


Parameterized Complexity of MinCSP over the Point Algebra

George Osipov  

Linköping University, Sweden

Marcin Pilipczuk  

University of Warsaw, Warsaw, Poland

Magnus Wahlström  

Royal Holloway, University of London, TW20 0EX, UK

Abstract

The input in the MINIMUM-COST CONSTRAINT SATISFACTION PROBLEM (MINCSP) over the Point Algebra contains a set of variables, a collection of constraints of the form $x < y$, $x = y$, $x \leq y$ and $x \neq y$, and a budget k . The goal is to check whether it is possible to assign rational values to the variables while breaking constraints of total cost at most k . This problem generalizes several prominent graph separation and transversal problems:

- MINCSP($<$) is equivalent to DIRECTED FEEDBACK ARC SET,
- MINCSP($<, \leq$) is equivalent to DIRECTED SUBSET FEEDBACK ARC SET,
- MINCSP($=, \neq$) is equivalent to EDGE MULTICUT, and
- MINCSP(\leq, \neq) is equivalent to DIRECTED SYMMETRIC MULTICUT.

Apart from trivial cases, MINCSP(Γ) for $\Gamma \subseteq \{<, =, \leq, \neq\}$ is NP-hard even to approximate within any constant factor under the Unique Games Conjecture. Hence, we study parameterized complexity of this problem under a natural parameterization by the solution cost k . We obtain a complete classification: if $\Gamma \subseteq \{<, =, \leq, \neq\}$ contains both \leq and \neq , then MINCSP(Γ) is W[1]-hard, otherwise it is fixed-parameter tractable. For the positive cases, we solve MINCSP($<, =, \neq$), generalizing the FPT results for DIRECTED FEEDBACK ARC SET and EDGE MULTICUT as well as their weighted versions. Our algorithm works by reducing the problem into a BOOLEAN MINCSP, which is in turn solved by flow augmentation. For the lower bounds, we prove that DIRECTED SYMMETRIC MULTICUT is W[1]-hard, solving an open problem.

2012 ACM Subject Classification Theory of computation \rightarrow Fixed parameter tractability

Keywords and phrases parameterized complexity, constraint satisfaction, point algebra, multicut, feedback arc set

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *George Osipov*: George was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Marcin Pilipczuk: During this research Marcin was part of BARC, supported by the VILLUM Foundation grant 16582.

Acknowledgements We thank the anonymous reviewers whose detailed feedback significantly improved presentation of the paper. This work was carried out during the Copenhagen Summer of Counting & Algebraic Complexity, funded by research grants from VILLUM FONDEN (Young Investigator Grant 53093) and the European Union (ERC, CountHom, 101077083). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.



© George Osipov, Marcin Pilipczuk, Magnus Wahlström;
licensed under Creative Commons License CC-BY 4.0

42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The study of graph transversal and separation problems parameterized by solution size is a central research direction in parameterized complexity. It is usually easy to obtain algorithms for such problems running in $n^{O(k)}$ time, where n is the size of the graph, by enumerating all possible solutions of size at most k . However, obtaining or ruling out *fixed-parameter tractable* (FPT) algorithms, i.e. those running in $f(k) \cdot n^{O(1)}$ time for some computable function f that depends solely on the parameter, is a more challenging task. Successful examples include the $3^k \cdot n^{O(1)}$ algorithm of Reed, Smith and Vetta [26] for ODD CYCLE TRANSVERSAL, which introduced *iterative compression* – a technique that has since become a standard opening in FPT algorithms (see e.g. [10, Chapter 4]). The algorithms of Marx [22] for MULTIWAY CUT and Marx and Razgon [23] for MULTICUT have introduced, respectively, *important separators* and *shadow removal* to the toolbox of parameterized algorithms; these techniques have also found numerous applications in the field. Another notable example is the algorithm of Chen et al. [8] for DIRECTED FEEDBACK ARC SET (DFAS).

The MINIMUM-COST CONSTRAINT SATISFACTION PROBLEM (MINCSP) provides a unifying framework for modeling optimization problems, including transversal and separator problems in graphs. The input to a MINCSP is a collection of constraints applied to a set of variables, and the goal is to find a solution of minimum cost, i.e. an assignment of values from a fixed domain to the variables that breaks the minimum number of constraints. One can cast many problems as MINCSPs by restricting the *constraint language*, i.e. the types of allowed constraints. More formally, let Γ be a set of finitary relations on a fixed domain D . Then $\text{MINCSP}(\Gamma)$ is the problem with constraints of the form $R(x_1, \dots, x_r)$, where $R \in \Gamma$ is a relation of arity r and $(x_1, \dots, x_r) \in V^r$ is a tuple of variables from V . An assignment $\alpha : V \rightarrow D$ satisfies the constraint $R(x_1, \dots, x_r)$ if $(\alpha(x_1), \dots, \alpha(x_r)) \in R$.

For example, $\text{MINCSP}(<)$ on domain \mathbb{Q} is equivalent to the DIRECTED FEEDBACK ARC SET (DFAS) problem that asks to find a minimum-size set of arcs in a directed graph meeting every cycle. In other words, deleting this set of arcs makes the graph acyclic. The reductions in both directions are straightforward: arcs uv in the graph translate into constraints $u < v$, and vice versa. Clearly, to make a set of $<$ -constraints satisfiable, it is necessary and sufficient to delete all cycles of $<$ -constraints. Another example is $\text{MINCSP}(=, \neq)$ on domain \mathbb{N} (or \mathbb{Q}). This problem is essentially equivalent to the EDGE MULTICUT problem defined as follows: given an undirected graph and vertex pairs $\{s_1, t_1\}, \dots, \{s_m, t_m\}$ called *cut requests*, find a minimum edge set that separates s_i and t_i for all i . Without loss of generality, we may also assume that cut requests are deletable at unit cost (see e.g. [23]). This makes the reductions rather simple: edges uv of the graph translate into constraints $u = v$, while cut requests $\{s_i, t_i\}$ translate into constraints $s_i \neq t_i$, and vice versa. To make a set of such constraints satisfiable, it is necessary and sufficient to ensure that, for every constraint $s \neq t$, there is no $=$ -path connecting s and t .

As evident from the examples above, many important MINCSPs are NP-hard, and to cope with this, it is natural to parameterize them by solution cost. This line of work has recently gained momentum, notably after Kim et al. introduced *flow augmentation* [17, 18] and successfully used it to resolve the complexity of $\text{MINCSP}(\Gamma)$ for every Boolean constraint language Γ , i.e. Γ with domain $\{0, 1\}$ [19]. Previously, Bonnet et al. [6, 5] classified the complexity of fpt-approximating BOOLEAN MINCSPs within a constant factor. Osipov and Wahlström [24] resolved the parameterized complexity of exactly solving and constant-factor approximation of EQUALITY MINCSPs, i.e. $\text{MINCSP}(\Gamma)$ for all Γ on domain \mathbb{N} with relations first-order definable using predicate $=$.

In this paper we consider MINCSP over subsets of *Point Algebra* [30, 29], i.e. $\text{MINCSP}(\Gamma)$ for constraint languages $\Gamma \subseteq \{<, \leq, =, \neq\}$ on domain \mathbb{Q} . Our motivation is twofold:

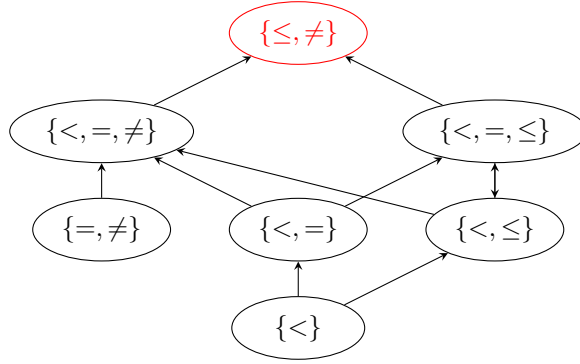
On the CSP side, constraint language $\{<, \leq, =, \neq\}$ arguably contains the most basic relations over \mathbb{Q} and understanding the complexity of $\text{MINCSP}(\Gamma)$ for all $\Gamma \subseteq \{<, \leq, =, \neq\}$ has been identified [20] as a necessary stepping stone towards broader classification projects, e.g. for all temporal [4] and interval constraint languages [1, 21, 2]. Additionally, Point Algebra is a prominent temporal reasoning formalism in artificial intelligence, and $\text{MINCSP}(<, \leq, =, \neq)$ provides a natural way of dealing with inconsistencies in knowledge bases encoded using this language. The problem being NP-hard even to approximate within any constant factor (under the Unique Games Conjecture [16, 28, 14]) motivates studying parameterized algorithms. For the study of exact exponential-time algorithms and polynomial-time approximation, see [15].

On the FPT side, the set of problems $\text{MINCSP}(\Gamma)$ for $\Gamma \subseteq \{<, \leq, =, \neq\}$ contains two classical NP-hard graph separation and transversal problems – DFAS and EDGE MULTICUT – as well as several robust generalizations and variants. Studying some of these problems has played an important role in the development of the parameterized algorithms, and looking at a broader unifying class allows exploring the power and limits of existing techniques. Among the problems within our scope are $\text{MINCSP}(<, \leq)$, which is equivalent to DIRECTED SUBSET FEEDBACK ARC SET (SUBSET-DFAS), and $\text{MINCSP}(\leq, \neq)$, which is equivalent to DIRECTED SYMMETRIC MULTICUT (DSMC).

SUBSET-DFAS is a variant of DFAS in which the input graph comes with a subset of special arcs, and the goal is to find a minimum transversal for the family of cycles that contain at least one special arc (special arcs translate into $<$ -constraints while other arcs translate into \leq -constraints, and vice versa). Parameterized complexity of this problem was resolved by Chitnis et al. [9] by generalizing the shadow removal technique of [23] to directed graphs. Recently, Kim et al. [20] gave a flow-augmentation based algorithm for this problem that can also handle the arc-weighted version where the weight budget is not part of the parameter.

In DSMC, we are given a directed graph G and cut requests $\{s_1, t_1\}, \dots, \{s_m, t_m\}$, and the goal is to find a minimum subset of arcs that separates s_i and t_i for all i into distinct strongly connected components. Again, without loss of generality, we may assume that a cut request can be ignored at unit cost. The translation into $\text{MINCSP}(\leq, \neq)$ is as follows: an arc uv in the graph becomes a constraint $u \leq v$ and a cut request $\{s_i, t_i\}$ becomes a constraint $s_i \neq t_i$, and vice versa. Note that a set of \leq -constraints is always satisfiable, and adding a constraint $s_i \neq t_i$ makes it unsatisfiable if and only if s_i reaches t_i and t_i reaches s_i by directed paths of \leq -constraints. Using the CSP language, it is easy to see that $\text{MINCSP}(\leq, \neq)$ generalizes both SUBSET-DFAS and EDGE MULTICUT: one obtains a cost-preserving reduction from $\text{MINCSP}(<, \leq)$ to $\text{MINCSP}(\leq, \neq)$ by replacing every constraint of the form $x < y$ with two constraints $x \leq y$ and $x \neq y$ and a cost-preserving reduction from $\text{MINCSP}(=, \neq)$ to $\text{MINCSP}(\leq, \neq)$ by replacing every constraint of the form $x = y$ with $x \leq y, y \leq x$.¹ In fact, these reductions show that $\text{MINCSP}(<, \leq, =, \neq)$ is equivalent to $\text{MINCSP}(\leq, \neq)$ under cost-preserving reductions. While EDGE MULTICUT and SUBSET-DFAS are known to be in FPT (by [23, 7] and [9], respectively), the parameterized complexity status of DSMC was open [13] (see also [11, 24, 20]) prior to our work. In a related problem called DIRECTED MULTICUT the input contains a directed graph and a set of ordered vertex pairs $\{(s_1, t_1), \dots, (s_m, t_m)\}$, and the goal is to delete the smallest number of arcs from the graph so

¹ The latter reduction is analogous to saying that EDGE MULTICUT is the same problem as DSMC on bidirected graphs.



■ **Figure 1** These are seven subsets of Point Algebra. Arrow represent polynomial-time cost-preserving reductions between corresponding MINCSPs that either follow by inclusion or using $(x = y) \equiv (x \leq y) \wedge (y \leq x)$ and $(x < y) \equiv (x \leq y) \wedge (x \neq y)$. MINCSPs for all of them are NP-hard, and in FPT for all except the red language, for which it is W[1]-hard. There are eight more non-empty subsets, out of which four give rise to polynomial-time solvable MINCSPs (\neq and $\{\leq, =\}$ with subsets), subset $\{<, \neq\}$, which reduces to $\{<\}$ because all \neq -constraints can be safely disregarded, and three more subsets that contain both \leq and \neq .

that no s_i reaches t_i . This problem is W[1]-hard [23] even with four pairs (i.e. $m = 4$) [25].

The scope of our study also includes less prominent and new problems: $\text{MINCSP}(<, =)$ and $\text{MINCSP}(<, =, \neq)$. We remark that the former problem appeared in [11] and was solved by reduction into $\text{MINCSP}(<, \leq)$. These can be thought of as generalizations of MULTICUT with asymmetric cut requests: intuitively, if one thinks of the equality constraints as edges in an undirected graph, the constraints of the form $x \neq y$ correspond to the usual, symmetric cut requests that require separating x and y into distinct connected components, while constraints of the form $x < y$ additionally require that the connected components can be ordered so that the component of x precedes the component of y .

Our contributions. We fully classify the complexity of MINCSP for all subsets of Point Algebra. For polynomial-time complexity, observe that $\text{MINCSP}(\Gamma)$ is NP-hard unless $\Gamma \subseteq \{=, \leq\}$ or $\Gamma = \{\neq\}$: on the one hand, if $\Gamma \subseteq \{=, \leq\}$ or $\Gamma = \{\neq\}$, then every instance is satisfiable at zero cost using, respectively, any constant or any injective assignment; on the other hand, if Γ is not covered by these cases, then it contains either $\{<\}$, $\{=, \neq\}$ or $\{\leq, \neq\}$ as a subset, all of which imply NP-hardness by reductions from DFAS and MULTICUT . Our main result is the parameterized complexity classification. We provide algorithms for the weighted version of $\text{MINCSP}(\Gamma)$, where every constraint comes with an integer weight, and the input also contains a weight budget W . We are allowed to break at most k constraints of total cost at most W – observe that the parameter is only k . On the other hand, our lower bounds work in the weightless version.

► **Theorem 1.** *Let $\Gamma \subseteq \{<, \leq, =, \neq\}$.*

1. *If $\Gamma \subseteq \{<, \leq, =\}$, then $\text{WEIGHTED MINCSP}(\Gamma)$ is fixed-parameter tractable.*
2. *If $\Gamma \subseteq \{<, =, \neq\}$, then $\text{WEIGHTED MINCSP}(\Gamma)$ is fixed-parameter tractable.*
3. *Otherwise, $\text{MINCSP}(\Gamma)$ is W[1]-hard.*

The first point is easily obtained by reduction to $\text{MINCSP}(<, \leq)$ and applying an FPT algorithm of Chitnis et al. [9] or Kim et al. [20]. For the second point, we design a new algorithm in Section 3 using flow augmentation (in the guise of BOOLEAN MINCSP). Our

algorithm for $\text{MINCSP}(<, =, \neq)$ combines two strains of flow augmentation-based algorithms that have appeared (explicitly or implicitly) in the literature: one that is suitable for undirected separation problems like MULTICUT (cf. [17, 12, 20]) and another suitable for directed transversal problems like DFAS and SUBSET-DFAS (cf. [18, 11, 20]). One could have hoped that pushing these ideas one step further would solve the more general DSMC problem – alas, this hope is squashed in Section 4 where we prove that $\text{MINCSP}(\leq, \neq)$, i.e. DSMC , is $\text{W}[1]$ -hard. By noting that every language not covered by the first and the second point of the theorem contains both \neq and \leq , this completes the classification. See Figure 1 for an illustration.

2 Preliminaries

Fix a domain of values D . A relation R of arity r on D is a subset of tuples in D^r . An instance of a *constraint satisfaction problem* (CSP) is a set of variables V and a collection of constraints \mathcal{C} of the form $R(x_1, \dots, x_r)$, where R is a relation of arity r and x_1, \dots, x_r are variables in V . The set $\{x_1, \dots, x_r\}$ is called the *scope* of constraint $R(x_1, \dots, x_r)$. An assignment $\alpha : V \rightarrow D$ *satisfies a constraint* $R(x_1, \dots, x_r)$ if $(\alpha(x_1), \dots, \alpha(x_r)) \in R$, otherwise we say that it *breaks* the constraint. Similarly, an assignment *satisfies an instance* $\mathcal{I} = (V, \mathcal{C})$ of a CSP if it satisfies all constraints in \mathcal{C} . A *constraint language* Γ is a set of relations on D , and $\text{CSP}(\Gamma)$ is the CSP problem in which constraint relations come from the set Γ .

The $\text{MINIMUM-COST CONSTRAINT SATISFACTION PROBLEM OVER } \Gamma$ ($\text{MINCSP}(\Gamma)$) is defined as follows: given an instance $\mathcal{I} = (V, \mathcal{C})$ of $\text{CSP}(\Gamma)$, a function $\kappa : \mathcal{C} \rightarrow \mathbb{Z}_+$ and an integer parameter k , decide whether there exists $X \subseteq \mathcal{C}$ such that $\sum_{C \in X} \kappa(C) \leq k$ and $(V, \mathcal{C} \setminus X)$ is satisfiable. We refer to the set X as the solution. Note that the constraints of cost more than k cannot be in the solution, while a constraint of cost $\ell \leq k$ can be replaced by ℓ unit-cost copies. Thus, we assume without loss of generality that the cost of every constraint is either 1 or ∞ , and refer to such constraints as *soft* and *crisp*, respectively. We can also handle an additional weight budget and arrive at the following definition.

| WEIGHTED $\text{MINCSP}(\Gamma)$ | |
|----------------------------------|--|
| INSTANCE: | Instance (V, \mathcal{C}) of $\text{CSP}(\Gamma)$, functions $\kappa : \mathcal{C} \rightarrow \{1, \infty\}$, $\omega : \mathcal{C} \rightarrow \mathbb{Z}_+$ and integers k, W . |
| PARAMETER: | k . |
| QUESTION: | Does there exist $X \subseteq \mathcal{C}$ such that $\sum_{C \in X} \kappa(C) \leq k$, $\sum_{C \in \mathcal{C}} \omega(C) \leq W$ and $(V, \mathcal{C} \setminus X)$ is satisfiable? |

Note that W is not part of the parameter. Moreover, no constraint of weight more than W can be part of a solution, so we assume those are always crisp.

The *Point Algebra* is a constraint language on the domain \mathbb{Q} with four binary relations $<$, \leq , $=$ and \neq . We use infix notation for Point Algebra constraints, e.g. $x < y$ and $x \neq y$. One can check in polynomial time whether an instance of $\text{CSP}(<, \leq, =, \neq)$ is satisfiable by, e.g., reducing to $\text{CSP}(\neq, \leq)$, constructing directed graph of \leq -constraints and, for every constraint $x \neq y$, checking that x and y are not strongly connected (cf. [29]).

3 FPT Algorithm for $\text{MinCSP}(<, =, \neq)$

In this section we prove the following theorem.

► **Theorem 2.** *WEIGHTED $\text{MINCSP}(<, =, \neq)$ is fixed-parameter tractable.*

Our algorithm follows the compress-branch-cut paradigm (cf. [26, 8, 23]). On the high level, we use iterative compression to obtain an approximate solution of size bounded by a function of the parameter. This allows us to exhaustively guess certain information about the variables appearing in the approximate solution and then reduce the problem to a fixed-parameter tractable BOOLEAN MINCSP. The last step blends two types of reductions suitable for symmetric and asymmetric properties, respectively.

We start this section by describing the compression and branching steps, which are quite standard. Then we set up the BOOLEAN MINCSP machinery, describe the algorithm for MINCSP($<, =, \neq$) and prove its correctness. We finish with some high-level observations about the algorithm and the obstacles in pushing it further.

Compression and Branching

Let $\mathcal{I} = (V, \mathcal{C}, \kappa, \omega, k, W)$ be an instance of WEIGHTED MINCSP($<, =, \neq$). By a standard iterative compression argument (cf. [10, Chapter 4]), we may assume access to a set of constraints $X_{\text{in}} \subseteq \mathcal{C}$ of size at most $k + 1$ such that $(V, \mathcal{C} \setminus X_{\text{in}})$ is satisfiable. Moreover, if \mathcal{I} is a yes-instance, then by branching over subsets of X_{in} we may assume that it admits an optimal solution X_{opt} disjoint from X_{in} , and an assignment α_{opt} that satisfies $(V, \mathcal{C} \setminus X_{\text{opt}})$. Since $|V(X_{\text{in}})| \leq 2(k + 1)$, we can furthermore enumerate ordered partitions of $V(X_{\text{in}})$ in $O^*(k^k)$ time and thus guess an assignment $\alpha_{\text{in}} : V(X_{\text{in}}) \rightarrow \mathbb{Q}$ that *agrees* with α_{opt} in the following sense: for every $x, y \in V(X_{\text{in}})$, we have

- $\alpha_{\text{in}}(x) = \alpha_{\text{in}}(y)$ if and only if $\alpha_{\text{opt}}(x) = \alpha_{\text{opt}}(y)$, and
- $\alpha_{\text{in}}(x) < \alpha_{\text{in}}(y)$ if and only if $\alpha_{\text{opt}}(x) < \alpha_{\text{opt}}(y)$.

Since $X_{\text{in}} \cap X_{\text{opt}} = \emptyset$ and α_{in} agrees with α_{opt} , we obtain that α_{in} satisfies X_{in} . Moreover, every assignment that agrees with α_{in} satisfies X_{in} as well, so we can safely remove X_{in} from \mathcal{C} . Assuming our guesses were correct, we now have a satisfiable instance $(V, \mathcal{C} \setminus X_{\text{in}})$ of CSP($<, =, \neq$) and, additionally, a partial assignment $\alpha_{\text{in}} : V(X_{\text{in}}) \rightarrow \mathbb{Q}$ with the promise that, if \mathcal{I} is a yes-instance, then there is an assignment to \mathcal{I} that agrees with α_{in} and breaks constraints of cost at most k and weight at most W . Moreover, if $\alpha_{\text{in}}(x) = \alpha_{\text{in}}(y)$ for some $x, y \in V(X_{\text{in}})$, then we can identify x and y . Thus, in time $O^*(k^k)$ we have reduced the problem to the following version.

| | |
|--|---|
| COMPRESSED WEIGHTED MINCSP($<, =, \neq$) | |
| INSTANCE: | A satisfiable instance (V, \mathcal{C}) of CSP($<, =, \neq$), functions $\kappa : \mathcal{C} \rightarrow \{1, \infty\}$ and $\omega : \mathcal{C} \rightarrow \mathbb{Z}_+$, integers k and W , a subset $U \subseteq V$ with $ U \leq 2(k + 1)$ and an injective assignment $\alpha : U \rightarrow \mathbb{Q}$. |
| PARAMETER: | k . |
| QUESTION: | Does there exist $X \subseteq \mathcal{C}$ such that $\sum_{C \in X} \kappa(C) \leq k$, $\sum_{C \in X} \omega(C) \leq W$, and $(V, \mathcal{C} \setminus X)$ is satisfiable by an assignment that agrees with α ? |

By a standard correctness argument (cf. [10, Chapter 4]), we have the following.

► **Proposition 3.** *If COMPRESSED WEIGHTED MINCSP($<, =, \neq$) is fixed-parameter tractable, then WEIGHTED MINCSP($<, =, \neq$) is fixed-parameter tractable.*

To solve the compressed version, we reduce it into a separation problem handled by the BOOLEAN MINCSP machinery.

Cutting tool: Boolean MinCSP

A Boolean constraint language is a set of relations on domain $\{0, 1\}$. Kim et al. [19] classified parameterized complexity of WEIGHTED MINCSP(Γ) for every finite Boolean constraint

language Γ . We will use the positive part of their classification. To describe it, we start with some definitions.

A Boolean relation $R \subseteq \{0, 1\}^r$ can be modeled as a set of satisfying assignments to a propositional formula on r variables. Formally, let ϕ be a propositional formula on variables x_1, \dots, x_r . For $b_1, \dots, b_r \in \{0, 1\}$, let $\phi(b_1, \dots, b_r)$ be the Boolean expression obtained by substituting variable x_i with the Boolean value b_i for all $i \in \{1, \dots, r\}$. Then, for every $R \subseteq \{0, 1\}^r$, there exists a formula ϕ such that

$$R = \{(b_1, \dots, b_r) \in \{0, 1\}^r \mid \phi(b_1, \dots, b_r) \text{ evaluates to true}\}.$$

We remark in passing that ϕ does not have to be unique.

A propositional formula is *bijunctive* if it is a conjunction of 1- or 2-clauses, i.e. subformulas of the form (x) , $(\neg x)$, $(x \rightarrow y)$, $(x \vee y)$ and $(\neg x \vee \neg y)$. Associate an undirected *Gaifman graph* G_ϕ with ϕ : let $\{1, \dots, r\}$ be the vertices of G_ϕ and let G_ϕ contain an edge ij whenever x_i and x_j appear together in a 2-clause in ϕ . We say that a Boolean relation is *bijunctive* if it is definable by a bijunctive propositional formula. Furthermore, a bijunctive relation is *$2K_2$ -free* if it is definable by a bijunctive formula ϕ such that G_ϕ does not contain an induced $2K_2$, i.e. for every pair of vertex-disjoint edges ij and $i'j'$ in G_ϕ , there is an edge with one endpoint in $\{i, j\}$ and another in $\{i', j'\}$.

► **Theorem 4** (Theorem 1.2 in [19]). *Let Γ be a set of $2K_2$ -free bijunctive Boolean relations of arity at most r . Then $\text{WEIGHTED MINCSP}(\Gamma)$ is in FPT parameterized by $k + r$.*

In the forthcoming description of the algorithm we will abuse the notation and define Boolean constraints explicitly using propositional formulas.

Solving the Compressed Version

We define a procedure that takes an instance \mathcal{I} of COMPRESSED WEIGHTED MINCSP($<, =, \neq$) as input and in polynomial time outputs an instance \mathcal{I}' of WEIGHTED MINCSP(Γ) with the same parameter k , with Γ being a Boolean constraint language. We will argue that all relations in Γ are bijunctive $2K_2$ -free and of arity at most $O(k)$ (in Lemma 5), and \mathcal{I} is a yes-instance if and only if \mathcal{I}' is a yes-instance (in Lemmas 6 and 7). Combined with Proposition 3 and Theorem 4, this yields Theorem 2.

We start by describing the reduction procedure. Let the input be an instance $\mathcal{I} = (V, \mathcal{C}, \kappa, \omega, k, W, U, \alpha)$ of COMPRESSED WEIGHTED MINCSP($<, =, \neq$). Construct the output instance $\mathcal{I}' = (V', \mathcal{C}', \kappa', \omega', k, W)$ of WEIGHTED MINCSP(Γ) with the same values k and W as follows. Let $\ell = |U|$ and observe that $\ell \leq 2(k + 1)$. Enumerate U as u_1, \dots, u_ℓ ordered so that $\alpha(u_1) < \alpha(u_2) < \dots < \alpha(u_\ell)$. For every variable $v \in V$, introduce Boolean variables $c_{v,i}$ for $i \in \{1, \dots, \ell\}$ and $p_{v,j}$ for $j \in \{1, \dots, 2\ell + 1\}$ in V' .

Before defining the constraints of \mathcal{C}' , we elaborate on the intended interpretation of the Boolean variables in V' . Suppose α_{opt} is an assignment that breaks constraints of total cost at most k , total weight at most W and agrees with α_{in} on U . Variables $c_{v,i}$ indicate whether $\alpha_{\text{opt}}(v)$ equals $\alpha_{\text{opt}}(u_i)$ for some $u_i \in U$. Since α_{in} is injective, $c_{v,i}$ equals 1 for at most one i . Variables $p_{v,j}$ encode how the values $\alpha_{\text{opt}}(v)$ are ordered with respect to $\alpha_{\text{opt}}(u_i)$. More precisely, $p_{v,2i}$ is set to 1 if and only if $\alpha_{\text{opt}}(v) \geq \alpha_{\text{opt}}(u_i)$ and $p_{v,2i+1}$ is set to 1 if and only if $\alpha_{\text{opt}}(v) > \alpha_{\text{opt}}(u_i)$. Note that $p_{v,1} = 1$ should hold vacuously in this interpretation, while $p_{v,j'} = 1$ implies $p_{v,j} = 1$ for all $j < j'$ since $\alpha_{\text{in}}(u_j) < \alpha_{\text{in}}(u_{j'})$. Thus, for a variable v , the variables $c_{v,i}$ and $p_{v,j}$ represent two distinct ways of encoding integer values into the Boolean domain; e.g., for $\ell = 3$ the valid combined values for the variables $c_{v,i}$ are 000, 001,

010, 100 while the valid values for the variables $p_{v,j}$ are 1000000, 1100000, \dots , 1111111. The c -variables are used to implement constraints ($v \neq w$) while the p -variables are used to implement constraints ($v < w$). The full argument for correctness here is somewhat intricate and depends on assumptions from the iterative compression step, but informally, the reason we need two encodings is that we need our constraints to be bijunctive. As an illustration, for variables v, w and some value $i \in [\ell]$, an assignment α that sets $\alpha(v) = \alpha(w) = \alpha(u_i)$ can be eliminated by a clause $(\neg c_{v,i} \vee \neg c_{w,i})$, and an assignment that sets $\alpha(v) \geq \alpha(u_i) \geq \alpha(w)$ can be eliminated by a clause $(p_{v,2i} \rightarrow p_{w,2i+1})$. In the other direction, the corresponding conditions cannot be detected by looking at only two variables at a time.

With the intuition in mind, we populate \mathcal{C}' with crisp constraints. For every $v \in V$ and every $1 \leq i < i' \leq \ell$, add a crisp Boolean constraint

$$(\neg c_{v,i} \vee \neg c_{v,i'}). \quad (1)$$

For every $u_i \in U$, add a crisp Boolean constraint

$$(c_{u_i,i}). \quad (2)$$

For every $v \in V$, add crisp Boolean constraints

$$(p_{v,1}) \text{ and } (p_{v,j} \leftarrow p_{v,j'}) \text{ for all } 1 \leq j < j' \leq 2\ell + 1. \quad (3)$$

For every $u_i \in U$, add crisp Boolean constraints

$$(p_{u_i,2i}), (\neg p_{u_i,2i+1}). \quad (4)$$

Finally, for every $v \in V$ and $i \in \{1, \dots, \ell\}$, add crisp Boolean constraints

$$\begin{aligned} (c_{v,i} \rightarrow p_{v,j}) & \quad \text{for all } 1 \leq j \leq 2i, \text{ and} \\ (c_{v,i} \rightarrow \neg p_{v,j}) & \quad \text{for all } 2i < j \leq 2\ell + 1. \end{aligned} \quad (5)$$

Now, for every $C \in \mathcal{C}$, we define a constraint C' in \mathcal{C}' with $\kappa'(C') = \kappa(C)$ and $\omega'(C') = \omega(C)$. We will use **purple** for clauses that also appear as crisp clauses. If C is an equality constraint ($v = w$), then let C' be the Boolean constraint

$$\begin{aligned} & \bigwedge_{i=1}^{\ell} (c_{v,i} = c_{w,i}) \wedge \bigwedge_{i=1}^{\ell} \bigwedge_{i'=i+1}^{\ell} (\neg c_{v,i} \vee \neg c_{v,i'}) \wedge \\ & \bigwedge_{j=1}^{2\ell+1} (p_{v,j} = p_{w,j}) \wedge \bigwedge_{j=1}^{2\ell+1} \bigwedge_{j'=j+1}^{2\ell+1} (p_{v,j} \leftarrow p_{v,j'}) \wedge \\ & \bigwedge_{i=1}^{\ell} \bigwedge_{j=1}^{2i} (c_{v,i} \rightarrow p_{v,j}) \wedge \bigwedge_{i=1}^{\ell} \bigwedge_{j=2i+1}^{2\ell+1} (c_{v,i} \rightarrow \neg p_{v,j}), \end{aligned} \quad (6)$$

where clause $(x = y)$ is a short-hand for the conjunction $(x \rightarrow y) \wedge (x \leftarrow y)$. If C is an disequality constraint ($v \neq w$), then let C' be the Boolean constraint

$$\bigwedge_{i=1}^{\ell} (\neg c_{v,i} \vee \neg c_{w,i}) \wedge \bigwedge_{i=1}^{\ell} \bigwedge_{i'=i+1}^{\ell} (\neg c_{v,i} \vee \neg c_{v,i'}) \quad (7)$$

Finally, if C is an ordering constraint ($v < w$), then let C' be the Boolean constraint

$$\bigwedge_{i=1}^{\ell} (p_{v,2i-1} \rightarrow p_{w,2i-1}) \wedge \bigwedge_{i=1}^{\ell} (p_{v,2i} \rightarrow p_{w,2i+1}) \wedge \bigwedge_{j=1}^{2\ell+1} \bigwedge_{j'=j+1}^{2\ell+1} (p_{v,j} \leftarrow p_{v,j'}) \quad (8)$$

This completes the construction.

Clearly, the reduction requires polynomial time. To show that \mathcal{I}' can be decided in FPT time with respect to k , we use Theorem 4 and the following lemma.

► **Lemma 5.** *Every Boolean relation used in the constraints of \mathcal{C}' is bijunctive $2K_2$ -free and of arity at most $8k + 10$.*

Proof. The Gaifman graphs of the propositional formulas in Equations (1)–(5) contain at most two vertices. The Gaifman graphs of propositional formulas in Equations (6)–(8) consist of a clique (formed by the edges corresponding to clauses that also appear as crisp constraints defined in Equations (1), (3), and (5) and are highlighted in purple) and edges with exactly one endpoint in the clique. These graphs are $2K_2$ -free. The constraints of maximum arity are defined in Equation (6), each contains $2(\ell + 1 + \ell) = 4\ell + 2 \leq 4(2k + 2) + 2 \leq 8k + 10$ variables in its scope. ◀

To show that the reduction is correct, we start with the forward direction.

► **Lemma 6.** *If \mathcal{I} is a yes-instance of COMPRESSED WEIGHTED MINCSP($<, =, \neq$), then \mathcal{I}' is a yes-instance of WEIGHTED MINCSP(Γ).*

Proof. Let X be a solution to \mathcal{I} , i.e. a subset of constraints in \mathcal{C} such that $\sum_{C \in X} \kappa(C) \leq k$, $\sum_{C \in X} \omega(C) \leq W$, and $(V, \mathcal{C} \setminus X)$ admits a satisfying assignment $\beta : V \rightarrow \mathbb{Q}$ that agrees with the partial assignment $\alpha : U \rightarrow \mathbb{Q}$. Define a subset of Boolean constraints $X' = \{C' \in \mathcal{C}' : C \in X\}$. By construction, X' has cost at most k and weight at most W , so it remains to show that $(V', \mathcal{C}' \setminus X')$ is satisfiable. To this end, we define an assignment $\beta' : V' \rightarrow \{0, 1\}$ as follows. For every $v \in V$ and $i \in \{1, \dots, \ell\}$, set $\beta'(c_{v,i}) = 1$ if and only if $\beta(c_{v,i}) = \beta(u_i)$. For every $v \in V$ and $i \in \{1, \dots, \ell\}$, set $\beta'(p_{v,1}) = 1$, $\beta'(p_{v,2i}) = 1$ if and only if $\beta(v) \geq \beta(u_i)$, and $\beta'(p_{v,2i+1}) = 1$ if and only if $\beta(v) > \beta(u_i)$.

Observe that since α is injective and β agrees with α , the assignment β is also injective on U , so β' satisfies the constraints defined in Equations (1), (2), and (4). Further, $(\beta'(p_{v,1}), \dots, \beta'(p_{v,2\ell+1}))$ is a vector starting with ones followed by (possibly no) zeros, so β' satisfies the constraints defined in Equation (3). Moreover, $\beta'(c_{v,i}) = 1$ if and only if $\beta(v) = \beta(u_i)$, in which case $\beta'(p_{v,2i}) = 1$ and $\beta'(p_{v,2i+1}) = 0$, hence β' satisfies the constraints defined in Equation (5).

Now consider a constraint $C \in \mathcal{C} \setminus X$ satisfied by β . We claim that β' satisfies the corresponding Boolean constraint $C' \in \mathcal{C}' \setminus X'$. Note that it is sufficient to check that the clauses not present in the crisp constraints defined in Equations (1), (3), and (5) are satisfied. Suppose C is an equality constraint ($v = w$) and recall the definition of C' from Equation (6). By definition of β' , it is clear that $\beta(v) = \beta(w)$ implies that $\beta'(c_{v,i}) = \beta'(c_{w,i})$ for all i and $\beta'(p_{v,j}) = \beta'(p_{w,j})$ for all j , so β' satisfies C' . Now suppose C is a disequality constraint ($v \neq w$) and recall the definition of C' from Equation (7). If $\beta(v) = \beta(u_i)$ for some i , then $\beta(w) \neq \beta(u_i)$, and $\beta'(c_{v,i}) = 1$ implies that $\beta'(c_{w,i}) = 0$. Otherwise, $\beta(v) \neq \beta(u_i)$ for all i , and $\beta'(c_{v,i}) = 0$ for all i . In both cases, β' satisfies C' . Finally, if C is an ordering constraint ($v < w$), recall the definition of C' from Equation (8). If $\beta(v) = \beta(u_i)$ for some i , then $\beta(u_i) < \beta(w)$, hence $\beta'(p_{v,2i}) = 1$, $\beta'(p_{v,2i+1}) = 0$ and $\beta'(p_{w,2i}) = \beta'(p_{w,2i+1}) = 1$. Otherwise, we have three more cases:

- if $\beta(v) < \beta(u_1)$, then $\beta'(p_{v,j}) = 0$ for all $j \geq 2$;
- if $\beta(u_{i-1}) < \beta(v) < \beta(u_i)$ for some i , then $\beta(u_{i-1}) < \beta(w)$, so $\beta'(p_{v,j}) = 0$ for all $j \geq 2i$ and $\beta'(p_{w,j'}) = 1$ for all $j' < 2i$;
- if $\beta(v) > \beta(u_\ell)$, then $\beta(w) > \beta(u_\ell)$, so $\beta'(p_{v,j}) = \beta'(p_{w,j}) = 1$ for all j .

In all cases, β' satisfies C' . ◀

23:10 MinCSP over the Point Algebra

To complete the proof of Theorem 2, it remains to show the converse Lemma 6.

► **Lemma 7.** *If \mathcal{I}' is a yes-instance of WEIGHTED MINCSP(Γ), then \mathcal{I} is a yes-instance of COMPRESSED WEIGHTED MINCSP($<, =, \neq$).*

Proof. Let X' be a solution to \mathcal{I}' , i.e. a subset of constraints in \mathcal{C}' such that $\sum_{C' \in X'} \kappa'(C') \leq k$, $\sum_{C' \in X'} \omega'(C') \leq W$, and $(V', \mathcal{C}' \setminus X')$ is satisfiable. Fix an assignment $\beta' : V' \rightarrow \{0, 1\}$ that satisfies $(V', \mathcal{C}' \setminus X')$ and define $X = \{C \in \mathcal{C} : C' \in X'\}$. By construction, we have $\sum_{C \in X} \kappa(C) \leq k$ and $\sum_{C \in X} \omega(C) \leq W$, so it remains to show that $(V, \mathcal{C} \setminus X)$ admits a satisfying assignment $\beta : V \rightarrow \mathbb{Q}$ that agrees with $\alpha : U \rightarrow \mathbb{Q}$. To define such an assignment, recall that (V, \mathcal{C}) is satisfiable, and let $\gamma : V \rightarrow \mathbb{Q}$ be a satisfying assignment. Without loss of generality, assume that the range of γ is $(0, 1)$. Define $\iota : V \rightarrow \{1, \dots, \ell\}$ as $\iota(v) := \max\{i : \beta'(p_{v,2i}) = 1\}$ and let

$$\beta(v) = \begin{cases} i & \text{if there exists } i \text{ such that } \beta'(c_{v,i}) = 1, \\ \iota(v) + \gamma(v) & \text{otherwise.} \end{cases}$$

Note that $\beta(u_i) = i$ for all i , so it agrees with α on U . We claim that if β' satisfies a constraint $C' \in \mathcal{C}' \setminus X'$, then β satisfies the corresponding constraint $C \in \mathcal{C} \setminus X$. Note that by the crisp constraints from Equation (1), we have $\beta'(c_{v,i}) = 1$ for at most one i . Suppose C is an equality constraint ($v = w$), and recall C' from Equation (6). If there exists i such that $\beta'(c_{v,i}) = 1$, then $\beta'(c_{w,i}) = 1$ and $\beta(v) = \beta(w) = i$. Otherwise, note that $\iota(v) = \iota(w)$ since $\beta'(p_{v,j}) = \beta'(p_{w,j})$ for all j , and $\gamma(v) = \gamma(w)$ since γ satisfies $(v = w)$. Hence, $\beta(v) = \beta(w)$ in this case as well. Now suppose that C is a disequality constraint ($v \neq w$), and recall C' from Equation (7). If there exists i such that $\beta'(c_{v,i}) = 1$, then $\beta'(c_{w,i}) = 0$ and $\beta(v) = i \neq \beta(w)$. Otherwise, $\gamma(v) \neq \gamma(w)$ since γ satisfies $(v \neq w)$, so the fractional parts of $\beta(v)$ and $\beta(w)$ are distinct, and $\beta(v) \neq \beta(w)$. Finally, suppose C is an ordering constraint ($v < w$). If there exists i such that $\beta'(c_{v,i}) = 1$, then $\beta'(p_{v,2i}) = 1$ and $\beta'(p_{w,2i+1}) = 1$, so $\beta(v) = i$ and $\beta(w) = \iota(w) + \gamma(w) > i$. Otherwise, since $p_{v,2i-1} \rightarrow p_{w,2i-1}$ for all i , we have $\iota(v) \leq \iota(w)$ and $\gamma(v) < \gamma(w)$ since γ satisfies $(v < w)$. Hence, $\beta(v) < \beta(w)$, as desired. ◀

4 W[1]-hardness of Directed Symmetric Multicut

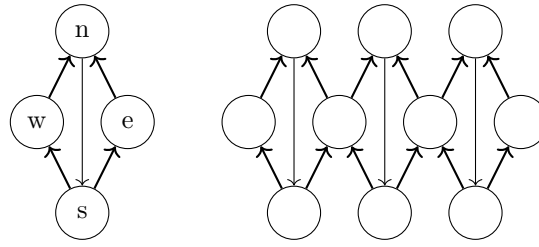
Recall the definition of the problem.

| DIRECTED SYMMETRIC MULTICUT (DSMC) | |
|------------------------------------|---|
| INSTANCE: | A directed graph D , a collection \mathcal{P} of vertex pairs in D , and integer k . |
| PARAMETER: | k . |
| QUESTION: | Does there exist a set X of at most k arcs in D such that no pair $\{s, t\} \in \mathcal{P}$ is strongly connected in $D - X$? |

Pairs in \mathcal{P} are referred to as cut requests. In this section we prove the following.

► **Theorem 8.** *DIRECTED SYMMETRIC MULTICUT parameterized by solution size is W[1]-hard.*

Our starting point is a multicolored variant of k -CLIQUE: given an undirected graph G , an integer k and a partition $V^1 \uplus \dots \uplus V^k$ of $V(G)$ such that $|V^1| = \dots = |V^k| = n$, the question is whether G contains a complete subgraph with one vertex from each V^i . Given an



■ **Figure 2** A diamond digraph on the left and a sequence of three joined diamonds on the right. Undeleteable arcs are drawn with thick lines, and deleteable arcs are drawn with a thin line.

instance $(G, k, V^1 \uplus \dots \uplus V^k)$ of this problem, we construct an instance (D, \mathcal{P}, k') of DSMC, where $k' = 3k^2$.

On the high level, the construction consists of two parts – choice gadgets for every vertex set V^i , and coordination gadgets that disallow choosing non-adjacent vertices into the solution. The building blocks of the choice gadgets are *diamonds*, which are graphs \diamond on four vertices w, n, e, s (for west, north, east and south) with four undeleteable arcs sw, wn, se, en and one deleteable arc ns (note that we can make an arc essentially undeleteable by having $k + 1$ parallel copies). Note that \diamond is strongly connected, but loses this property if ns is deleted. We refer to vertices w and e of a diamond as *junction* vertices. By *picking a diamond* we mean deleting the arc ns . By *joining two diamonds* we mean identifying the eastern vertex of the first with the western vertex of the other. Note that the result of sequentially joining diamonds is a strongly connected graph which loses this property if any diamond is picked. See Figure 2 for an illustration.

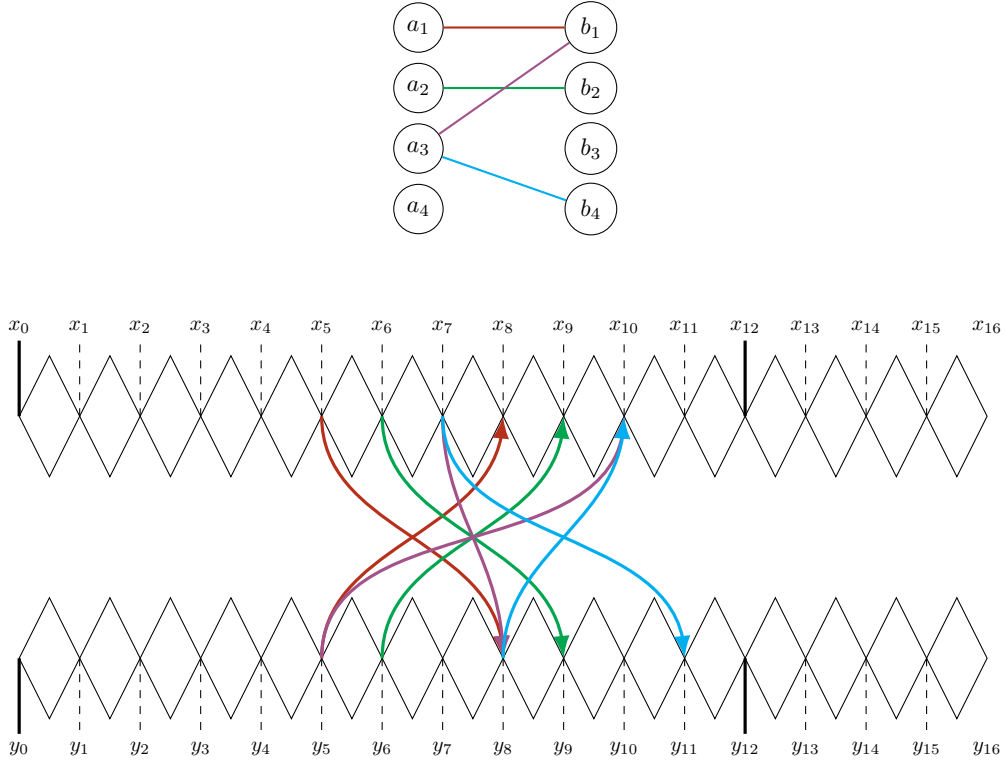
Choice gadgets

Denote the vertices of V^i by v_1^i, \dots, v_n^i . For every subset V^i in G , create a *necklace* of $3kn$ diamonds in D joined in a cyclic fashion: construct k strings S_j^i for $1 \leq j \leq k$, where each S_j^i consists of $3n$ sequentially joined diamonds $\diamond_1^{i,j}, \dots, \diamond_{3n}^{i,j}$; create the necklace by joining the last diamond of string S_j^i with the first diamond of $S_{j+1 \bmod k}^i$ for all $1 \leq j \leq k$. Let the junction vertices of the diamonds on the necklace be c_0, \dots, c_{3kn-1} . Add cut requests $\{c_\alpha, c_{\alpha+n \bmod 3kn}\}$ to \mathcal{P} for all $0 \leq \alpha < 3kn$.

Before proceeding with coordination gadgets, we observe some properties of the necklace. To satisfy the cut requests introduced so far, a solution needs to pick at least $3k$ diamonds since we have requests $\{c_{(i-1)n}, c_{in}\}$ for every $i \in [3k]$ whose endpoints occur in arc-disjoint strongly connected subgraphs. The budget is $k' = k \cdot 3k$, and there are k such necklaces, so a solution has to pick exactly $3k$ diamonds in each necklace. We claim that picked diamonds are evenly spaced, i.e. there exists $\alpha \in \{1, \dots, n\}$ such that the solution picks diamonds $\diamond_\alpha^{i,j}, \diamond_{\alpha+n}^{i,j}$ and $\diamond_{\alpha+2n}^{i,j}$ in the strings S_j^i for all $1 \leq j \leq k$. Indeed, if a hypothetical solution picks any other set of $3k$ diamonds, then a sequence of at least n joined and unpicked diamonds remains on the necklace, and any such sequence contains an unsatisfied cut request. Intuitively, we can interpret a solution picking diamonds $\diamond_\alpha^{i,1}$ as choosing vertex $v_\alpha^i \in V^i$ to be part of the clique in G .

Coordination gadgets

Now we construct the coordination gadgets that disallow choosing non-adjacent vertices. Consider a pair of non-adjacent vertices v_α^i and v_β^j in G with $i < j$. Let the junction vertices



■ **Figure 3** Snippet of the construction of the gadget (bottom) for the graph G of non-edges (top) with $n = 4$ and $k = 2$. Vertices x_0, \dots, x_{16} and y_0, \dots, y_{16} are junction vertices of the necklaces corresponding to the left hand side and the right hand side of G , respectively. Colorful arrows in the gadget represent paths of crossing arcs. Solid delimiters at x_0, x_{12} and y_0, y_{12} indicate starts of new strings. For the sake of clarity, s - and t -vertices are omitted from the picture.

in strings S_j^i and S_i^j be $x_0^{i,j}, \dots, x_{3n-1}^{i,j}$ and $y_0^{j,i}, \dots, y_{3n-1}^{j,i}$, respectively. Introduce vertices $s_{\alpha,\beta}^{i,j}$ and $t_{\alpha,\beta}^{j,i}$ and add four undeletable arcs forming directed paths

$$x_{n+\alpha}^{i,j} \rightarrow s_{\alpha,\beta}^{i,j} \rightarrow y_{2n+\beta-1}^{j,i} \quad \text{and} \quad y_{n+\beta}^{j,i} \rightarrow t_{\alpha,\beta}^{j,i} \rightarrow x_{2n+\alpha-1}^{i,j}.$$

Call these four arcs *crossing*. Add cut request $\{s_{\alpha,\beta}^{i,j}, t_{\alpha,\beta}^{j,i}\}$ to \mathcal{P} . Observe that this request requires the solution to pick a diamond between $x_{n+\alpha}^{i,j}$ and $x_{2n+\alpha-1}^{i,j}$ or a diamond between $y_{n+\beta}^{j,i}$ and $y_{2n+\beta-1}^{j,i}$.

The construction is complete (see Figure 3). We proceed with the correctness proof.

Directed Symmetric Multicut to Clique

Suppose X is a solution to (D, \mathcal{P}, k') . Let $X' \subseteq V(G)$ contain the vertices of G picked by X , i.e. X' contains v_α^i whenever the deletable arc of $\diamond_\alpha^{i,1}$ is in X . Note that $|X'| = k$ by the observation that α is unique to the set V^i . We claim that X' forms a clique in G . Suppose for the sake of contradiction that two non-adjacent vertices v_α^i, v_β^j are in X' . Let the junction vertices in S_j^i and S_i^j be $x_0^{i,j}, \dots, x_{3n-1}^{i,j}$ and $y_0^{j,i}, \dots, y_{3n-1}^{j,i}$, respectively. By construction, X picks diamonds $\diamond_\alpha^{i,j}, \diamond_{\alpha+n}^{i,j}, \diamond_{\alpha+2n}^{i,j}$ in S_j^i and $\diamond_\beta^{j,i}, \diamond_{\beta+n}^{j,i}, \diamond_{\beta+2n}^{j,i}$ in S_i^j . Then $D - X$ contains a closed walk

$$x_{n+\alpha}^{i,j} \rightarrow s_{\alpha,\beta}^{i,j} \rightarrow y_{2n+\beta-1}^{j,i} \rightarrow \dots \rightarrow y_{n+\beta}^{j,i} \rightarrow t_{\alpha,\beta}^{j,i} \rightarrow x_{2n+\alpha-1}^{i,j} \rightarrow \dots \rightarrow x_{n+\alpha}^{i,j},$$

contradicting that $\{s_{\alpha,\beta}^{i,j}, t_{\alpha,\beta}^{j,i}\}$ is satisfied.

Clique to Directed Symmetric Multicut

Suppose $Z \subseteq V(G)$ induces a complete subgraph in G , and $|Z \cap V^i| = 1$ for all $1 \leq i \leq k$. Define a set of arcs Z' in D by picking diamonds $\diamond_{\alpha}^{i,j}, \diamond_{\alpha+n}^{i,j}, \diamond_{\alpha+2n}^{i,j}$ for all $v_{\alpha}^i \in Z$ and $1 \leq j \leq k$. Clearly, $|Z'| = 3k|Z| = k'$. Further, we show that Z' is a solution to (D, \mathcal{P}, k') . To this end, we prove an auxiliary claim, and show that it implies that every cut request in \mathcal{P} is satisfied in $D - Z'$.

Observe that removing arcs of Z' partitions each necklace into $3k$ strongly connected components of size more than one which we call *runs*. We claim that no two neighbouring runs are strongly connected in $D - Z'$. It suffices to show that for every pair of consecutive runs, there is no closed walk containing vertices from both of them. Note that a counterexample would need to contain a crossing arc because chosen diamonds separate runs within the necklace. By construction, junction vertices x_0, \dots, x_{3n-1} in any string S_j^i are of one of the following types with respect to the crossing arcs:

- x_0, \dots, x_n admit neither incoming nor outgoing crossing arcs,
- x_{n+1}, \dots, x_{2n-1} admit only outgoing crossing arcs,
- x_{2n} admits both incoming and outgoing crossing arcs, and
- $x_{2n+1}, \dots, x_{3n-1}$ admit only incoming crossing arcs.

Moreover, since the construction is periodic, $n + 1$ vertices following x_{3n-1} also admit no incoming or outgoing crossing arcs. Chosen diamonds are evenly spaced, so a run contains n junction vertices. Note that if a run admits both incoming and outgoing crossing arcs, then it contains x_{2n} and one of its neighbouring runs admits only outgoing and another – only incoming crossing arcs. Hence, no pair among these three runs is strongly connected.

With this observation, we first verify that every cut request within a necklace is satisfied by Z' . Indeed, endpoints of cut requests in $D - Z'$ are in neighbouring runs, so they are not strongly connected. Now consider cut requests $\{s_{\alpha,\beta}^{i,j}, t_{\alpha,\beta}^{j,i}\}$ for every pair of non-adjacent vertices v_{α}^i and v_{β}^j in G . Let the junction vertices on strings S_j^i and S_i^j be $x_0^{i,j}, \dots, x_{3n-1}^{i,j}$ and $y_0^{j,i}, \dots, y_{3n-1}^{j,i}$, respectively. Four endpoints of the crossing arcs incident to $s_{\alpha,\beta}^{i,j}$ and $t_{\alpha,\beta}^{j,i}$ are $x_{n+\alpha}^{i,j}, x_{2n+\alpha-1}^{i,j}$ and $y_{n+\beta}^{j,i}, y_{2n+\beta-1}^{j,i}$. We claim that either $x_{n+\alpha}^{i,j}$ and $x_{2n+\alpha-1}^{i,j}$ are not strongly connected or $y_{n+\beta}^{j,i}$ and $y_{2n+\beta-1}^{j,i}$ are not strongly connected. Recall that Z induces a clique in G , and $v_{\alpha}^i, v_{\beta}^j$ are non-adjacent in G , hence either $v_{\alpha}^i \notin Z$ or $v_{\beta}^j \notin Z$. Without loss of generality, assume that $v_{\alpha}^i \notin Z$, and observe that Z contains $v_{\alpha'}^i$, for some $\alpha' \neq \alpha$. Then Z' chooses a diamond that lies between $x_{n+\alpha}^{i,j}$ and $x_{2n+\alpha-1}^{i,j}$, hence these two vertices end up in neighbouring runs and are not strongly connected. The case with $v_{\beta}^j \notin Z$ is symmetric.

5 Concluding remarks

We classified parameterized complexity of WEIGHTED MINCSP(Γ) for all subsets Γ of Point Algebra, i.e. $\{<, \leq, =, \neq\}$ on domain \mathbb{Q} . In particular, we prove that MINCSP(\leq, \neq) is W[1]-hard, settling the complexity of DIRECTED SYMMETRIC MULTICUT. DSMC was a roadblock to classifying interval and temporal constraint languages, i.e. first-order generalizations of Point Algebra and Allen's Interval Algebra, respectively. Tackling these two classifications is a natural continuation. There are several difficulties: for interval constraint languages, even a polynomial-time CSP dichotomy is still unavailable (see [2]), and this is a prerequisite for a MINCSP classification. A more approachable fragment is the set of all binary interval relations for which the dichotomy is known [21].

For temporal constraints, a CSP dichotomy is known [4]. However, if we compare with Boolean and equality languages, the only ones for which parameterized MINCSP dichotomies have been obtained, the temporal CSP classification is much less manageable – the Boolean dichotomy is Schaefer’s celebrated result from 1978 [27], while the equality CSP dichotomy [3] has only one nontrivial tractable class. In contrast, there are ten nontrivial tractable fragments of temporal relations defined by their algebraic invariants [4, Theorem 50].

On the algorithmic side, we want to highlight MINCSP($<, =$). By reduction to SUBSET-DFAS and using the best known algorithm [9], one can solve this problem in $O^*(2^{O(k^3)})$ time. Can this running time be improved? We remark that such an improvement would speed up the 2-approximation algorithm for fixed-parameter tractable MINCSPs over basic interval relations [11].

References

- 1 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.
- 2 Manuel Bodirsky, Peter Jonsson, Barnaby Martin, Antoine Mottet, and Žaneta Semanišinová. Complexity classification transfer for csps via algebraic products, 2022. [arXiv:2211.03340](https://arxiv.org/abs/2211.03340).
- 3 Manuel Bodirsky and Jan Kára. The complexity of equality constraint languages. *Theory of Computing Systems*, 43:136–158, 2008.
- 4 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57:1–41, 2010.
- 5 Édouard Bonnet, László Egri, Bingkai Lin, and Dániel Marx. Fixed-parameter approximability of Boolean MinCSPs, 2016. [arXiv:1601.04935](https://arxiv.org/abs/1601.04935).
- 6 Édouard Bonnet, László Egri, and Dániel Marx. Fixed-parameter approximability of Boolean MinCSPs. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA 2016)*, pages 18:1–18:18, 2016.
- 7 Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. *SIAM Journal on Computing*, 47:166–207, 2018.
- 8 Jianer Chen, Yang Liu, Songjian Lu, Barry O’sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5), 2008.
- 9 Rajesh Chitnis, Marek Cygan, Mohammadtaghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11:1–28, 2015.
- 10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 11 Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, Marcin Pilipczuk, and Roohani Sharma. Parameterized complexity classification for interval constraints. In *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*, pages 11:1–11:19, 2023.
- 12 Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, and Magnus Wahlström. Almost consistent systems of linear equations. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 3179–3217, 2023.
- 13 Eduard Eiben, Clément Rambaud, and Magnus Wahlström. On the parameterized complexity of symmetric directed multicut. In *Proceedings of the 17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, pages 11:1–11:17, 2022.
- 14 Venkatesan Guruswami and Euiwoong Lee. Simple proof of hardness of feedback vertex set. *Theory of Computing*, 12:1–11, 2016.
- 15 Yoichi Iwata and Yuichi Yoshida. Exact and approximation algorithms for the maximum constraint satisfaction problem over the point algebra. In *Proc. 30th International Symposium on Theoretical Aspects of Computer Science, (STACS 2013)*, pages 127–138, 2013.

- 16 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 767–775, 2002.
- 17 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Solving hard cut problems via flow-augmentation. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, pages 149–168, 2021.
- 18 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC 2022)*, pages 938–947, 2022.
- 19 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation III: Complexity dichotomy for Boolean CSPs parameterized by the number of unsatisfied constraints. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, pages 3218–3228, 2023.
- 20 Eun Jung Kim, Tomáš Masařík, Marcin Pilipczuk, Roohani Sharma, and Magnus Wahlström. On weighted graph separation problems and flow augmentation. *SIAM Journal on Discrete Mathematics*, 38:170–189, 2024.
- 21 Andrei A. Krokhin, Peter Jeavons, and Peter Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen’s interval algebra. *Journal of the ACM*, 50:591–640, 2003.
- 22 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351:394–406, 2006.
- 23 Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM Journal on Computing*, 43:355–388, 2014.
- 24 George Osipov and Magnus Wahlström. Parameterized complexity of Equality MinCSP. In *Proceedings of the 31st Annual European Symposium on Algorithms (ESA 2023)*, pages 86:1–86:17, 2023.
- 25 Marcin Pilipczuk and Magnus Wahlström. Directed multicut is $W[1]$ -hard, even for four terminal pairs. *ACM Transactions on Computation Theory*, 10:1–18, 2018.
- 26 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32:299–301, 2004.
- 27 Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM Symposium on Theory of computing (STOC 1978)*, pages 216–226, 1978.
- 28 Ola Svensson. Hardness of vertex deletion and project scheduling. In *International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2012)*, pages 301–312, 2012.
- 29 Marc Vilain, Henry Kautz, and Peter Van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In *Readings in qualitative reasoning about physical systems*, pages 373–381. Elsevier, 1990.
- 30 Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI 1986)*, pages 377–382, 1986.