *There exists an ℓ-consistent function f_x for $\widetilde{C}[X,x]$ such that $f_x(x) = \mathbf{b}$ and for every vertex v of $\widetilde{C}[X,x]$ that does not belong to the face of the forward clock, it holds that $f_x(v) = \mathbf{b}$.*
2. *The function that assigns $\mathbf{b}$ to every vertex of $\widetilde{C}[X,y]$ is ℓ-consistent for $\widetilde{C}[X,y]$.*
3. *There exists an ℓ-consistent function f_z for $\widetilde{C}[X,z]$ such that $f_z(z) = \mathbf{b}$ and for every vertex v of $\widetilde{C}[X,z]$ that does not belong to the face of the reverse clock, it holds that $f_z(v) = \mathbf{b}$.*

3.4 The Synchronization Gadget

Let $n,k \in \mathbb{N}$ such that $k \geq 100$, and let $I \subseteq [n]\times[n]$ be a set of pairs of indices. Here, we define an (n,k,I)-*synchronizer*. Since n and k would be clear from context, we simply write I-*synchronizer* rather than (n,k,I)-synchronizer. When I is also clear from context (or immaterial), we omit it as well.

Structure. The *hands* of a synchronizer S are four red directed paths, H, H', $\widetilde{H}$ and $\widetilde{H}'$. The vertex set of H is the union of two pairwise disjoint sets, $R = \{r_i : i \in [n]_0\}$ (red) and $P = \{p_i : i \in [n]\}$ (pink). For all $i \in [n]$, we denote $\mathrm{pre}(p_i) = r_{i-1}$ and $\mathrm{post}(p_i) = r_i$. The arc set of H is $\{(\mathrm{pre}(p_i), p_i) : i \in [n]\} \cup \{(p_i, \mathrm{post}(p_i)) : i \in [n]\}$ (see Fig. 7). The path $\widetilde{H}$ is defined as the path H where we use tilde notation to specify vertices. Similarly, H' and $\widetilde{H}'$ are defined as the path H and $\widetilde{H}$, respectively, where we further use prime notation to specify vertices. The weight of each vertex on these paths is 10, and the label of each arc on these paths is 0. Now, to obtain the

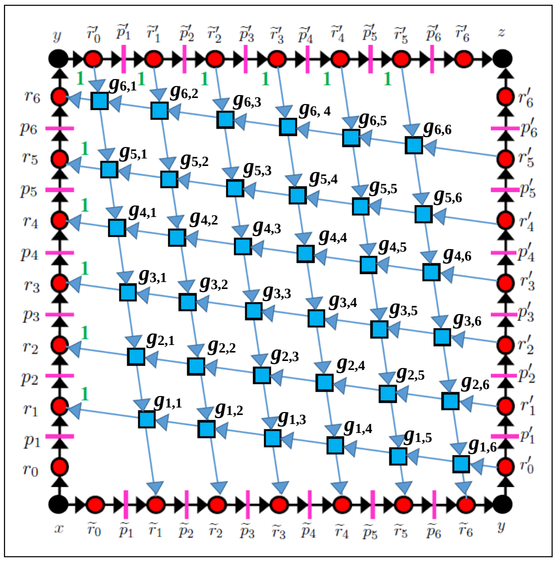

Figure 7: A synchronizer where $n = 6$. The arcs labeled 1 are marked by a green '1'. The weight of vertices marked by circles is $2k+1$, the weight of vertices marked by lines is 10, and the weight of each vertex marked by a square is either $2k+1$ or 1.

frame of S, we add three vertices, x, y and z, each of weight $2k+1$. Moreover, we add the arcs (x, r_0), $(x, \widetilde{r}_0)$, (r_n, y), $(\widetilde{r}_n, y)$, (y, r'_0), $(y, \widetilde{r}'_0)$, (r'_n, z) and $(\widetilde{r}'_n, z)$. The label of each of these arcs is 0. For the sake of clarity of illustrations, the vertex y is drawn twice (see Fig. 7).

Next, we define the *interior* of S (see Fig. 7). Roughly speaking, this part is a grid where each vertex has either a very high weight or a very low weight, depending on whether or not the pair of indices that the vertex represents belongs to I. Formally, the interior of S is the graph G on the vertex set $\{g_{i,j} : i,j \in [n]\}$ and the arc set $\{(g_{i+1,j}, g_{i,j}) : i \in [n-1], j \in [n]\} \cup \{(g_{i,j+1}, g_{i,j}) : i \in [n], j \in [n-1]\}$. The label of each of the arcs is 0. Moreover, for all $i,j \in [n]$, the weight of $g_{i,j}$ is 1 if $(i,j) \in I$ and $2k+1$ otherwise.

Finally, we attach the frame of S to the interior of S (see Fig. 7). To this end, for all $i \in [n]$, we add two arcs labeled 1: $(g_{i,1}, \mathrm{post}(p_i))$ and $(\mathrm{pre}(\widetilde{p}'_i), g_{n,i})$. Moreover, for all $i \in [n]$, we add two arcs labeled 0: $(g_{1,i}, \mathrm{post}(\widetilde{p}_i))$ and $(\mathrm{pre}(p'_i), g_{i,n})$. When the synchronizer S is not clear from context, we add the notation (S) to an element (vertex set or vertex) of the synchronizer.

Properties. By the definition of a synchronizer, we first directly identify which directed odd cycles are present in such a gadget.

Observation 3.9. *Let S be a synchronizer. The set of directed odd cycles of S is the union of the following sets.*

- **Horizontal Match:** *For all $i \in [n]$, this set contains the direct odd cycle consisting of the directed path from y to* $\mathrm{pre}(p'_i)$ *on H', the (unique) directed path from* $\mathrm{pre}(p'_i)$ *to* $\mathrm{post}(p_i)$ *on the interior, and the directed path from* $\mathrm{post}(p_i)$ *to y on H.*

- **Horizontal Mismatch:** *For all $i, j \in [n]$ such that $j < i$, this set contains every direct odd cycle consisting of the directed path from y to* $\mathrm{pre}(p'_i)$ *on H', some directed path from* $\mathrm{pre}(p'_i)$ *to* $\mathrm{post}(p_j)$ *on the interior, and the directed path from* $\mathrm{post}(p_j)$ *to y on H.*

- **Vertical Match:** *For all $i \in [n]$, this set contains the direct odd cycle consisting of the directed path from y to* $\mathrm{pre}(\widetilde{p}'_i)$ *on $\widetilde{H}'$, the (unique) directed path from* $\mathrm{pre}(\widetilde{p}'_i)$ *to* $\mathrm{post}(\widetilde{p}_i)$ *on the interior, and the directed path from* $\mathrm{post}(\widetilde{p}_i)$ *to y on $\widetilde{H}$.*

- **Vertical Mismatch:** *For all $i, j \in [n]$ such that $j < i$, this set contains every direct odd cycle consisting of the directed path from y to* $\mathrm{pre}(\widetilde{p}'_i)$ *on $\widetilde{H}'$, some directed path from* $\mathrm{pre}(\widetilde{p}'_i)$ *to* $\mathrm{post}(\widetilde{p}_j)$ *on the interior, and the directed path from* $\mathrm{post}(\widetilde{p}_j)$ *to y on $\widetilde{H}$.*

We proceed to derive properties of "cuts" of a synchronizer. To this end, we again first need to define the kind of sets using which we would like to "cut" synchronizers.

DEFINITION 3.6. *Let S be an I-synchronizer. We say that a set $X \subseteq V(S)$* cuts S precisely *if there exist $i, j \in [n]$ such that $X = \{p_i, p'_i, \widetilde{p}_j, \widetilde{p}'_j, g_{i,j}\}$ and $(i, j) \in I$.*

DEFINITION 3.7. *Let S be an I-synchronizer. We say that a set $X \subseteq V(S)$* cuts S roughly *if X does not cut S precisely and there exist $i, j \in [n]$ such that $\{p_i, p'_i, \widetilde{p}_j, \widetilde{p}'_j\} \subseteq X$.*

We are now ready to present desired properties of "cuts" of a synchronizer. Unlike the cases of clocks, here we only analyze cuts of the forms presented in Definitions 3.6 and 3.7. The first property follows directly from Definition 3.6.

OBSERVATION 3.10. *The weight of a set that cuts a synchronizer precisely is exactly 41. In particular, the weight of the intersection of this set with the interior is exactly 1.*

Let us now argue that all directed odd cycles are intersected.

LEMMA 3.11. *Let S be a synchronizer, and let X be a set that cuts S precisely. Then, $S \setminus X$ does not contain a directed odd cycle.*

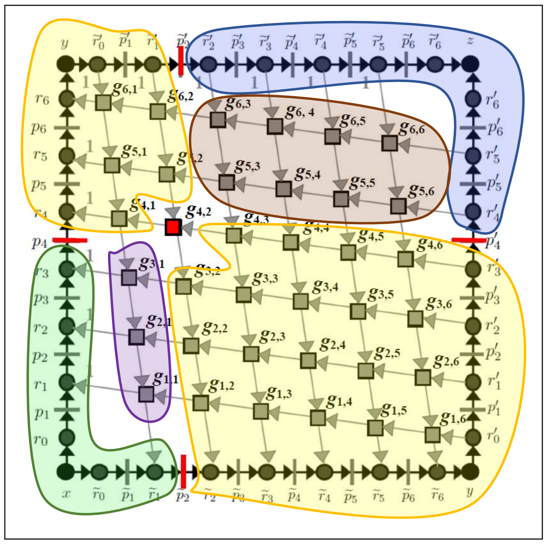

Figure 8: The components $S[X, x]$ (green), $S[X, y]$ (yellow), $S[X, z]$ (blue), $S[X, l_x]$ (purple) and $S[X, l_z]$ (brown), where X is the set of red vertices (see Definition 3.8).

Next, we analyze the weight of rough cuts.

LEMMA 3.12. *Let S be a synchronizer, and let X be a set that cuts S roughly. Then, $w(X) \geq 42$.*

As in the case of the double clock, we need to strengthen Lemma 3.11. For this purpose, we introduce the following definition, which breaks a synchronizer into five "pieces" (see Fig. 8).

DEFINITION 3.8. *Let S be a synchronizer, and let X be a set that cuts S precisely. Then, $S[X, y]$ denotes the subgraph of $S \setminus X$ induced by the set of vertices that both can reach y and are reachable from y. Moreover, $S[X, x]$ denotes the subgraph of $S \setminus X$ induced by the set of vertices reachable from x, and $S[X, z]$ denotes the subgraph of S induced by the set of vertices that can reach z. Finally, $S[X, l_x]$ denotes the subgraph of $S \setminus X$ induced by the set of vertices outside $S[X, x]$ that can reach a vertex of $S[X, x]$ without using any vertex of $S[X, y]$, and $S[X, l_z]$ denotes the subgraph of $S \setminus X$ induced by the set of vertices outside $S[X, z]$ that are reachable from a vertex of $S[X, z]$ without using any vertex of $S[X, y]$.*

Notice that for a synchronizer S, the only two directed paths from x to y are those internally consisting of H and $\widetilde{H}$, the only two directed paths from y to z are those internally consisting of H' and $\widetilde{H}'$, and all of the directed paths from x to z contain the vertex y. Moreover, the vertex y and every vertex $g_{i,j}$ of the interior are both contained in the two following (even) directed cycles, among other directed cycles, whose only common vertices are y and $g_{i,j}$: (i) the directed cycle consisting of the path from y to $\mathrm{pre}(p'_i)$ on H', the (unique) directed path from $\mathrm{pre}(p'_i)$ to $g_{i,j}$ on the interior, the (unique) directed path from $g_{i,j}$ to $\mathrm{post}(\widetilde{p}_j)$ on the interior, and the directed path from $\mathrm{post}(\widetilde{p}_j)$ to y on $\widetilde{H}$; (ii) the directed cycle consisting of the path from y to $\mathrm{pre}(\widetilde{p}'_j)$ on $\widetilde{H}'$, the (unique) directed path from $\mathrm{pre}(\widetilde{p}'_j)$ to $g_{i,j}$ on the interior, the (unique) directed path from $g_{i,j}$ to $\mathrm{post}(p_i)$ on the interior, and the directed path from $\mathrm{post}(p_i)$ to y on H. In light of these observations, we identify each piece of Definition 3.8 as follows (see Fig. 8).

Observation 3.11. *Let S be a synchronizer, and let $X = \{p_i, p'_i, \widetilde{p}_j, \widetilde{p}'_j, g_{i,j}\}$ be a set that cuts S precisely. Then, the following five conditions are satisfied:*

1. $V(S[X,x]) = \{x\} \cup \{p_{i'} \in V(H) : i' < i\} \cup \{r_{i'} \in V(H) : i' < i\} \cup \{\widetilde{p}_{j'} \in V(\widetilde{H}) : j' < j\} \cup \{\widetilde{r}_{j'} \in V(\widetilde{H}) : j' < j\}$.
2. $V(S[X,y]) = \{y\} \cup \{p_{i'} \in V(H) : i' > i\} \cup \{r_{i'} \in V(H) : i' \geq i\} \cup \{\widetilde{p}_{j'} \in V(\widetilde{H}) : j' > j\} \cup \{\widetilde{r}_{j'} \in V(\widetilde{H}) : j' \geq j\} \cup \{p'_{i'} \in V(H') : i' < i\} \cup \{r'_{i'} \in V(H') : i' < i\} \cup \{\widetilde{p}'_{j'} \in V(\widetilde{H}') : j' < j\} \cup \{\widetilde{r}'_{j'} \in V(\widetilde{H}') : j' < j\} \cup (\{g_{i',j'} \in V(G) : i' \leq i, j' \geq j\} \cup \{g_{i',j'} \in V(G) : i' \geq i, j' \leq j\}) \setminus \{g_{i,j}\}$.
3. $V(S[X,z]) = \{z\} \cup \{p'_{i'} \in V(H') : i' > i\} \cup \{r'_{i'} \in V(H') : i' \geq i\} \cup \{\widetilde{p}'_{j'} \in V(\widetilde{H}') : j' > j\} \cup \{\widetilde{r}'_{j'} \in V(\widetilde{H}') : j' \geq j\}$.
4. $V(S[X,l_x]) = \{g_{i',j'} \in V(G) : i' < i, j' < j\}$.
5. $V(S[X,l_z]) = \{g_{i',j'} \in V(G) : i' > i, j' > j\}$.

Furthermore, in light of Observation 3.11, we notice that the five pieces are locally "isolated" in a synchronizer as follows.

Observation 3.12. *Let S be a synchronizer, and let X be a set that cuts S precisely. Then, the following five conditions are satisfied:*

1. *There do not exist vertices $u \in V(\widetilde{C}[X,x])$ and $v \in V(S[X,l_x]) \cup V(S[X,y]) \cup V(S[X,l_z]) \cup V(S[X,z])$ such that there exists a directed path from u to v in $S \setminus X$.*
2. *There do not exist vertices $u \in V(\widetilde{C}[X,l_x])$ and $v \in V(S[X,y]) \cup V(S[X,l_z]) \cup V(S[X,z])$ such that there exists a directed path from u to v in $S \setminus X$.*
3. *There do not exist vertices $u \in V(\widetilde{C}[X,y])$ and $v \in V(S[X,l_z]) \cup V(S[X,z])$ such that there exists a directed path from u to v in $S \setminus X$.*
4. *There do not exist vertices $u \in V(\widetilde{C}[X,l_z])$ and $v \in V(S[X,z])$ such that there exists a directed path from u to v in $S \setminus X$.*

Finally, we need to internally analyze each piece separately.

Lemma 3.13. *Let S be a synchronizer, and let X be a set that cuts S precisely. Then, the following two conditions are satisfied: (1) For each of the graphs $S[X,x], S[X,l_x], S[X,l_z]$ and $S[X,z]$, the function that assigns $\mathbf{b}$ to every vertex of the graph is ℓ-consistent. (2) There exists an ℓ-consistent function f_y for $S[X,y]$ such that for every vertex v of the frame that belongs to $S[X,y]$, it holds that $f_y(v) = \mathbf{b}$.*

3.5 Reduction We are now ready to present the complete reduction from PSI to A-DOCT. For this purpose, let (H, G, col) be an instance PSI. We assume that $|V(G)| \geq 100$, else a solution can be found by brute force in polynomial time. If G contains an isolated vertex to which no vertex in H is mapped by col, then the input instance is a no-instance; otherwise, by removing all of the isolated vertices of G and the vertices of H that are mapped to them, we obtain an instance of PSI that is equivalent to (H, G, col). Thus, we next assume that G does not contain isolated vertices. For all $g \in V(G)$, denote $V^g = \{v \in V(H) : col(v) = g\}$. We next assume that for all $g, g' \in V(G)$, it holds that $|V^g| = |V^{g'}| = n$ (for the appropriate n), else we can add isolated vertices to H to ensure that this condition holds. Then, for all $g \in V^g$, denote $V^g = \{v^g_1, v^g_2, \ldots, v^g_n\}$. Let $<$ be some arbitrary order on $V(G)$.

We construct an instance $\mathbf{red}(H, G, col) = (D, k, \ell, w)$ of A-DOCT as follows. First, we set $k = 60|V(G)| + |E(G)|$. Next, we turn to construct (D, k, ℓ, w). For every $g \in V(G)$, we insert one (n,k)-double clock $\widetilde{C}^g$. For every edge $e = \{g, g'\} \in E(G)$ where $g < g'$, we insert one (n, k, I^e)-synchronizer S^e where $I^e = \{(i,j) : \{v^g_i, v^{g'}_j\} \in E(H)\}$. We identify the vertices x, y and z of all double clocks and synchronizers. That is, we now have a single vertex called x, a single vertex called y and a single vertex called z. Finally, for every edge $e = \{g, g'\} \in E(G)$ where $g < g'$, we identify the red hand of the forward clock of $\widetilde{C}^g$ with the hand H of S^e, the red hand of the reverse clock of $\widetilde{C}^g$ with the hand H' of S^e, the red hand of the forward clock of $\widetilde{C}^{g'}$ with the hand $\widetilde{H}$ of S^e, and the red hand of the reverse clock of $\widetilde{C}^{g'}$ with the hand $\widetilde{H}'$ of S^e. Here, by identifying two directed paths of the same number $2n+1$ of vertices, we mean that for all $i \in [2n+1]$, we identify the ith vertex on one path with the ith vertex on the other path. Consequently, for all $i \in [2n]$, we also identify the ith arc on one path with the ith arc on the other path. We remark that next, when we refer to an element of a specific double clock or a specific synchronizer, we would refer to the new unified vertex. For example, given an edge $e = \{g, g'\} \in E(G)$ where $g < g'$, we have that $r_3(C^g) = r_3(S^e)$ and $r'_5(C^{g'}) = \widetilde{r}'_5(S^e)$. This completes the description of the reduction.

3.6 Correctness It remains to derive the correctness of Theorem 1.1. To this end, first note that since $|V(G)| \geq 100$ and G does not contain isolated vertices, we have the following observation.

OBSERVATION 3.13. *Let (H, G, col) be an instance of* PSI. *Then, for $(D, k, \ell, w) = \mathbf{red}(H, G, col)$, it holds that $100 \leq k \leq 121|E(G)|$.*

Clearly, we also have the following observation.

OBSERVATION 3.14. *Let (H, G, col) be an instance of* PSI. *Then, the instance $\mathbf{red}(H, G, col)$ can be constructed in polynomial time.*

To verify the correctness of the reduction, we first prove the forward direction, summarized in the following lemmata.

LEMMA 3.14. *Let (H, G, col) be a yes-instance of* PSI. *Then, $(D, k, \ell, w) = \mathbf{red}(H, G, col)$ is a yes-instance of* A-DOCT.

We next prove a strengthened version of the reverse direction, which would be necessary to derive our inapproximability result.

LEMMA 3.15. *Fix $\epsilon \geq 0$. There exists $\delta = \delta(\epsilon) \geq 0$, where if $\epsilon > 0$ then $\delta > 0$, such that the following condition holds: Given an instance (H, G, col) of* PSI, *if $\mathbf{red}(H, G, col)$ admits a $(1+\delta)$-approximate solution, then there exists a colorful mapping of a subgraph G' of G into H such that G' contains at least $(1-\epsilon)|E(G)|$ edges.*

As a direct corollary to Lemma 3.15 with $\epsilon = 0$, we have the following result.

COROLLARY 3.2. *If (H, G, col) is a no-instance of* PSI, *then (D, k, ℓ, w) is a no-instance of* A-DOCT.

We are now ready to derive the correctness of Theorem 1.1.

Proof. [Proof of Theorem 1.1] By Lemma 3.14, Corollary 3.2 and Observations 3.13 and 3.14, given any instance (H, G, col) of PSI, we can construct in polynomial time an equivalent instance (D, k, ℓ, w) of A-DOCT such that $k \leq 121|E(G)|$. Thus, by Proposition 3.1, A-DOCT is W[1]-hard. Moreover, by Proposition 3.1, unless ETH fails, A-DOCT cannot be solved in time $f(k) \cdot n^{o(\frac{k}{\log k})}$ for any function f. Hence, by Corollary 3.1, we conclude that Theorem 1.1 is correct. ☐

4 Parameterized Inapproximability

In this section, we prove Theorem 1.3. For convenience, let us restate the theorem below.

THEOREM 4.1. *Assuming* Gap-ETH *or* PIH *and* FPT $\neq$ W[1], *there exists an $\epsilon > 0$ such that* DOCT *does not admit an* FPT*-approximation algorithm with approximation ratio $1 + \epsilon$.*

We first recall basic concepts concerning constraint satisfaction, after which we define the gap-ETH and show that an FPT-approximation algorithm for ϵ-gap-BCSP would violate the gap-ETH. Then we reduce ϵ-gap-BCSP to the special case of this problem where every variable occurs in at most three constraints. Finally we conclude the proof of Theorem 1.3.

4.1 Constraint Satisfaction Given a set of variables $X = \{x_1, x_2, \ldots, x_k\}$ and a family of pairwise-disjoint domains $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$, a *binary constraint* is a pair $c = ((x_i, x_j), R)$ where $x_i, x_j \in X$, $i \neq j$, and R is a binary relation over $D_i \times D_j$. An *evaluation* is a function $\psi : X \to \bigcup \mathcal{D}$ such that for all $x_i \in X$, $\psi(x_i) \in D_i$. An evaluation ψ is said to *satisfy* $((x_i, x_j), R)$ if $(\psi(x_i), \psi(x_j)) \in R$. Moreover, given a multiset C of binary constraints, an evaluation ψ is said to *satisfy* C if it satisfies every constraint $c \in C$. For all $i \in [k]$, let $C_i \subseteq C$ denote the sub(multi)set of constraints where x_i occurs, and let $s_i = |C_i|$. We assume w.l.o.g. that for all $i \in [k]$, $s_i \geq 1$, that is, for every variable in X, there exists at least one binary constraint where it occurs.

Here, we cannot assume w.l.o.g. that C is a set rather than a multiset, or more generally, that for all distinct $i, j \in [k]$, $|C_i \cap C_j| \leq 1$.[3] Indeed, here we do not only care whether we can satisfy C or not (if this were the case, then it would have been trivial to see how, given an instance where $|C_i \cap C_j| \geq 2$ for some $i, j \in [k]$, obtain an equivalent instance with the same set of variables and where $|C_i \cap C_j| \leq 1$ for all $i, j \in [k]$),

[3] In other words, we cannot assume w.l.o.g. that for every pair of variables in X, there exists at most one binary constraint in C where both of these variables occur.

but also what is the fraction of constraints that are satisfied. However, as we would see later, it would actually still be simple to ensure the property above. We do not ensure it just yet, as in the reduction in Section 4.2, we implicitly rely on the possibility to allow pairs of variables to occur in more than one constraint.

The Binary Constraint Satisfaction Problem (BCSP) is defined as follows. The input is a set $X = \{x_1, x_2, \ldots, x_k\}$ of k variables, a family of pairwise-disjoint domains $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$, and a multi-set C of binary constraints. The objective is to decide whether there exists an evaluation that satisfies C?

Recall that the promise problem ϵ-gap-BCSP is defined as BCSP where the input instance is promised to either be satisfiable, or has the property that every evaluation satisfies fewer than $(1-\epsilon)$ fraction of the constraints.

4.2 Hardness of ϵ-gap-BCSP assuming the gap-ETH

First we state the gap-ETH. This definition is taken verbatim from Chalermsook et al. [6]. Max q-SAT is a maximization version of q-SAT that asks to compute the maximum number of clauses in ϕ that can be simultaneously satisfied. We will abuse q-SAT to mean Max q-SAT, and for a formula ϕ, we use $SAT(\phi)$ to denote the maximum number of clauses satisfied by any assignment. The Gap Exponential Time Hypothesis can now be stated in terms of SAT as follows.

Conjecture 4.1. (*(rand.) Gap Exponential-Time Hypothesis* (gap-ETH) [19, 36]) For some constant $\delta, \rho, c > 0$, no algorithm can, given a 3-SAT formula ϕ on n variables and $m \le cn$ clauses, distinguish between the following cases correctly with probability $\ge 2/3$ in $\mathcal{O}(2^{\delta n})$ time: (i) $SAT(\phi) = m$ and (ii) $SAT(\phi) < (1-\rho)m$.

Next we relate gap-ETH with hardness of ϵ-gap-BCSP, the problem underlying PIH.

Theorem 4.2. *Assuming the* gap-ETH *there exists an $\epsilon > 0$ such that there is no FPT-approximation algorithm for ϵ-gap-BCSP.*

Proof. Given ρ, an integer k and a SAT formula ϕ we construct an instance $I = (X, \mathcal{D}, C)$ of ϵ-gap-BCSP with $2k$ variables as follows. We will assume that both the number n of variables and the number m of clauses in ϕ is divisible by k, and that the variables are $v_0, v_1, \ldots, v_{n-1}$ and the clauses are $Q_0, Q_1, \ldots, Q_{m-1}$. We assume without loss of generality that each clause Q_i contains precisely 3 literals, q_i^1, q_i^2, q_i^3.

We group the variables into k groups of n/k variables each: for $0 \le r \le k-1$ the r'th group consists of $\{v_{\frac{rn}{k}}, v_{\frac{rn}{k}+1}, \ldots, v_{\frac{(r+1)n}{k}-1}\}$. Similarly we group the clauses into k groups with m/k clauses in each group as follows. For $0 \le r \le k-1$ the r'th group consists of $\{Q_{\frac{rm}{k}}, Q_{\frac{rm}{k}+1}, \ldots, Q_{\frac{(r+1)m}{k}-1}\}$.

The set X of variables of I is $\{x_0, \ldots, x_{k-1}, y_0, \ldots, y_{k-1}\}$ and the domain of each variable in $x_0, \ldots, x_k$ is $\{0,1\}^{n/k}$. The domain of each variable in $y_0, \ldots, y_{k-1}$ is $\{1,2,3\}^{m/k}$. There is a natural one-to-one correspondence between an assignment of a value in $\{0,1\}^{n/k}$ to x_r and an assignment of truth values to the variables in the r'th group of variables ϕ. In particular the i'th bit of x_r is equal to the value of $v_{\frac{rn}{k}+i}$. We will interpret an assignment of $\{1,2,3\}^{m/k}$ to a variable y_r as follows. If the i'th character of y_r is t then the t'th literal of the clause $Q_{\frac{rm}{k}+i}$ (namely $q^t_{\frac{rm}{k}+i}$) is set to true.

For each clause Q_i of ϕ and literal $q_i^t \in Q_i$ we add a constraint to the instance I. (We remark that it would thus be possible that for a pair of variables, more than one constraint where they occur may be added.) Let $i = \frac{rm}{k} + j$, in other words Q_i is the j'th clause of the r'th group. Similarly, let q_i^t be a literal corresponding to the variable $v_{\frac{sn}{k}+\ell}$, either positively or negatively. Thus q_i^t is a literal of the ℓ'th variable in the s'th variable group of ϕ.

We add a constraint to I between x_s and y_r. If q_i^t is a positive literal, this constraint is satisfied unless the j'th character of y_r is t and the ℓ'th bit of x_r is 0. If q_i^t is a negative literal, this constraint is satisfied unless the j'th character of y_r is t and the ℓ'th bit of x_r is 1. In other words, the constraint is falsified if y_r "claims" that the literal q_i^t is set to true while x_s claims that the variable corresponding to q_i^t is set such that q_i^t is false. The total number of such constraints is exactly $3m$. Thus the number of variables in the constructed instance is $2k$ and the total size of the instance is $\mathcal{O}(k3^{cn/k} + m)$.

For completeness, suppose there is an assignment of truth values to $v_0, \ldots, v_{n-1}$ that satisfies all clauses. Set x_i's according to this assignment. For every clause Q_i there is a literal $q_i^{t_i}$ that is set to true. Let $i = \frac{mr}{k} + j$, set the j'th character of y_r to t_i. By construction all $3m$ constraints of the instance I are satisfied.

For soundness, suppose that there is an assignment to $x_0, \ldots, x_{k-1}$ and $y_0, \ldots, y_{k-1}$ that satisfies at least $(1-\frac{\epsilon}{3})3m$ constraints of I. We consider the assignment to $v_0, \ldots, v_{n-1}$ that corresponds to the assignment to $x_0, \ldots, x_{k-1}$. We argue that at least $(1-\epsilon)m$ clauses are satisfied by this assignment.

There are at most ϵm unsatisfied constraints in I. Each clause Q_i of ϕ gave rise to 3 constraints in I, call the clause Q_i *happy* if all of its 3 constraints are satisfied. Since there are at most ϵm unsatisfied constraints, there are at least $(1-\epsilon)m$ happy clauses. We now argue that all happy clauses are satisfied.

Let Q_i be a happy clause, and let $i = \frac{rm}{k} + j$. Let t_i be the j'th character of y_r. Let $v_{\frac{sn}{k}+\ell}$ be the variable that corresponds to the literal $q_i^{t_i}$. In the construction of the instance I we added a constraint that would be falsified if the j'th character of y_r is t_i and at the same time the ℓ'th bit of x_s was set such that setting $v_{\frac{sn}{k}+\ell}$ to the ℓ'th bit of x_s falsifies the literal $q_i^{t_i}$. Since the clause Q_i is happy this constraint is not falsified and hence the clause Q_i is satisfied.

Suppose now that $\epsilon/3$-GAP-BCSP has an FPT algorithm with running time $f(k)|I|^{\mathcal{O}(1)}$ where $|I|$ is the size of the input instance I. Let $g(k)$ be a non-decreasing function of k that tends to infinity with k such that $f(2 \cdot g(k)) \leq k$.

Construct from ϕ a $\epsilon/3$-GAP-BCSP instance as defined above with $k = g(n)$, run the algorithm for $\epsilon/3$-GAP-BCSP, and return the same answer. The total running time of the algorithm is upper bounded by $f(2k)|I|^{\mathcal{O}(1)} \leq f(g(n)) \cdot (k3^{cn/k} + m)^{\mathcal{O}(1)} \leq n \cdot g(n) \cdot 3^{\mathcal{O}(cn/g(n))} \cdot m^{\mathcal{O}(1)} \cdot 2^{o(n)}$. This contradicts the gap-ETH, concluding the proof. ∎

4.3 Constraint Satisfaction with Bounded Degree

For all $i, j \in [k]$, denote $C_{\{i,j\}} = C_i \cap C_j$, that is, the multiset of constraints in C where x_i and x_j occur together (if a constraint c belongs to $C_{\{i,j\}}$, then its number of occurrences is equal to its number of occurrences in C). Moreover, for all $i, j \in [k]$, denote $s_{\{i,j\}} = |C_{\{i,j\}}|$. Note that $|C| = \sum_{i,j\in[k],i<j} s_{\{i,j\}}$. For a fixed integer d, the ϵ-GAP-BCSP$_d$ is defined as the special case of ϵ-GAP-BCSP where every variable is present in at most d constraints, and for all $i, j \in [k]$, $s_{\{i,j\}} \leq 1$ (that is, every two variables occur together in at most one constraint). Before we turn to prove the hardness of ϵ-GAP-BCSP$_3$, which is the main part of this section, we first simplify the instances of ϵ-GAP-BCSP that we need to analyze.

We prove that in what follows, it would be "safe" to assume that $|C| = k^{\mathcal{O}(1)}$. To this end, we first prove the following simple lemma.

LEMMA 4.1. *There exists a polynomial-time algorithm that, given an instance $I = (X, \mathcal{D}, C)$ of ϵ-GAP-BCSP with $k = |X|$, constructs another instance $\widehat{I} = (X, \mathcal{D}, \widehat{C})$ of ϵ-GAP-BCSP with $|\widehat{C}| = |C|$ and the following properties:*

- *for all $i, j \in [k]$, there is at most one distinct constraint in $\widehat{C}_{\{i,j\}}$;*[4]
- *if I is satisfiable, then $\widehat{I}$ is satisfiable;*
- *if $\widehat{I}$ admits an evaluation that satisfies at least $(1-\epsilon)|\widehat{C}|$ constraints, then I admits an evaluation that satisfies at least $(1-\epsilon)|C|$ constraints.*

[4]The distinct constraint in $\widehat{C}_{\{i,j\}}$ may occur more than once in $\widehat{C}_{\{i,j\}}$, that is, $\widehat{s}_{\{i,j\}}$ may be larger than 1.

We now show how to ensure that $|C|$ is small.

LEMMA 4.2. *There exists a polynomial-time algorithm that, given an instance $I = (X, \mathcal{D}, C)$ of ϵ-GAP-BCSP with $k = |X|$, constructs another instance $\widehat{I} = (X, \mathcal{D}, \widehat{C})$ of $\frac{\epsilon}{2}$-GAP-BCSP with the following properties:*

- *$|\widehat{C}| \leq \frac{k^4}{\epsilon}$;*
- *if I is satisfiable, then $\widehat{I}$ is satisfiable;*
- *if $\widehat{I}$ admits an evaluation that satisfies at least $(1-\frac{\epsilon}{2})|\widehat{C}|$ constraints, then I admits an evaluation that satisfies at least $(1-\epsilon)|C|$ constraints.*

Proof. Let $I' = (X, \mathcal{D}, C')$ be an instance of ϵ-GAP-BCSP with $k = |X|$. Suppose that $|C| > \frac{k^4}{\epsilon}$, else we are already done. By Lemma 4.1, we obtain in polynomial time an instance $I = (X, \mathcal{D}, C)$ such that for all $i, j \in [k]$, there is at most one distinct constraint in $C_{\{i,j\}}$, and $|C| = |C'|$. Now, we construct in polynomial time an instance $\widehat{I} = (X, D, \widehat{C})$ of $\frac{\epsilon}{2}$-GAP-BCSP as follows. Denote $K = \binom{k}{2}$ and $T = \frac{\epsilon}{2}|C| (> \frac{k^4}{2})$. Let $\widehat{C}_{\{i,j\}}$ be a multiset of $\widehat{s}_{\{i,j\}} := \lfloor \frac{K}{T} s_{\{i,j\}} \rfloor (\leq s_{\{i,j\}})$ occurrences of the distinct constraint in $C_{\{i,j\}}$. Define $\widehat{C} = \bigcup_{i,j\in[k],i<j} \widehat{C}_{\{i,j\}}$ (note that $\widehat{C}$ is a multiset).

Observe that $|\widehat{C}| = \sum_{i,j\in[k],i<j} \widehat{s}_{\{i,j\}} = \sum_{i,j\in[k],i<j} \lfloor \frac{K}{T} s_{\{i,j\}} \rfloor \leq \sum_{i,j\in[k],i<j} \frac{K}{\frac{\epsilon}{2}|C|} s_{\{i,j\}}$. Since for all $i, j \in [k]$, $s_{\{i,j\}} \leq |C|$, we have that $|\widehat{C}| \leq \sum_{i,j\in[k],i<j} \frac{K}{\frac{\epsilon}{2}} = \frac{2K^2}{\epsilon} \leq \frac{k^4}{\epsilon}$ as required. Moreover, because $\widehat{C} \subseteq C$ and by Lemma 4.1, we have that if I' is satisfiable, then I is satisfiable, and therefore $\widehat{I}$ is satisfiable as well.

Now, suppose that $\widehat{I}$ admits an evaluation ψ that satisfies at least $(1-\frac{\epsilon}{2})|\widehat{C}|$ constraints. By Lemma 4.1, to show that I' admits an evaluation that satisfies at least $(1-\epsilon)|C'|$ constraints, it suffices to show that I admits an evaluation that satisfies at least $(1-\epsilon)|C|$ constraints. Let A be the collection of sets $\{i,j\}$, $i, j \in [k]$ where $i < j$, such that ψ satisfies the distinct constraint in $C_{\{i,j\}}$. Then, the number of constraints in C that ψ satisfies is $\sum_{\{i,j\}\in A} s_{\{i,j\}} \geq \frac{T}{K} \sum_{\{i,j\}\in A} \widehat{s}_{\{i,j\}} \geq \frac{T}{K}(1-\frac{\epsilon}{2})|\widehat{C}| \geq \frac{T}{K}(1-\frac{\epsilon}{2}) \sum_{i,j\in[k],i<j} \widehat{s}_{\{i,j\}} \geq \frac{T}{K}(1-\frac{\epsilon}{2}) \sum_{i,j\in[k],i<j} (\frac{K}{T} s_{\{i,j\}} - 1) = \frac{T}{K}(1-\frac{\epsilon}{2})(\frac{K}{T}|C| - K) = (1-\frac{\epsilon}{2})|C| - (1-\frac{\epsilon}{2})T \leq (1-\frac{\epsilon}{2})|C| - T = (1-\frac{\epsilon}{2})|C| - \frac{\epsilon}{2}|C| = (1-\epsilon)|C|$. This concludes the proof. ∎

In light of Lemma 4.2, throughout the remainder of this section we assume that $|C| \leq \frac{k^4}{\epsilon}$.

Recall that for a fixed integer d, the ϵ-GAP-BCSP$_d$ is defined as the special case of ϵ-GAP-BCSP where every variable is present in at most d constraints, and for all $i,j \in [k]$, $s_{\{i,j\}} \leq 1$. Towards the proof of the hardness of ϵ-GAP-BCSP$_3$, we first consider ϵ-GAP-BCSP$_4$. For this proof, we need to recall the notion of an expander.

Definition 4.1. *Given $n, d \in \mathbb{N}$ and $0 \leq \gamma \leq 1$, an (n,d,γ)-*expander *is an undirected d-regular graph G on n vertices such that for every set $S \subseteq V(G)$ of size at most $\frac{1}{2}|V(G)|$, the number of edges with one endpoint in S and the other endpoint in $V(G) \setminus S$ is at least $\gamma \cdot d \cdot |S|$.*

For the sake of brevity, given $n_1, n_2 \in \mathbb{N}$, we refer to any (n,d,γ)-expander where $n_1 \leq n \leq n_2$ as an $([n_1,n_2],d,\gamma)$-*expander*. We would also need to rely on the following result.

Proposition 4.1. ([2]) *There exist $\gamma > 0$ and $\ell \in \mathbb{N}$ such that for all $s \in \mathbb{N}$, an $([s,\ell s],3,\gamma)$-expander can be constructed in polynomial time.*

Lemma 4.3. *Assuming* PIH*, there exists an $\epsilon > 0$ such that ϵ-GAP-BCSP$_4$ is* W[1]*-hard.*

Proof. To prove that the lemma is correct, we present a reduction from ϵ-GAP-BCSP to δ-GAP-BCSP$_4$ where $\delta = \dfrac{\epsilon}{4\ell(1+\frac{1}{3\gamma})}$. Here, γ and ℓ are the fixed constants stated in Proposition 4.1. For this purpose, let $I = (X, \mathcal{D}, C)$ be an instance of ϵ-GAP-BCSP. Then, we construct an instance $\widehat{I} = (\widehat{X}, \widehat{\mathcal{D}}, \widehat{C})$ of δ-GAP-BCSP$_4$ as follows. First, for all $i \in [k]$, apply Proposition 4.1 to construct an $([s_i, \ell s_i],3,\gamma)$-expander G_i (recall that $s_i = |C_i|$, the number of constraints where x_i occurs), and denote $n_i = |V(G_i)|$. Now, for all $i \in [k]$, define $\widehat{X}^i = \{\widehat{x}^i_1, \widehat{x}^1_2, \ldots, \widehat{x}^i_{n_i}\}$, and let $f^i : \widehat{X}^i \rightarrow V(G_i)$ be an arbitrarily chosen bijective function. Accordingly, set $\widehat{X} = \bigcup_{i=1}^k \widehat{X}^i$. Recall that we assume that $|C| \leq \frac{k^4}{\epsilon}$. Moreover, since the constraints are binary, we have that $\displaystyle\sum_{i=1}^k s_i = 2|C|$. Therefore, $|\widehat{X}| = \displaystyle\sum_{i=1}^k n_i \leq \ell \sum_{i=1}^k s_i = 2\ell|C| \leq 2\ell\frac{k^4}{\epsilon}$. That is, the number of variables in the new instance is bounded by a function of k.

We proceed to define $\widehat{D}$ by letting the domain of every $\widehat{x}^i_j$, where $i \in [k]$ and $j \in [n_i]$, be $\widehat{D}^i_j = \{\widehat{d}^i_j : d \in D_i\}$. Finally, let us define $\widehat{C}$ as follows. For all $i \in [k]$, let $g^i : C_i \rightarrow X^i$ be an arbitrarily chosen injective function. Then, for all $c = ((x_i,x_j),R) \in C$, denote $\widehat{c} = ((x^i_p, x^j_q), \widehat{R} = \{(\widehat{d}^i_p, \widehat{d'}^j_q) : (d,d') \in R\})$ where $x^i_p = g^i(x_i)$ and $x^j_q = g^j(x_j)$. Define $C^\star = \{\widehat{c} : c \in C\}$. We now define a set of equality constraints on the set of 'copies' of each variable. Define $C_= = \{((x^i_p, x^i_q), \{(d^i_p, d^i_q) : d \in D_i\}) : i \in [k], p, q \in [n_i], (p,q) \in E(G_i)\}$. Finally, set $\widehat{C} = C^\star \cup C_=$. Since for all $i \in [k]$, the graph G_i is 3-regular, we have that every variable in $\widehat{X}^i$ occurs in at most three constraints in $C_=$. Moreover, since for all $i \in [k]$ the function g^i is injective, we have that every variable in $\widehat{X}^i$ occurs in at most one constraint in $C^\star$. Thus, every variable in $\widehat{X}$ occurs in at most four constraints in total. Thus, since every constraint is binary, we also have that $|\widehat{C}| \leq 2|\widehat{X}|$. Since we have already shown that $|\widehat{X}| \leq 2\ell|C|$, we derive that $|\widehat{C}| \leq 4\ell|C|$.

To prove that the reduction is correct, we argue that if I is satisfiable (admits an evaluation satisfying all constraints) then so is $\widehat{I}$ and conversely, if $\widehat{I}$ admits an evaluation that satisfies at least $(1-\delta)|\widehat{C}|$ constraints, then I admits an evaluation that satisfies at least $(1-\epsilon)|C|$ constraints.

Suppose that I admits an evaluation ψ that satisfies all of the constraints. Then, we have that $\widehat{I}$ also admits an evaluation $\widehat{\psi}$ that satisfies all of the constraints. Indeed, we simply define $\widehat{\psi}$ by setting $\widehat{\psi}(x^i_j) = d^i_j$, where $d = \psi(x_i)$, for all $i \in [k]$ and $j \in [n_i]$.

Next, suppose that $\widehat{I}$ admits an evaluation $\widehat{\psi}$ that satisfies at least $(1-\delta)|\widehat{C}|$ constraints. We define an evaluation ψ for I as follows. For all $i \in [k]$ and $d \in D_i$, define $X^i(d) = \{x^i_j \in X^i : \widehat{\psi}(x^i_j) = d^i_j\}$. Now, for all $i \in [k]$, let $\widetilde{d^i}$ be a value in D_i that among all values in D_i, maximizes $|X^i(d)|$ (if there is more than one choice, choose one arbitrarily). Moreover, for all $i \in [k]$, denote $Y^i = X^i \setminus X^i(\widetilde{d^i})$. Then, for all $i \in [k]$, we set $\psi(x_i) = \widetilde{d^i}$. We claim that ψ satisfies at least $(1-\epsilon)|C|$ of the constraints in C. To show this, we first note that all $\widehat{c} = ((x^i_p, x^i_q), R)$ such that either both $\widehat{\psi}(x^i_p) = \widetilde{d^i}$ and $\widehat{\psi}(x^i_q) \neq \widetilde{d^i}$ or both $\widehat{\psi}(x^i_p) \neq \widetilde{d^i}$ and $\widehat{\psi}(x^i_q) = \widetilde{d^i}$, a unique constraint in $C_=$ is violated by $\widehat{\psi}$. Since for all $i \in [k]$, G_i is an $([s_i, \ell s_i],3,\gamma)$-expander, we derive that at least $\displaystyle\sum_{i=1}^k \gamma \cdot 3 \cdot |Y^i| = 3\gamma \sum_{i=1}^k |Y^i|$ constraints in $C_=$ are violated by $\widehat{\psi}$. Thus, $3\gamma \displaystyle\sum_{i=1}^k |Y_i| < \delta|\widehat{C}|$, which implies that $\displaystyle\sum_{i=1}^k |Y_i| \leq \frac{\delta}{3\gamma}|\widehat{C}|$. Let us now denote by C_Y the set of all constraints $c = ((x_i,x_j),R) \in C$ such that $g^i(c) \in Y^i$ or $g^j(c) \in Y_j$. Then, since for all $i \in [k]$, g^i is an injective function, we have that $|C_Y| \leq \displaystyle\sum_{i=1}^k |Y_i|$, and thus we derive that $|C_Y| \leq \frac{\delta}{3\gamma}|\widehat{C}|$. Notice that for all $c \in C \setminus C_Y$ that is violated by ψ, there is a unique constraint in $C^\star$ that is also violated by $\widehat{\psi}$. Thus, the number of constraints in $C \setminus C_Y$ that are violated by ψ is smaller than $\delta|\widehat{C}|$. Overall, we conclude that ψ violates less than $\delta|\widehat{C}| + |C_Y| \leq (1 + \frac{1}{3\gamma})\delta|\widehat{C}| \leq 4\ell(1 + \frac{1}{3\gamma})\delta|C|$ constraints in C. Since $\delta = \frac{\epsilon}{4\ell(1+\frac{1}{3\gamma})}$, we conclude that ψ violates less than $\epsilon|C|$ constraints in C. This concludes the proof of the lemma. $\square$

We now turn to prove the hardness of ϵ-GAP-BCSP$_3$.

LEMMA 4.4. *Assuming* PIH, *there exists an $\epsilon > 0$ such that ϵ-*GAP*-BCSP$_3$ is* W[1]*-hard.*

Proof. By Lemma 4.3, we have that there exists an $\epsilon > 0$ such that ϵ-GAP-BCSP$_4$ is W[1]-hard. To prove that the lemma is correct, we present a reduction from ϵ-GAP-BCSP$_4$ to δ-GAP-BCSP$_3$ where $\delta = \epsilon/15$. For this purpose, let $I = (X, \mathcal{D}, C)$ be an instance of ϵ-GAP-BCSP$_4$. Then, we construct an instance $\widehat{I} = (\widehat{X}, \widehat{\mathcal{D}}, \widehat{C})$ of δ-GAP-BCSP$_3$ as follows. First, define $\widehat{X} = X \cup X'$ where $X' = \{x'_1, x'_2, \ldots, x'_k\}$. Hence, $|\widehat{X}| \leq 2k$. Now, let us define $\widehat{\mathcal{D}}$. For all $i \in [k]$, the domain of x_i is defined as $D_i \in \mathcal{D}$, and the domain of x'_i is defined as $D'_i = \{d' : d \in D_i\}$. Now, for all $i \in [k]$, let (A_i, B_i) be a partition of C_i such that $|A_i|, |B_i| \leq 2$. Note that the existence of such a partition follows from the fact that $|C_i| \leq 4$. Then, for all $c = ((x_i, x_j), R) \in C$, denote $\widehat{c} = ((x_i, x_j), R)$ if $((x_i, x_j), R) \in A_i \cap A_j$, $\widehat{c} = ((x_i, x'_j), R' = \{(p, q') : (p, q) \in R\})$ if $((x_i, x_j), R) \in A_i \cap B_j$, $\widehat{c} = ((x'_i, x_j), R' = \{(p', q) : (p, q) \in R\})$ if $((x_i, x_j), R) \in B_i \cap A_j$ and $\widehat{c} = ((x'_i, x'_j), R' = \{(p', q') : (p, q) \in R\})$ otherwise. Define $C^\star = \{\widehat{c} : c \in C\}$. We now define a set of equality constraints on each pair of copies of the variables. Define $C_= = \{((x_i, x'_i), \{(d, d') : d \in D_i\}) : i \in [k]\}$. Finally, set $\widehat{C} = C^\star \cup C_=$. Clearly, every variable in $\widehat{X}$ occurs in at most three constraints in $\widehat{C}$, and the construction can be performed in polynomial time.

To prove that the reduction is correct, we argue that if I is satisfiable then so is $\widehat{I}$ and conversely, if $\widehat{I}$ admits an evaluation that satisfies at least $(1 - \delta)|\widehat{C}|$ constraints, then I admits an evaluation that satisfies at least $(1 - \epsilon)|C|$ constraints.

Suppose that I admits an evaluation ψ that satisfies all of the constraints. Then, we have that $\widehat{I}$ also admits an evaluation $\widehat{\psi}$ that satisfies all of the constraints. Indeed, we simply define $\widehat{\psi}$ by setting $\widehat{\psi}(x_i) = \psi(x_i)$ and $\widehat{\psi}(x'_i) = \psi(x_i)'$ for all $i \in [k]$.

Next, suppose that $\widehat{I}$ admits an evaluation $\widehat{\psi}$ that satisfies at least $(1 - \delta)|\widehat{C}|$ constraints. We define an evaluation ψ for I as follows. For all $i \in [k]$, we set $\psi(x_i) = \widehat{\psi}(x_i)$. We claim that ψ satisfies at least $(1 - \epsilon)|C|$ of the constraints in C. To show this, we denote $Y = \{x_i \in X : \widehat{\psi}(x_i) \neq \widehat{\psi}(x'_i)\}$. Since $\widehat{\psi}$ violates less than $\delta|\widehat{C}|$ constraints, we have that $|Y| < \delta|\widehat{C}|$. Let C_Y denote the subset of constraints of C where at least one variable of Y occurs. Note that since every variable in X occurs in at most four constraints in C, we have that $|C_Y| \leq 4|Y| \leq 4\delta|\widehat{C}|$. Moreover, note that for every constraint $\widehat{c} \in C^\star \setminus C_Y$ that is satisfied by $\widehat{\psi}$ is also satisfied by ψ. Since $\widehat{\psi}$ violates less than $\delta|\widehat{C}|$ constraints in total, we have that $\widehat{\psi}$ also violates less than $\delta|\widehat{C}|$ constraints from $C^\star \setminus C_Y$. Thus, we have that ψ violates less than $5\delta|\widehat{C}|$ constraints from C. Since every variable occurs in at least one constraint, we have that $|\widehat{C}| = |C^\star| + |C_=| = |C| + |X| \leq 3|C|$. We thus conclude that violates less than $15\delta|C| = \epsilon|C|$ constraints from C. This concludes the proof of the lemma. $\square$

4.4 Proof of Theorem 1.3 We now translate Hypothesis 1 in terms of PSI. For this purpose, we define the promise problem ϵ-GAP-PSI as PSI where the input instance is promised to either be a yes-instance, or has the property that for every subgraph G' of G with at least $(1 - \epsilon)|E(G)|$ edges, there does not exist a colorful mapping of G' into H. It is straightforward to see that if ϵ-GAP-BCSP is W[1]-hard, then ϵ-GAP-PSI is W[1]-hard as well.

LEMMA 4.5. *Assuming* PIH, *there exists an $\epsilon > 0$ such that ϵ-*GAP*-PSI is* W[1]*-hard.*

We remark that it is also straightforward to see that if ϵ-GAP-PSI is W[1]-hard, then ϵ-GAP-BCSP$_3$ is W[1]-hard, and hence ϵ-GAP-BCSP is W[1]-hard as well.

Finally, we ready to prove the correctness of Theorem 1.3.

Proof. [Proof of Theorem 1.3] Suppose that there exists an $\epsilon > 0$ for which there does not exist an FPT algorithm for ϵ-GAP-PSI. Let $\delta = \delta(\epsilon)$ be defined according to Lemma 3.15. We claim that A-DOCT does not admit a $(1 + \delta)$-approximation algorithm that runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ for any function f. By Lemma 4.5 and Corollary 3.1, we would thus conclude the correctness of Theorem 1.3. Suppose, by way of contradiction, that our claim is false. Then, let $\mathcal{B}$ be a $(1 + \delta)$-approximation algorithm for A-DOCT that runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f. We define an algorithm, Algorithm $\mathcal{A}$ as follows. Given an instance

(H, G, col) of PSI, it constructs the instance (D, k, ℓ, w) of A-DOCT as described in Section 3.5, and calls Algorithm $\mathcal{B}$ with (D, k, ℓ, w) as input. Then, if $\mathcal{B}$ outputs No, then $\mathcal{A}$ outputs No, and otherwise $\mathcal{A}$ outputs Yes. By Observations 3.13 and 3.14, Algorithm $\mathcal{A}$ runs in time $f(|E(H)|) \cdot |I|^{\mathcal{O}(1)}$ where $|I|$ is the size of the input instance. On the one hand, by Lemma 3.14, if Algorithm $\mathcal{A}$ is given as input a yes-instance of PSI, then it constructs a yes-instance of A-DOCT. Next, since $\mathcal{B}$ is a $(1+\delta)$-approximation algorithm for A-DOCT, Algorithm $\mathcal{A}$ outputs Yes. On the other hand, suppose that Algorithm $\mathcal{A}$ is given as input an instance (H, G, col) of PSI for which there does not exist a subgraph G' of G with at least $(1-\epsilon)|E(G)|$ edges such that there exists a colorful mapping of G' into H. By Lemma 3.15, Algorithm $\mathcal{A}$ constructs an instance (D, k, ℓ, w) of A-DOCT which does not admit a $(1+\delta)$-approximate solution. Then, since $\mathcal{B}$ is a $(1+\delta)$-approximation algorithm for A-DOCT, Algorithm $\mathcal{A}$ outputs No. We have thus reached a contradiction to the choice of ϵ. This concludes the proof of Theorem 1.3. □

5 Conclusions

Our results on Directed Odd Cycle Transversal raise a few natural questions. The first question is whether one can improve on the approximation factor of 2 in Theorem 1.2 or strengthen the inapproximability result in Theorem 1.3 to show that even such an improvement is unlikely. Secondly, although Theorem 1.1 implies that DOCT is unlikely to have a kernel of *any size*, our FPT-approximation algorithm implies that DOCT does have a 2-approximate kernel of exponential size (see Proposition 3.2, [32]). Therefore, an exciting new challenge related to DOCT is to determine whether it has a c-approximate kernel of *polynomial size* for some constant c and if so, to find the smallest such constant. Note that Theorem 1.3 also rules out a $(1+\epsilon)$-approximate kernel (for some $\epsilon > 0$) of *any* size for DOCT. We conclude by pointing out that the parameterized complexity of the Directed Multicut problem where the number of terminal pairs is 3, remains open. As was the case for DOCT, it is quite likely that an FPT algorithm or a W-hardness proof for this problem would require new insights into the structure of directed cuts.

Acknowledgements

The authors would like to thank Michał Włodarczyk for enlightening discussions on the DOCT problem.

References

[1] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev, *$O(\sqrt{(\log n)})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005, 2005, pp. 573–581. 1

[2] N. Alon, O. Schwartz, and A. Shapira, *An elementary construction of constant-degree expanders*, Combinatorics, Probability & Computing, 17 (2008), pp. 319–327. 18

[3] J. Bang-Jensen and G. Gutin, *Digraphs - theory, algorithms and applications*, Springer, 2002. 5

[4] A. Bhattacharyya, S. Ghoshal, S. Karthik C., and P. Manurangsi, *Parameterized Intractability of Even Set and Shortest Vector Problem from Gap-ETH*, ArXiv e-prints, (2018). 2

[5] A. Bhattacharyya, S. Ghoshal, Karthik C. S., and P. Manurangsi, *Parameterized intractability of even set and shortest vector problem from gap-eth*, in 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic, 2018, pp. 17:1–17:15. 2

[6] P. Chalermsook, M. Cygan, G. Kortsarz, B. Laekhanukit, P. Manurangsi, D. Nanongkai, and L. Trevisan, *From gap-eth to fpt-inapproximability: Clique, dominating set, and more*, in 58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017, 2017, pp. 743–754. 2, 16

[7] J. Chen, Y. Liu, and S. Lu, *An improved parameterized algorithm for the minimum node multiway cut problem*, Algorithmica, 55 (2009), pp. 1–13. 3

[8] J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon, *A fixed-parameter algorithm for the directed feedback vertex set problem*, J. ACM, 55 (2008). 1, 3, 4

[9] R. Chitnis, *Directed Graphs: Fixed-Parameter Tractability & Beyond*, PhD thesis, University of Maryland, 2014. 1

[10] R. Chitnis, A. E. Feldmann, and P. Manurangsi, *Parameterized approximation algorithms for bidirected steiner network problems*, in 26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland, 2018, pp. 20:1–20:16. 2

[11] R. Chitnis and M. T. Hajiaghayi, *Shadowless solutions for fixed-parameter tractability of directed graphs*, in Encyclopedia of Algorithms, 2016, pp. 1963–1966. 1, 4

[12] R. H. Chitnis, M. Cygan, M. T. Hajiaghayi, and D. Marx, *Directed subset feedback vertex set is fixed-parameter tractable*, ACM Transactions on Algorithms, 11 (2015), p. 28. 1, 3, 4

[13] R. H. Chitnis, M. Hajiaghayi, and D. Marx, *Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset*, SIAM J. Comput., 42 (2013), pp. 1674–1696. 3, 4

[14] H. Choi, K. Nakajima, and C. S. Rim, *Graph*

bipartization and via minimization, SIAM J. Discrete Math., 2 (1989), pp. 38–47. 1

[15] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015. 1, 3, 4, 5

[16] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk, *Subset feedback vertex set is fixed-parameter tractable*, SIAM J. Discrete Math., 27 (2013), pp. 290–309. 3

[17] E. D. Demaine, G. Gutin, D. Marx, and U. Stege, *07281 open problems – Structure Theory and FPT algorithmcs for graphs, digraphs and hypergraphs*, in Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs, 08.07. - 13.07.2007, 2007. 1

[18] R. Diestel, *Graph Theory*, Springer-Verlag, Heidelberg, 4th ed., 2010. 5

[19] I. Dinur, *Mildly exponential reduction from gap 3sat to polynomial-gap label-cover*, Electronic Colloquium on Computational Complexity (ECCC), 23 (2016), p. 128. 2, 16

[20] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity*, Texts in Computer Science, Springer, 2013. 1, 4, 5

[21] G. Even, J. Naor, B. Schieber, and M. Sudan, *Approximating minimum feedback sets and multicuts in directed graphs*, Algorithmica, 20 (1998), pp. 151–174. 1

[22] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006. 5

[23] V. Guruswami and E. Lee, *Simple proof of hardness of feedback vertex set*, Theory of Computing, 12 (2016), pp. 1–11. 1

[24] R. Impagliazzo and R. Paturi, *On the complexity of k-sat*, J. Comput. Syst. Sci., 62 (2001), pp. 367–375. 2

[25] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?*, J. Comput. Syst. Sci., 63 (2001), pp. 512–530. 2

[26] Y. Iwata, K. Oka, and Y. Yoshida, *Linear-time FPT algorithms via network flow*, in SODA, 2014, pp. 1749–1761. 4

[27] R. M. Karp, *Reducibility among combinatorial problems*, in Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York., 1972, pp. 85–103. 1

[28] S. Khot and N. K. Vishnoi, *The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into ℓ_1*, J. ACM, 62 (2015), pp. 8:1–8:39. 1, 2

[29] S. Kratsch, M. Pilipczuk, M. Pilipczuk, and M. Wahlström, *Fixed-parameter tractability of multicut in directed acyclic graphs*, SIAM J. Discrete Math., 29 (2015), pp. 122–144. 3

[30] D. Lokshtanov and D. Marx, *Clustering with local restrictions*, Inf. Comput., 222 (2013), pp. 278–292. 3

[31] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh, *Faster parameterized algorithms using linear programming*, ACM Trans. Algorithms, 11 (2014), pp. 15:1–15:31. 4

[32] D. Lokshtanov, F. Panolan, M. S. Ramanujan, and S. Saurabh, *Lossy kernelization*, in Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017, 2017, pp. 224–237. 20

[33] D. Lokshtanov and M. S. Ramanujan, *Parameterized tractability of multiway cut with parity constraints*, in Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I, 2012, pp. 750–761. 3

[34] D. Lokshtanov, M. S. Ramanujan, and S. Saurabh, *A linear-time parameterized algorithm for node unique label cover*, in 25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria, 2017, pp. 57:1–57:15. 3

[35] ———, *Linear time parameterized algorithms for subset feedback vertex set*, ACM Trans. Algorithms, 14 (2018), pp. 7:1–7:37. 3

[36] P. Manurangsi and P. Raghavendra, *A birthday repetition theorem and complexity of approximating dense csps*, in 44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland, vol. 80 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 78:1–78:15. 2, 16

[37] D. Marx, *Parameterized graph separation problems*, Theoret. Comput. Sci., 351 (2006), pp. 394–406. 3

[38] D. Marx, *Can you beat treewidth?*, Theory of Computing, 6 (2010), pp. 85–112. 2, 6

[39] D. Marx, *What's next? future directions in parameterized complexity*, in The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday, vol. 7370 of Lecture Notes in Computer Science, 2012, pp. 469–496. 1

[40] ———, *Some open problems in parameterized complexity (slides, dagstuhl seminar 17041)* http://www.cs.bme.hu/ dmarx/papers/marx-dagstuhl2017-open.pdf, 2017. 1

[41] D. Marx and I. Razgon, *Fixed-parameter tractability of multicut parameterized by the size of the cutset*, SIAM J. Comput., 43 (2014), pp. 355–388. 3, 4

[42] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*, vol. 31 of Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, Oxford, 2006. 5

[43] M. Pilipczuk and M. Wahlström, *Directed multicut is* W[1]*-hard, even for four terminal pairs*, TOCT, 10 (2018), pp. 13:1–13:18. 3, 6

[44] M. S. Ramanujan and S. Saurabh, *Linear-time parameterized algorithms via skew-symmetric multicuts*, ACM Trans. Algorithms, 13 (2017), pp. 46:1–46:25. 4

[45] B. A. Reed, K. Smith, and A. Vetta, *Finding odd*

cycle transversals, Oper. Res. Lett., 32 (2004), pp. 299–301. 1, 4