Binary constraint satisfaction problems defined by excluded topological minors ☆

David A. Cohen [a], Martin C. Cooper [b], Peter G. Jeavons [c], Stanislav Živný [c,*]

[a] *Royal Holloway, University of London, UK*
[b] *IRIT, University of Toulouse, France*
[c] *University of Oxford, Oxford, UK*

Keywords:
Constraint satisfaction
Forbidden substructures
Forbidden patterns
Topological minors

A B S T R A C T

The binary Constraint Satisfaction Problem (CSP) is to decide whether there exists an assignment to a set of variables which satisfies specified constraints between pairs of variables. A binary CSP instance can be presented as a labelled graph encoding both the forms of the constraints and where they are imposed. We consider subproblems defined by restricting the allowed form of this graph. One type of restriction is to forbid certain specified substructures (patterns). This captures some tractable classes of the CSP, but does not capture classes defined by language restrictions, or the well-known structural property of acyclicity.

We extend the notion of pattern and introduce the notion of a topological minor of a binary CSP instance. By forbidding a *finite* set of patterns from occurring as topological minors we obtain a compact mechanism for expressing novel tractable subproblems of the CSP, including new generalisations of the class of acyclic instances.

1. Introduction

The Constraint Satisfaction Problem (CSP) is to decide whether it is possible to find an assignment to a set of variables which satisfies constraints between certain subsets of the variables. This paradigm has been applied in diverse application areas such as Artificial Intelligence, Bioinformatics and Operations Research [40,30].

As the CSP is known to be NP-complete, much theoretical work has been devoted to the identification of tractable subproblems. Important tractable cases have been identified by restricting the hypergraph *structure* of the constrained subsets of variables [26,17]. Other tractable cases have been identified by restricting the forms of constraints (sometimes called the constraint *language*) [32,24]. Work on both of these areas is now essentially complete: full complexity classifications have been established for all structural restrictions [28,37] and all language restrictions [4,43].

☆ An extended abstract of part of this work appeared in the *Proceedings of the 24th International Joint Conference on Artificial Intelligence* (IJCAI'15) [9]. The authors were supported by EPSRC grant EP/L021226/1. Stanislav Živný was supported by a Royal Society University Research Fellowship. 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. 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

* Corresponding author.

E-mail addresses: dave@cs.rhul.ac.uk (D.A. Cohen), cooper@irit.fr (M.C. Cooper), peter.jeavons@cs.ox.ac.uk (P.G. Jeavons), standa.zivny@cs.ox.ac.uk (S. Živný).

https://doi.org/10.1016/j.ic.2018.09.013

However, identifying the subproblems of the CSP that can be obtained by restricting either the structure or the language alone is not a sufficiently rich framework in which to investigate the full complexity landscape. For example, we may wish to identify all the instances solved by a particular algorithm, such as enforcing arc-consistency [18,40]. It has been shown [24,10] that this class of instances includes all instances defined by a certain structural restriction, together with all instances defined by a certain language restriction, as well as further instances that are not defined by either kind of restriction alone. Hence we need a more flexible mechanism for describing subproblems that will allow us to unify and generalise such descriptions.

Here we develop a new mechanism of this kind that uses certain tools from graph theory to define restricted classes of labelled graphs that represent binary CSP instances. Our mechanism allows us to impose simultaneous restrictions on both the structure and the language of an instance, and hence obtain a more refined collection of subproblems, allowing a more detailed complexity analysis. Subproblems of the CSP of this kind are sometimes referred to as *hybrid* subproblems and, currently, very little is known about the complexity of such subproblems [15].

The tools that we use to obtain restricted classes of labelled graphs build on a well-established line of research in graph theory, by considering local "obstructions" or "forbidden patterns". The idea of using forbidden patterns has previously been applied to the binary CSP and resulted in the discovery of a number of new tractable classes [7,8,12,23]; related ideas also appeared in [36,34]. In more detail, [7] characterised all so-called negative patterns that give rise to tractable classes of binary CSPs (this result is summarised in Theorem 3.12 below). Moreover, [12] characterised all patterns consisting of at most two constraints that give rise to tractable classes of binary CSPs. Finally, [8] investigated the notion of forbidden patterns in the context of variable and value elimination in CSPs.

However, the existing theory of forbidden patterns is not sufficient to capture all known tractable structural restrictions, or language restrictions, as we show below. In particular, we show that even the simplest tractable structural class, the class of tree-structures CSP instances, cannot be captured by forbidding any finite set of patterns (Corollary 4.4). To describe all the relevant structural, language and hybrid restrictions that can ensure tractability therefore requires a more flexible way to define restricted classes of instances.

In graph theory it proved useful to go beyond the idea of forbidden subgraphs and introduce the more flexible concept of forbidden minors. A well-known result of Robertson and Seymour states that any set of graphs closed under the operation of taking minors is specified by a finite set of forbidden minors. Rather than adapting the full machinery of graph minors to the CSP framework, we consider here the slightly simpler notion of a *topological minor* [20]. We show that by adapting the notion of topological minor to the CSP framework we are able to provide a unified description of all tractable structural classes, all tractable language classes, and some new hybrid tractable classes that cannot be captured as either structural classes or language classes. Moreover, we are able to show that the class of tree-structured CSP instances has a very simple description in this framework, and there exist tractable classes of the binary CSP that properly extend this class and yet still have a very simple description.

An extended abstract of part of this work appeared in [9].

The structure of the paper is as follows: in Section 2 we define the CSP and the notion of a pattern, and show how to associate each CSP instance with a corresponding pattern. In Section 3 we define what it means for a pattern to occur in another pattern, either as a *sub-pattern* or as a *topological minor*, and use these notions to define restricted classes of CSP instances where specified patterns are forbidden from occurring in one or other of these ways.

In Section 4 we show that all tractable structural classes of the CSP can be characterised by forbidding certain patterns from occurring as topological minors. We extend this idea in Section 5 to obtain novel hybrid tractable classes of CSP instances, including classes that properly extend the class of acyclic instances.

In Section 6 we consider the complexity of determining whether a given pattern occurs as a topological minor in a CSP instance, and in Section 7 we show that including additional structure in patterns allows us to characterise more classes of CSP instances, including all tractable language classes. Finally, in Section 8, we conclude with a discussion of our results and present some open questions.

2. Preliminaries

2.1. The CSP

Constraint satisfaction is a paradigm for describing computational problems. Each problem instance is represented as a constraint network: a collection of variables that take their value from some given domain. Some subsets of the variables have a further restriction on their allowed simultaneous assignments, called a constraint. A solution to such a network assigns a value to each variable such that every constraint is satisfied.

In this paper we consider only *binary* constraint networks, where every constraint limits the possible assignments of precisely two variables. It has been shown that any constraint network can be reduced to an equivalent binary network over a different domain of values [19,41].

Definition 2.1. An instance of the binary constraint satisfaction problem (CSP) is a triple (V, D, C) where V is a finite set of variables, for each $v \in V$, $D(v)$ is a finite domain of values for v, and C is a set of constraints, containing a constraint R_{uv}

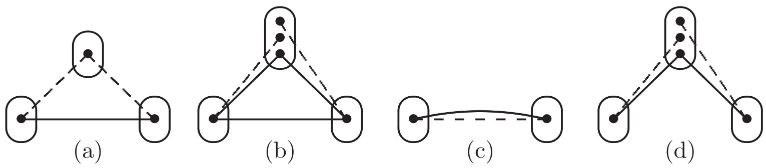

Fig. 1. Some example patterns. Points are shown as filled circles, parts as ovals, positive edges as solid lines and negative edges as dashed lines.

for each pair of variables (u, v). The constraint $R_{uv} \subseteq D(u) \times D(v)$ is the set of compatible assignments to the variables u and v.

A *solution* to a binary CSP instance is an assignment $s : V \to D$ of values to variables such that, for each constraint R_{uv}, $(s(u), s(v)) \in R_{uv}$.

We will assume that there is *exactly one* binary constraint between any two variables. That is, if we define R'_{uv} as $\{(b, a) \mid (a, b) \in R_{uv}\}$, then $R_{vu} = R'_{uv}$. This is just a notational convenience since we can pre-process each instance, replacing R_{uv} with $R_{uv} \cap R'_{vu}$. A constraint will be called *trivial* if it is equal to the Cartesian product of the domains of its two variables.

The size of a CSP instance will be taken to be the sum of the sizes of the constraint relations. Given a fixed bound on the size of the domain for any variable and the arity of the constraints, this is polynomial in the number of variables. We will say that a class of CSP instances is *tractable* if there is a polynomial-time algorithm to decide whether any instance in the class has a solution.

Note that Definition 2.1 describes a standard form of mathematical specification for a CSP instance that is convenient for theoretical analysis. In the next subsection we will introduce an alternative representation in terms of patterns (see Construction 2.5). Often more concise representations are used, and trivial constraints are usually not represented [40].

Arc-consistency (AC) is a fundamental concept for the binary CSP [18,40].

Definition 2.2. A pair of variables (u, v) is said to be arc-consistent if for each value $a \in D(u)$ in the domain of u, there is a value $b \in D(v)$ in the domain of v such that $(a, b) \in R_{uv}$.

A binary CSP instance is *arc-consistent* if every pair of variables is arc-consistent.

Given an arbitrary CSP instance I there is a unique minimal set of domain values which can be removed to make the instance arc-consistent. Furthermore the discovery of this unique minimal set of domain values and their removal, called establishing arc-consistency, can be done in polynomial time [11]. For a given instance I we will denote by $\mathrm{AC}(I)$ the instance obtained after establishing arc-consistency.

2.2. Patterns

We now introduce the central notion of a *pattern*, which can be thought of as a labelled graph, with three distinct kinds of edges.

Definition 2.3. A *pattern* is a structure $(X, E^{\sim}, E^{+}, E^{-})$, where

- X is a set of *points*;
- $E^{\sim}$ is a binary equivalence relation over X whose equivalence classes are called *parts*;
- E^{+} is a symmetric binary relation over X whose tuples are called *positive edges*;
- E^{-} is a symmetric binary relation over X whose tuples are called *negative edges*.

The sets $E^{\sim}$ and E^{+} are disjoint, and the sets $E^{\sim}$ and E^{-} are disjoint.

In a general pattern there may be pairs of points x and y in distinct parts such that (x, y) is neither a positive nor a negative edge, and there may be pairs of points x and y in distinct parts such that (x, y) is *both* a positive and a negative edge. A pattern is called *complete* if every pair of points x and y in distinct parts are connected by either a positive or negative edge (but not both), and hence $E^{\sim} \cup E^{+} \cup E^{-} = X^2$.

Example 2.4. Some examples of patterns are illustrated in a standard way in Fig. 1.

The pattern shown in Fig. 1(a) is complete, but the others are not. □

It will often be convenient to build special patterns to represent binary CSP instances, so we now define the following construction.

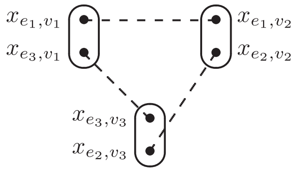

Fig. 2. The pattern Patt(C_3) constructed from the cycle graph C_3 by Construction 2.7.

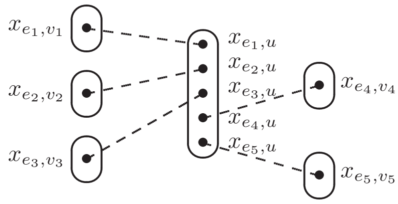

Fig. 3. The pattern Patt$(K_{1,5})$ constructed from the star graph $K_{1,5}$ by Construction 2.7.

Construction 2.5. *For any binary CSP instance $I = (V, D, C)$, where $C = \{R_{uv} \mid u, v \in V, u \neq v\}$, we define a corresponding complete pattern* Patt$(I) = (X, E^{\sim}, E^{+}, E^{-})$ *where*

- $X = \{x_{v,a} \mid v \in V, a \in D(v)\}$;
- $E^{\sim} = \{(x_{u,a}, x_{v,b}) \in X \times X \mid u = v\}$;
- $E^{+} = \{(x_{u,a}, x_{v,b}) \in X \times X \mid u \neq v, (a, b) \in R_{uv}\}$;
- $E^{-} = \{(x_{u,a}, x_{v,b}) \in X \times X \mid u \neq v, (a, b) \notin R_{uv}\}$.

We remark that for any instance I the points of Patt(I) are the possible assignments for each individual variable, and the parts of Patt(I) correspond to sets of possible assignments for a particular variable. Positive edges in Patt(I) correspond to allowed pairs of assignments and are therefore closely related to the edges of the *microstructure* representation of I defined in [33]; negative edges correspond to disallowed pairs of assignments and are closely related to the edges of the *microstructure complement* discussed in [6].

Example 2.6. Fig. 1(a) shows the pattern Patt(I) for a rather trivial instance I with three variables, each of which has only one possible value. Note that I has no solution because the only possible assignments for two pairs of variables are in negative edges and hence disallowed by the constraints. □

A pattern with no positive edges will be called a *negative pattern*. It will sometimes be convenient to build negative patterns from graphs, so we now define the following construction.

Construction 2.7. *For any graph $G = (V, E)$, we define a corresponding negative pattern* Patt$(G) = (X, E^{\sim}, \emptyset, E^{-})$ *where*

- $X = \{x_{e,v} \mid e \in E, v \in e\}$;
- $E^{\sim} = \{(x_{e,u}, x_{f,v}) \in X \times X \mid u = v\}$;
- $E^{-} = \{(x_{e,u}, x_{f,v}) \in X \times X \mid e = f, u \neq v\}$.

Example 2.8. Let C_3 be the 3-cycle, that is, the graph with three vertices, v_1, v_2, v_3, and 3 edges e_1, e_2, e_3, where $e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_3\}$ and $e_3 = \{v_3, v_1\}$. The associated negative pattern Patt(C_3) defined by Construction 2.7 is the pattern with 6 points, 3 parts, and 3 negative edges, shown in Fig. 2.

Let $K_{1,k}$ be a star graph with k leaves; that is, the graph with vertices $\{u, v_1, \ldots, v_k\}$ and edges $\{u, v_i\}$ for $1 \leq i \leq k$. The pattern Patt$(K_{1,k})$ has $2k$ points, $k+1$ parts, and k negative edges. The case of $k = 5$ is shown in Fig. 3. □

In graph theory, a *subdivision* operation on a graph replaces an edge (u, v) with a path of length two by introducing a new vertex z_{uv}, and connecting u to z_{uv} and z_{uv} to v [20]. A graph G is said to be a topological minor of a graph H if some sequence of subdivision operations on G yields a subgraph of H [20]. We now define an operation on patterns that is analogous to the subdivision operation on graphs, but takes into account the three different types of edges that are present in a pattern. This subdivision operation for patterns is crucial to the idea of defining topological minors in patterns, as described in Section 3.

Definition 2.9. Let $P = (X, E^{\sim}, E^{+}, E^{-})$ be a pattern.

For any two distinct parts U, V of P, we define $E^+_{UV} = E^+ \cap (U \times V)$, $E^-_{UV} = E^- \cap (U \times V)$, and $Z_{UV} = \{z_{xy} \mid (x, y) \in E^+_{UV}\} \cup \{z'_{xy}, z''_{xy} \mid (x, y) \in E^-_{UV}\}$. The *subdivision* of P at U, V is defined to be the pattern $P_d = (X_d, E^{\sim}_d, E^+_d, E^-_d)$ where

- $X_d = X \cup Z_{UV}$;
- $E^{\sim}_d = E^{\sim} \cup (Z_{UV} \times Z_{UV})$;
- $E^+_d = (E^+ \backslash \{(x, y), (y, x) \mid (x, y) \in E^+_{UV}\}) \cup \{(x, z_{xy}), (z_{xy}, x), (z_{xy}, y), (y, z_{xy}) \mid (x, y) \in E^+_{UV}\}$;
- $E^-_d = (E^- \backslash \{(x, y), (y, x) \mid (x, y) \in E^-_{UV}\}) \cup \{(x, z'_{xy}), (z'_{xy}, x), (z''_{xy}, y), (y, z''_{xy}) \mid (x, y) \in E^-_{UV}\}$.

Pattern P' is called a *subdivision* of P if it can be obtained from P by some (possibly empty) sequence of subdivision operations.

Example 2.10. The pattern shown in Fig. 1(d) can be obtained by performing a single subdivision operation on the pattern shown in Fig. 1(c). □

We remark that positive and negative edges are treated differently in Definition 2.9: a single extra point, z_{xy}, is added for each positive edge (x, y), and *two* extra points, z'_{xy} and z''_{xy}, are added for each negative edge (see Example 2.10). This difference reflects a semantic difference between positive and negative edges in a CSP instance, which we illustrate as follows. Suppose that the assignment of value a to variable u and value b to variable v extends to a solution. In this case, for any other variable w, the points $\langle u, a\rangle$ and $\langle v, b\rangle$ must both be compatible with some common point $\langle w, c\rangle$. On the other hand, the assignment of a to variable u and b to variable v may fail to extend to a solution if there are points $\langle w, c\rangle$ and $\langle w, d\rangle$ where $\langle u, a\rangle$ is incompatible with $\langle w, c\rangle$, $\langle v, b\rangle$ is incompatible with $\langle w, d\rangle$ and the rest of the instance forces w to take either value c or value d.

3. Forbidding patterns

In the remainder of this paper we consider classes of binary CSP instances that are defined by *forbidding* a specified set of patterns from occurring in certain ways, which we now define.

3.1. Occurrences of one pattern in another

Definition 3.1. A pattern $P_1 = (X_1, E^{\sim}_1, E^+_1, E^-_1)$ is said to have a *homomorphism* to a pattern $P_2 = (X_2, E^{\sim}_2, E^+_2, E^-_2)$, if there is a mapping $h : X_1 \to X_2$ such that

- if $(x, y) \in E^{\sim}_1$ then $(h(x), h(y)) \in E^{\sim}_2$, and
- if $(x, y) \in E^+_1$ then $(h(x), h(y)) \in E^+_2$, and
- if $(x, y) \in E^-_1$ then $(h(x), h(y)) \in E^-_2$.

A homomorphism h from a pattern $P_1 = (X_1, E^{\sim}_1, E^+_1, E^-_1)$ to a pattern $P_2 = (X_2, E^{\sim}_2, E^+_2, E^-_2)$ will be said to *preserve parts* if it satisfies the additional property that for all $(x, y) \in X_1^2$, if $(x, y) \notin E^{\sim}_1$, then $(h(x), h(y)) \notin E^{\sim}_2$.

Definition 3.2. A pattern P_1 is said to *occur as a sub-pattern* in a pattern P_2, denoted $P_1 \xrightarrow{SP} P_2$, if there is a homomorphism from P_1 to P_2 that preserves parts.

Earlier papers [7,12] have defined the notions of pattern and the notion of occurring as a sub-pattern in slightly different ways, but these are all essentially equivalent to Definition 3.2.

Example 3.3. The pattern shown in Fig. 1(d) has a homomorphism to the pattern shown in Fig. 1(c), but does not occur as a sub-pattern in this pattern. The pattern shown in Fig. 1(d) does occur as a sub-pattern in the pattern shown in Fig. 1(b). □

Now we introduce a new form of occurrence that will be our focus in this paper, and will allow us to define a wider range of restricted subproblems of the CSP.

Definition 3.4. A pattern P_1 is said to *occur as a topological minor* in a pattern P_2, denoted $P_1 \xrightarrow{TM} P_2$, if some subdivision of P_1 occurs as a sub-pattern in P_2.

Example 3.5. The pattern shown in Fig. 1(c) occurs as a topological minor in the pattern shown in Fig. 1(d) and in the pattern shown in Fig. 1(b). □

Lemma 3.6. *For any patterns P, P' and P'' the following properties hold:*

(a) $P \overset{SP}{\to} P$ *and* $P \overset{TM}{\to} P$;
(b) *If* $P \overset{SP}{\to} P'$, *then* $P \overset{TM}{\to} P'$;
(c) *If* $P \overset{SP}{\to} P'$ *and* $P' \overset{SP}{\to} P''$, *then* $P \overset{SP}{\to} P''$;
(d) *If* $P \overset{TM}{\to} P'$ *and* $P' \overset{TM}{\to} P''$, *then* $P \overset{TM}{\to} P''$.

Proof. Part (a) is obtained by taking the identity function as a homomorphism, and an empty sequence of subdivisions. Part (b) is obtained by taking an empty sequence of subdivisions. Part (c) is obtained by composing the two homomorphisms.

Part (d) follows from the following observation: assume that h is a homomorphism from P_1 to P_2 that preserves parts, and that P_3 is the pattern obtained by performing a subdivision operation on P_2 at parts U and V. Now consider the pattern Q obtained by performing a subdivision operation on P_1 at the parts that are mapped by h to U and V. By our definition of subdivision, it follows that h can be extended to a homomorphism h' from Q to P_3 that preserves parts.

Hence in any sequence of subdivision operations and homomorphisms that preserve parts we can re-order the operations to perform all subdivisions at the start, and then compose all the homomorphisms. □

Recall that establishing arc-consistency in an instance I involves removing domain values from I and yields the (unique) instance $\mathrm{AC}(I)$, hence it cannot introduce an occurrence of a pattern as a sub-pattern or as a topological minor if it did not already occur. This gives the following result.

Lemma 3.7. *For any patterns P and I, where I represents an instance, the following properties hold:*

(a) *If* $P \overset{SP}{\to} \mathrm{Patt}(\mathrm{AC}(I))$, *then* $P \overset{SP}{\to} \mathrm{Patt}(I)$;
(b) *If* $P \overset{TM}{\to} \mathrm{Patt}(\mathrm{AC}(I))$, *then* $P \overset{TM}{\to} \mathrm{Patt}(I)$.

Establishing arc-consistency can be done in polynomial time, so for many of our results we will only need to consider arc-consistent CSP instances.

3.2. Restricted classes of instances

We can use Definition 3.2 to define restricted classes of binary CSP instances by forbidding the occurrence of certain patterns as sub-patterns in those instances.

Definition 3.8. Let $\mathcal{S}$ be a set of patterns.

We denote by $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathcal{S})$ the set of all binary CSP instances I such that for all $P \in \mathcal{S}$ it is not the case that $P \overset{SP}{\to} \mathrm{Patt}(I)$.

Definition 3.9. We will say that a pattern P is *sub-pattern tractable* if $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\{P\})$ is tractable; we will say that a pattern P is *sub-pattern* NP*-complete* if $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\{P\})$ is NP-complete.

For simplicity, we write $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ for $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\{P\})$.

The complexity of the class $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathcal{S})$ has been determined for a wide range of patterns [13,7,12]. In fact, for all *negative* patterns P the complexity of $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ has been completely characterised [7]. To define this characterisation, we need to introduce the idea of *star patterns*.

A connected graph G is called a *star* if it is acyclic, and has exactly one vertex of degree greater than 2. The vertex of degree greater than 2 in a star graph will be called the central vertex. A pattern P will be called a *star pattern* if it can be obtained from the pattern $\mathrm{Patt}(G)$ for some star graph G by merging zero or more points in the part of $\mathrm{Patt}(G)$ corresponding to the central vertex of G.

Example 3.10. Since the empty graph is a star graph, the simplest star pattern is the empty pattern, which has no points. Some other examples of star patterns are shown in Fig. 4. □

Definition 3.11 *([7]).* For any $k \geq 1$, the star pattern with 3 branches, each of length k, where exactly two points are merged in the central part, as shown in Fig. 5, is called $\mathrm{Pivot}(k)$.

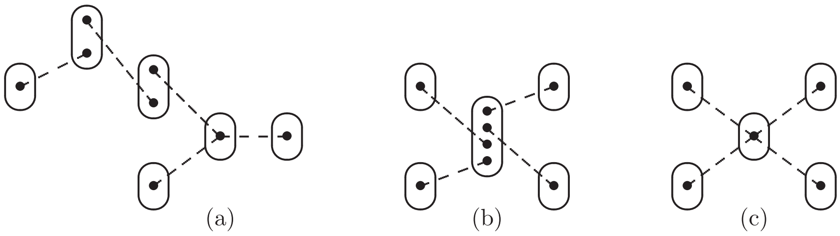

Fig. 4. Examples of star patterns.

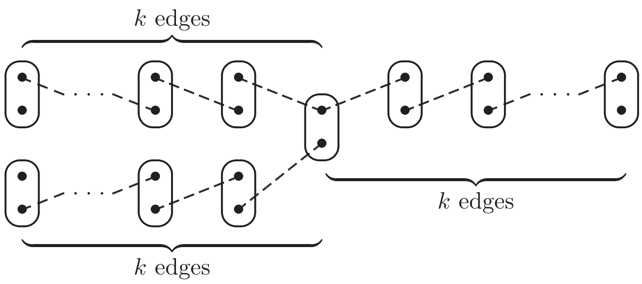

Fig. 5. The pattern Pivot(k).

Theorem 3.12 *([7]). For any $k \geq 1$, the negative pattern Pivot(k) shown in Fig. 5 is sub-pattern tractable, as are all negative patterns P such that $P \overset{SP}{\to} Pivot(k)$; all other negative patterns are sub-pattern* NP*-complete.*

Example 3.13. By Theorem 3.12 all the negative patterns shown in Figs. 2, 3 and 4 are sub-pattern NP-complete. □

To go beyond the earlier results for forbidden sub-patterns [7,8,12,23], and define a wider range of restricted classes, we use Definition 3.4 to define restricted classes of binary CSP instances by forbidding the occurrence of certain patterns as topological minors in those instances.

Definition 3.14. Let $\mathcal{S}$ be a set of patterns.

We denote by $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\mathcal{S})$ the set of all binary CSP instances I such that for all $P \in \mathcal{S}$ it is not the case that $P \overset{TM}{\to} \mathrm{Patt}(I)$.

Definition 3.15. We will say that a pattern P is *topological-minor tractable* if $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\{P\})$ is tractable; we will say that a pattern P is *topological-minor* NP*-complete* if $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\{P\})$ is NP-complete.

For simplicity, we write $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ for $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\{P\})$.

By Lemma 3.6(b), if P occurs as a sub-pattern of some pattern Q, then it also occurs as a topological minor of Q. Hence for any pattern P we have that $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P) \subseteq \mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$. The following is an immediate consequence.

Lemma 3.16. *If a pattern P is sub-pattern tractable then P is also topological-minor tractable.*

Example 3.17. By the results of earlier work, the two patterns shown in Fig. 1(a) and 1(b) are known to be sub-pattern tractable: the tractability of the pattern shown in Fig. 1(a) follows from the tractability of a more general pattern (called JWP) defined in [14]; the tractability of the pattern shown in Fig. 1(b) follows from [23, Lemma 46] (where it corresponds to pattern U'_{30}).

Hence both patterns are also topological-minor tractable, by Lemma 3.16. □

By Lemma 3.6(d), if P occurs as a topological minor in Q then $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P) \subseteq \mathrm{CSP}_{\overline{\mathrm{TM}}}(Q)$. The following is an immediate consequence.

Lemma 3.18. *If pattern $P \overset{TM}{\to} Q$, and Q is topological-minor tractable, then P is also topological-minor tractable.*

Example 3.19. We can deduce from Lemma 3.18 that Fig. 1(d) is topological-minor tractable, since Fig. 1(d) occurs as a sub-pattern (and hence also as a topological minor) in Fig. 1(b), and it was shown in Example 3.17 that Fig. 1(b) is topological-minor tractable. □

The converse of Lemma 3.16 does not hold: there exist patterns that are topological-minor tractable but sub-pattern NP-complete, as the following example demonstrates. More significant examples will be discussed in Section 5.

Example 3.20. Fig. 1(c) is sub-pattern NP-complete, since it cannot occur as a sub-pattern of any instance, so for this pattern P, $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ contains all possible CSP instances. However, by Lemma 3.18, Fig. 1(c) is topological-minor tractable, since it occurs as a topological minor in Fig. 1(d), and it was shown in Example 3.19 that Fig. 1(d) is topological-minor tractable. $\square$

For some patterns P, the sets $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$ and $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ are identical, as our next result shows. A pattern P will be called *star-like* if removing the positive edges from P gives a negative pattern P' such that $P' \overset{SP}{\to} P''$ for some star pattern P''.

Example 3.21. All of the patterns shown in Figs. 1, 3 and 4 are star-like, but the pattern shown in Fig. 2 is not star-like. $\square$

Proposition 3.22. *If P is a star-like negative pattern, then* $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P) = \mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$.

Proof. By Lemma 3.6(b), for any pattern P we have that $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P) \subseteq \mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$.

To obtain the reverse inclusion, let P be a star-like negative pattern, and let Q be a star pattern such that $P \overset{SP}{\to} Q$. By the definition of star pattern, for any subdivision Q' of Q, we have that $Q \overset{SP}{\to} Q'$. Hence, by Lemma 3.6(c) $P \overset{SP}{\to} Q'$, so $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P) \subseteq \mathrm{CSP}_{\overline{\mathrm{SP}}}(Q')$. But this implies, by Definition 3.4, that $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P) \subseteq \mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$. $\square$

Example 3.23. By Theorem 3.12, any pattern Pivot(k) is sub-pattern tractable, and by Proposition 3.22 we know that forbidding Pivot(k) as a topological minor defines the same set of instances as forbidding Pivot(k) as a sub-pattern. Therefore, for any $k \geq 1$, the pattern Pivot(k) is also topological-minor tractable.

Similarly, by Theorem 3.12, each star pattern P shown in Fig. 4 is sub-pattern NP-complete. By Proposition 3.22, for each of these patterns $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P) = \mathrm{CSP}_{\overline{\mathrm{SP}}}(P)$. Consequently, these patterns are also topological-minor NP-complete. $\square$

We now give a partial converse of Proposition 3.22, by showing that for all patterns P that are not star-like, $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ cannot be expressed by forbidding any finite set of sub-patterns. This means that the notion of forbidding the occurrence of a pattern as a topological minor provides more expressive power than forbidding arbitrary (finite) sets of patterns from occurring as sub-patterns.

Proposition 3.24. *If P is a pattern that is not star-like, then* $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P) \neq \mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathcal{S})$ *for all finite sets of patterns $\mathcal{S}$.*

Proof. Let P be a pattern that is not star-like, and let P' be the negative pattern obtained by removing all positive edges of P. Note that $P' \overset{SP}{\to} P$.

In any pattern, say that a part U is *distinguished* if two negative edges share a single point in U or if there are negative edges from U to more than two other parts.

Since P is not star-like, the negative pattern P' must contains a cycle of parts connected by negative edges, or at least two distinguished parts.

Hence, for any fixed k, by a sufficiently long sequence of subdivision operations, we can construct a subdivision P'' of P' which either has a cycle of parts of length greater than k or two distinguished parts separated by a sequence of connected parts of length greater than k. By adding positive edges, we can then convert P'' into a complete pattern of the form Patt(I) for some CSP instance I.

Now for any fixed finite set of patterns $\mathcal{S}$ there will be a bound k on the number of parts of any pattern in $\mathcal{S}$. It follows that $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ cannot be defined by forbidding the sub-patterns in $\mathcal{S}$, since $I \notin \mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ but no pattern in $\mathcal{S}$ can occur as a sub-pattern in Patt(I). $\square$

4. Structural restrictions

For any CSP instance $I = (V, D, C)$, the *constraint graph* of I is defined to be the graph (V, E), where E is the set of pairs $\{x, y\}$ for which the associated constraint R_{xy} is non-trivial. A number of tractable subproblems of the CSP have been defined by specifying restrictions on the constraint graph; such restricted classes of instances are known as *structural classes* [28,37].

It is known that a structural class of binary CSP instances defined in this way is tractable if and only if every instance has a constraint graph of bounded treewidth [28, Theorem 5.1] (subject to the standard complexity-theoretic assumption that FPT $\neq$ W[1], which we will assume throughout this section; we refer the reader to the textbooks [22,25] for more details). We show in this section that structural classes of this kind cannot be defined by forbidding the occurrence of a finite set of sub-patterns. However, they *can* be defined by forbidding the occurrence of one or more patterns as topological minors.

We will also use this characterisation of tractable structural classes to show that a large class of negative patterns are topological minor tractable.

First we extend the notion of a constraint graph to arbitrary patterns.

Definition 4.1. For any pattern P, the *constraint graph* of P, denoted G_P, is defined to be the graph (V, E), where V is the set of all parts of P, and E is the set of pairs of parts $\{U, W\}$ such that there is a negative edge $(x, y) \in P$ with $x \in U$ and $y \in W$.

For any binary CSP instance I, the constraint graph of I defined above is equal to $G_{\mathrm{Patt}(I)}$. For simplicity, this graph will usually be denoted by G_I.

Now we note the close link between our notion of a pattern occurring as a topological minor of another pattern and the standard notion of a topological minor in a graph [20].

Lemma 4.2. *For any graph G and any pattern P, $\mathrm{Patt}(G) \overset{TM}{\to} P$ if and only if G is a topological minor of the graph G_P.*

The simplest structural class of CSP instances of bounded treewidth is the class of instances whose constraint graph is *acyclic* (that is, has treewidth 1). This class is known as the class of acyclic binary CSP instances and was one of the first sub-problems of the CSP to be shown to be tractable [26]. We now show that this class can be characterised very simply by excluding the single pattern $\mathrm{Patt}(C_3)$ shown in Fig. 2 from occurring as a topological minor.

Proposition 4.3. *The class of acyclic binary CSP instances equals $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\mathrm{Patt}(C_3))$.*

Proof. The class of acyclic graphs may be characterised as graphs which do not contain C_3 as a topological minor [20]. Hence, by Lemma 4.2 and Definition 4.1, a binary CSP instance I has an acyclic constraint graph if and only if it is not the case that $\mathrm{Patt}(C_3) \overset{TM}{\to} \mathrm{Patt}(I)$. □

Since the pattern $\mathrm{Patt}(C_3)$ is not star-like (see Example 3.21), it follows immediately from Proposition 3.24 that acyclic CSP instances cannot be defined by any finite set of forbidden sub-patterns.

Corollary 4.4. *The class of acyclic binary CSP instances is not equal to $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathcal{S})$ for any finite set of patterns $\mathcal{S}$.*

Proposition 4.3 can easily be extended to any of the tractable structural classes of binary CSP instances defined by imposing any fixed bound on the treewidth of the constraint graph [27], although in this case the set of forbidden patterns is explicitly known only for $k \le 3$ [1].

Theorem 4.5. *For any fixed $k \ge 1$, the class of binary CSP instances whose constraint graph has treewidth at most k equals $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\mathcal{S}_k)$, for some finite set of patterns $\mathcal{S}_k$.*

Proof. The graph minor theorem [39] implies that for any fixed $k \ge 1$ there is a finite set O_k of graphs such that the class of graphs of treewidth at most k is precisely the class of graphs excluding all graphs from the set O_k as topological minors [20]. (More precisely, the graph minor theorem gives a finite set of minors as obstructions but this set can be turned into a finite set of topological minors as obstructions in a standard way, see [20, Exercise 34, Chapter 12].) Consequently, by Lemma 4.2, for any $k \ge 1$ the class of binary CSP instances with constraint graphs of treewidth at most k can be defined as $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\mathcal{S}_k)$ for the finite set of negative patterns $\mathcal{S}_k$ given by $\mathcal{S}_k = \{\mathrm{Patt}(G) \mid G \in O_k\}$. □

In fact, we are able to show that many other patterns are topological-minor tractable using other standard results from graph theory. The following theorem characterises the topological-minor tractability of patterns of the form $\mathrm{Patt}(G)$, for all graphs G of maximum degree three.

Theorem 4.6. *Let G be an arbitrary graph of maximum degree three. Then, $\mathrm{Patt}(G)$ is topological-minor tractable if and only if G is planar (assuming $\mathrm{FPT} \ne \mathrm{W}[1]$).*

Proof. One of the well-known results of Robertson and Seymour shows that the class of graphs obtained by excluding G as a minor has bounded treewidth if and only if G is planar [38] (see also [20, Theorem 12.4.3]). It is known that for a graph G of maximum degree three and any graph G', G is a minor of G' if and only if G is a topological minor of G' [20, Proposition 1.7.4 (ii)]. Thus, for a graph G of maximum degree three, the class of graphs obtained by excluding G as a topological minor has bounded treewidth if and only if G is planar. The theorem then follows from Lemma 4.2 and the fact that, assuming $\mathrm{FPT} \ne \mathrm{W}[1]$, a structural class of binary CSP instances is tractable if and only if the associated class of constraint graphs is of bounded treewidth [28]. □

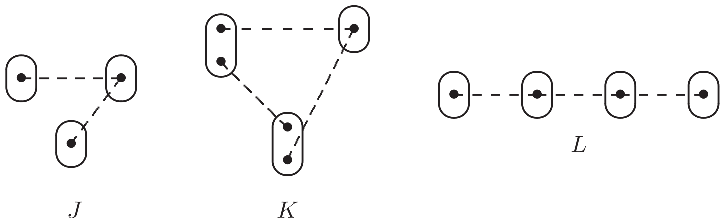

Fig. 6. Three patterns which are topological-minor tractable.

Unfortunately this result does not extend to graphs of higher degree, as the following example shows.

Example 4.7. Consider a star graph G where the central vertex has degree 4. Note that G is planar.

In all subdivisions of G, the central vertex still has degree 4, so it cannot occur as a topological minor in any graph of maximum degree three. Hence, by Lemma 4.2, $\mathrm{Patt}(G)$ cannot occur as a topological minor in any CSP instance whose constraint graph is a hexagonal grid. Since the treewidth of the class of hexagonal grids is unbounded [20], this structural class of CSP instances is intractable, assuming $\mathrm{FPT} \neq \mathrm{W}[1]$, by the results of [28]. $\square$

5. Tractable classes that generalise acyclicity

In this section we will give several more examples of patterns that are topological-minor tractable. We conclude the section with Theorem 5.4 where we define several new tractable classes which properly extend the class of acyclic CSP instances discussed in Section 4.

Consider the patterns shown in Fig. 6. By Theorem 3.12, J is sub-pattern tractable and hence also topological-minor tractable, by Lemma 3.16. However, the remaining patterns, K and L are more interesting.

Theorem 5.1. *The pattern K, shown in Fig. 6, is sub-pattern* NP*-complete but topological-minor tractable.*

Proof. By Theorem 3.12, K is sub-pattern NP-complete.

To show that K is topological-minor tractable, consider an instance I in which the pattern K does not occur as a topological minor. If the pattern J from Fig. 6 does not occur as a sub-pattern in $\mathrm{Patt}(I)$ then we are done since, as noted above, $\mathrm{CSP}_{\overline{\mathrm{SP}}}(J)$ is tractable and thus I can be solved in polynomial time.

On the other hand, if J does occur as a sub-pattern in $\mathrm{Patt}(I)$, then we will build a special tree decomposition T of the constraint graph of I, where each node of T is a subset of the vertices of the constraint graph of I, and all non-leaf nodes of T have size 1.

In more detail, let G_I be the constraint graph of I. Suppose the pattern J, shown in Fig. 6, occurs as a sub-pattern in $\mathrm{Patt}(I)$ on the three parts corresponding to the triple of variables (x, y, z) in I, with y being the variable at which the two negative edges meet. Since K does not occur as a topological minor in I, it follows that there is no path from x to z in G_I that does not pass through y. Hence y is an articulation point of G_I.

Let $C_1, \ldots, C_k$ be the components of $G_I \setminus \{y\}$, and denote by I_{C_i} the sub-instance of I on the variables of $C_i \cup \{y\}$. We form a tree decomposition of G_I as follows: the root of T is the subset containing just the variable y and has k children. If the pattern J does not occur as a sub-pattern in $C_i \cup \{y\}$, then the i-th child of the root is a leaf node corresponding to the sub-instance I_{C_i}. Otherwise, if the pattern J does occur as a sub-pattern in $C_i \cup \{y\}$, then we proceed in the same fashion and decompose C_i into a sub-tree rooted at the i-th child.

Since $\mathrm{CSP}_{\overline{\mathrm{SP}}}(J)$ is tractable, any sub-instance corresponding to a leaf of this tree decomposition can be solved in polynomial time for each possible assignment to its unique articulation variable which joins it to its parent node in the tree-decomposition. Hence in polynomial time we can solve this sub-instance, eliminate the corresponding leaf, and possibly eliminate some values in the domain of this articulation variable. After eliminating all non-trivial leaf nodes in this way, the remaining sub-instance of G_I is tree structured and hence can be solved in polynomial time. $\square$

We will show in Theorem 5.3 below that the pattern L shown in Fig. 6 is also topological-minor tractable. In order to do so, we will extend the proof technique used in Theorem 5.1 to a *generic scheme* for proving topological-minor tractability of patterns.

To develop our generic scheme we need some standard results from graph theory. If S is a set of vertices of a graph G, we write $G[S]$ for the induced graph on S.

A *tree decomposition* of a graph $G = (V, E)$ is a tree T, together with a subset V_t of the vertices of G for each node $t \in T$, such that $\bigcup_{t\in T} V_t = V$, each edge $e \in E$ is contained in V_t for some $t \in T$, and for any vertex $v \in V$ the set $\{t \mid v \in V_t\}$ is a connected sub-tree of T. The *torsos* of a tree decomposition $(T, (V_t)_{t\in T})$ of a graph G are the graphs H_t, $t \in T$, obtained from $G[V_t]$ by adding all the edges $\{x, y\}$ such that $x, y \in V_t \cap V_{t'}$ where t' is any neighbour of t in T.

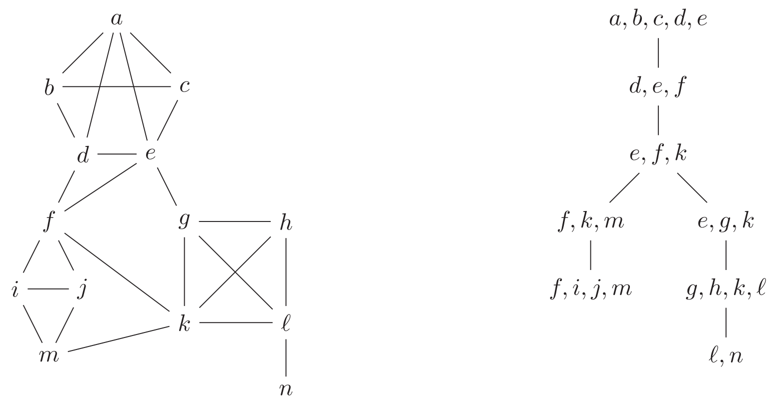

Fig. 7. A graph and its Tutte decomposition.

A *Tutte decomposition* of a graph G is a tree decomposition $(T, (V_t)_{t\in T})$ of G, where $|V_t \cap V_{t'}| \leq 2$ for every pair of neighbours t and t' in T, and the torso of each node is either three-connected, or a cycle, or has at most 2 vertices. It is known that every finite graph has a Tutte decomposition of this kind [42], and that such a decomposition can be found in linear time [31].

Example 5.2. Fig. 7 shows a graph and a possible Tutte decomposition. □

To demonstrate topological-minor tractability for a pattern P we proceed as follows. Let I be an instance in which P does not occur as a topological minor and let G_I be its constraint graph. We denote by n the number of variables in I and by d the maximum domain size of any variable in I.

Build a Tutte decomposition of G_I, and consider any leaf node s in this decomposition. The subset of variables associated with node s will be denoted S, and the variables associated with the remainder of the nodes of the tree decomposition after removing the leaf s will be denoted by T. Note that S and T share at most 2 variables. Let $I[S]$ be the sub-instance of I on S and $I[T]$ be the sub-instance of I on T. Suppose that the following two assumptions hold:

(**A1**) $I[S]$ can be solved and its solutions projected onto the variables shared with T in polynomial time; the resulting reduced instance on T will be denoted by $I'[T]$.
(**A2**) P does not occur as a topological minor in $\mathrm{Patt}(I'[T])$.

Then it follows that a recursive algorithm, which at each step chooses some leaf s of the decomposition, and then solves the associated sub-problem $I[S]$ to obtain the reduced instance $I'[T]$, will solve the original instance using a polynomial (in n and d) number of calls to the polynomial-time algorithm from (**A1**).

In the proofs below we will omit the simple cases where S and T share only 1 variable, or S contains at most 3 vertices, or the torso of S is a cycle (and hence has treewidth 2 and is solvable in polynomial time). Hence we will assume that the torso of S contains more than three vertices and is three-connected.

Finally, note that if S and T share the variables $\{u, v\}$, then we have the following:

- Any path in G_I from a vertex in S to a vertex in T must pass through u or v;
- There must exist some path from u to v in $G_I[T]$, which we will denote $path_T(u, v)$.

We now use this generic scheme to prove the tractability of pattern L from Fig. 6.

Theorem 5.3. *The pattern L, shown in Fig. 6, is sub-pattern* NP*-complete but topological-minor tractable.*

Proof. By Theorem 3.12, L is sub-pattern NP-complete.

To establish topological-minor tractability using the generic scheme described above we only need to establish the two assumptions.

(**A1**) Let J be the pattern consisting of two intersecting negative edges, shown in Fig. 6. Suppose that J occurs in $\mathrm{Patt}(I[S])$ as a sub-pattern on two *disjoint* triples of variables (x, y, z) and (x', y', z') in $I[S]$. As explained above for the generic scheme, we can assume that the torso of S is 3-connected. It follows by Menger's theorem [21] that there are three disjoint paths from x to x' in the torso of S. There must be one of these paths, π, which does not pass through y or y'. We claim that there must be a subpath σ of π which begins at x or z and ends at x' or z' and which does not pass through any other variables in $\{x, y, z, x', y', z'\}$. To prove the claim first note that if π does not pass through z and z' then π satisfies

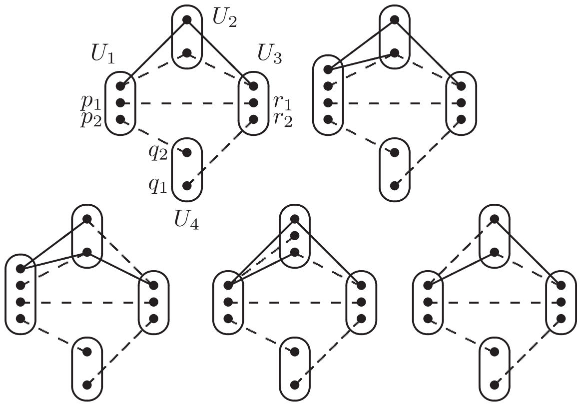

Fig. 8. Topological-minor tractable patterns derived from sub-pattern tractable patterns.

the claim. If z appears on π but z' does not appear on π then the subpath σ of π from z to x' satisfies the claim. A similar argument works for the case when z' appears on π but z does not. If both z and z' appear on π then we have a subpath of π from z to z'. Without loss of generality, suppose that σ joins x to x'. But then L occurs as a topological minor on the extended path σ^+ given by $z \to y \to x, \sigma, x' \to y' \to z'$.

But this implies that L occurs as a topological minor in Patt(I), since if σ^+ passes by the edge $\{u, v\}$ in the torso of S, this edge can be replaced by $path_T(u, v)$ which is a path from u to v in T, whose existence was noted in the discussion above. Since this contradicts our initial assumption, we can deduce that J does not occur in Patt($I[S]$) as a sub-pattern on two *disjoint* triples.

We can therefore deduce that all pairs of triples of variables (x, y, z), (x', y', z') for which J occurs as a sub-pattern in Patt($I[S]$) intersect, i.e., $\{x, y, z\} \cap \{x', y', z'\} \neq \emptyset$. Now, consider an arbitrary triple of variables (x, y, z) on which J occurs as a sub-pattern. It follows that the instance which results after any instantiation (and removal) of the three variables x, y, z contains no occurrence of J as a sub-pattern, since for each triple of variables (x', y', z') on which J occurs in $I[S]$, at least one of its variables has been eliminated by instantiation.

Thus, after instantiation of at most three variables, Patt($I[S]$) does not contain J as a sub-pattern. This also holds for any version of $I[S]$ obtained by instantiating the variables u, v. As noted above, $\mathrm{CSP}_{\overline{\mathrm{SP}}}(J)$ is tractable. We can therefore determine in polynomial time which instantiations of u, v can be extended to a solution of $I[S]$. We remove the pair (p, q) from R_{uv} in I whenever the assignment of p to u and q to v cannot be extended to a solution to $I[S]$. Finally, we delete all variables in S from I apart from u and v. Proceeding in this way we construct $I'[T]$ in polynomial time as required.

(**A2**) Suppose, for a contradiction, that we introduce some occurrence of the pattern L as a topological minor in Patt($I'[T]$) when reducing I to $I'[T]$. This occurrence of L must use a newly-introduced edge in $I'[T]$. During the reduction from I to $I'[T]$, we can introduce negative (but not positive) edges in Patt($I'[T]$) between the parts corresponding to u and v. Suppose that a negative edge (p, q) is introduced by the reduction from I to $I'[T]$. This can only be the case if there was a path $\pi = (u, w_1, \ldots, w_t, v)$ in the constraint graph $G_I[S]$ and hence a sequence of negative edges between the corresponding parts in Patt($I[S]$) linking p to q. This means that we can replace the newly-introduced edge in the occurrence of L in Patt($I'[T]$) by a sequence of negative edges so that L occurs as a topological minor in Patt(I) for the original instance I. This contradiction shows that we cannot introduce L as a topological minor in Patt($I'[T]$) when reducing I to $I'[T]$.

Hence we have established both assumptions, so the result follows by our generic proof scheme. Note that the number of instances of $\mathrm{CSP}_{\overline{\mathrm{SP}}}(J)$ that need to be solved is $O(nd^5)$. □

As our final result in this section we show how the well-known tractable class of acyclic instances can be generalised to obtain larger tractable classes defined by forbidding the occurrence of certain patterns as topological minors. The main tool we use will again be the generic scheme based on Tutte decompositions described above.

Theorem 5.4. *Let P_0 be any sub-pattern tractable pattern with three parts, U_1, U_2, U_3 where there is at most one negative edge between U_1 and U_2, and between U_2 and U_3, and no edges between U_1 and U_3.*

Let P be a pattern with four parts U_1, U_2, U_3, U_4 obtained by extending P_0 as follows. The pattern P has six new points $p_1, p_2 \in U_1$, $q_1, q_2 \in U_4$, and $r_1, r_2 \in U_3$, together with three new negative edges $\{p_1, r_1\}$, $\{p_2, q_1\}$, $\{q_2, r_2\}$ (see Fig. 8). Any such P is topological-minor tractable.

Proof. The proof uses the generic scheme described in this section, so we only need to establish the two assumptions.

(**A1**) Suppose first that P_0 occurs as a sub-pattern in Patt($I[S]$) on the triple of variables (x, y, z). As explained above, when using the generic scheme we will assume that the torso of S is three-connected. Then, by Menger's theorem there are three disjoint paths π_1, π_2, π_3 from x to z in the torso of S. Hence there must be two of these paths, say π_1 and π_2, which do not pass through y. But this implies that P occurs as a topological minor in $\mathrm{Patt}(I)$, since if either π_1 or π_2 passes through the edge $\{u, v\}$ in the torso of S, this edge can be replaced by $path_T(u, v)$ which is a path from u to v in $G_I[T]$, whose existence was shown in the discussion of the generic scheme above. Since this contradicts our initial assumption, we can assume that P_0 does *not* occur as a sub-pattern in $\mathrm{Patt}(I[S])$. This also holds for any sub-problem of $I[S]$ obtained by instantiating the variables u, v. Therefore, by the sub-pattern tractability of P_0, we can determine in polynomial time which instantiations of u, v can be extended to a solution of $I[S]$. We remove the pair (p, q) from R_{uv} in I whenever the assignment of p to u and q to v cannot be extended to a solution to $I[S]$. Finally, we delete all variables in S from I except for u and v. Proceeding in this way we construct $I'[T]$ in polynomial time, as required.

(**A2**) Suppose, for a contradiction, that we introduce the pattern P as a topological minor of $\mathrm{Patt}(I'[T])$ when reducing I to $I'[T]$. This occurrence of P must use a newly-introduced negative edge. Observe that, by definition, P contains at most one negative edge between any two parts. Suppose that a negative edge (p, q) is introduced by the reduction from I to $I'[T]$. This can only be the case if there was a path $\pi = (u, w_1, \ldots, w_t, v)$ in the constraint graph $G_I[S]$ and hence a sequence of negative edges between the corresponding parts in $\mathrm{Patt}(I[S])$ linking p to q. Furthermore, in $I'[T]$, if there is a positive edge (p', q') between the parts corresponding to u and v then there is necessarily a solution to $I[S]$ including the assignments p' to u and q' to v (and hence a solution on the subinstance $I[\pi]$ of $I[S]$ on the path $\pi = (u, w_1, \ldots, w_t, v)$ in $I[S]$). This means that we can replace the edge (p, q) in the occurrence of P in $I'[T]$ by a sequence of negative edges so that P occurs as a topological minor in $\mathrm{Patt}(I)$ for the original instance I. This contradiction shows that we cannot introduce an occurrence of P as a topological minor in $\mathrm{Patt}(I'[T])$ when reducing I to $I'[T]$.

Hence we have established both assumptions, so the result follows by our generic proof scheme. Note that the number of instances of $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P_0)$ that need to be solved is $O(nd^2)$. □

By [12, Theorem 1], all sub-pattern tractable patterns P_0 satisfying the conditions of Theorem 5.4 can be reduced to sub-patterns of one of five specific patterns. Extending each of these to a pattern P as described in Theorem 5.4 gives the five topological-minor tractable patterns shown in Fig. 8. For each of these patterns P, the pattern shown in Fig. 2 occurs as a sub-pattern and hence as a topological minor of P. Thus, by the transitivity of occurrence as a topological minor, each tractable class $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ necessarily contains all acyclic binary CSP instances.

6. Detection of topological minors

For every fixed undirected graph H, there is an $O(n^3)$ time algorithm that tests, given a graph G with n vertices, if H is a topological minor of G [29].

However, for detecting topological minors in *patterns* the situation is different. Characterising all patterns P for which it is possible to decide in polynomial time whether P occurs as a topological minor in a given pattern P' remains an open problem. However, we have the following partial results.

By Lemma 4.2, deciding whether a negative pattern of the form $\mathrm{Patt}(G)$ for some graph G occurs as a topological minor in a pattern P' amounts to detecting whether G is a topological minor of the constraint graph of P', and hence can be achieved in polynomial time [29]. By Proposition 3.22, deciding whether a star-like negative pattern occurs as a topological minor in an instance can also be achieved in polynomial time because this is equivalent to deciding whether it occurs as a sub-pattern, which is achievable in polynomial time by exhaustive search.

Proposition 6.1. *For each of the patterns J, K or L shown in Fig. 6, deciding whether that pattern occurs as a topological minor in a given instance I can be done in polynomial time.*

Proof. The pattern J shown in Fig. 6 is star-like, and hence the result follows from the observation just made. For the pattern K shown in Fig. 6 it is sufficient to discover by exhaustive search all occurrences of J as a sub-pattern of $\mathrm{Patt}(I)$ on the three parts corresponding to the triple of variables (x, y, z) in I, with y being the variable at which the two negative edges meet, and then check for each one whether x and z are connected in $G_I \setminus y$.

For the pattern L shown in Fig. 6 it is sufficient to consider all pairs of occurrences of J as a sub-pattern of $\mathrm{Patt}(I)$ on parts corresponding to (x, y, z) and (x', y', z') (where the negative edges meet in parts y and y'). We can then check that either (y, z) and (x', y') coincide, or z and x' coincide, or z and x' are connected by a path in G_I that does not pass through any of the parts x, y, y', z'. □

For each of the patterns shown in Fig. 8 the complexity of deciding whether it occurs as a topological minor in a given instance I is currently unknown. However, in polynomial time we can build a Tutte decomposition for I and decide whether each of the sub-problems associated with its nodes are members of $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P_0)$ for the appropriate pattern P_0, and this is the only condition required to solve I in polynomial time using the algorithm described in the proof of Theorem 5.4.

Our next result shows that for some patterns (such as the 4-part pattern M shown in Fig. 9), it is coNP-complete to determine whether the pattern occurs as a topological minor in an arbitrary given pattern.

Theorem 6.2. *The problem of deciding $I \in \mathrm{CSP}_{\overline{\mathrm{TM}}}(M)$ is* coNP*-complete.*

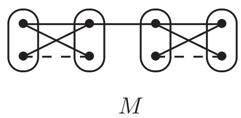

Fig. 9. A pattern that is coNP-complete to detect as a topological minor.

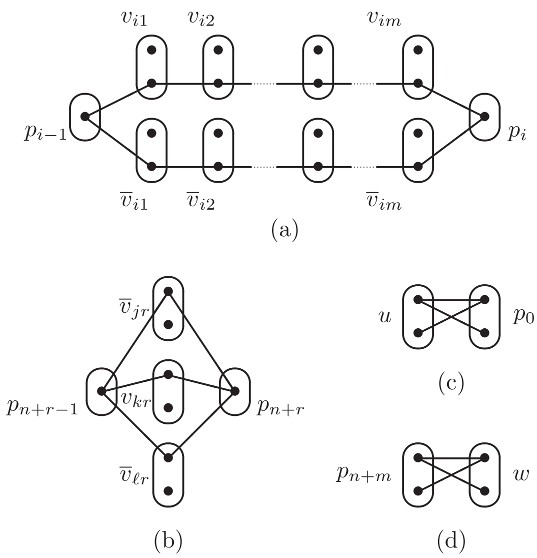

Fig. 10. The building blocks for the CSP instance I constructed in the proof of Theorem 6.2.

Proof. The problem is clearly in coNP, so it suffices to give a reduction from 3-SAT to the complement of the problem of deciding $I \in \mathrm{CSP}_{\overline{\mathrm{TM}}}(M)$.

Let I_{SAT} be an instance of 3-SAT with variables $x_1, \dots, x_n$ and clauses $C_1, \dots, C_m$. We will create a binary CSP instance I with variables $\{u, w\} \cup \{p_i \mid i = 0 \dots n+m\} \cup \{v_{ir}, \overline{v}_{ir} \mid i = 1 \dots n, r = 1 \dots m\}$, such that determining whether $M \xrightarrow{TM} \mathrm{Patt}(I)$ is equivalent to deciding whether I_{SAT} has a solution. The instance I that we create will be Boolean in the sense that all variables will have domain size at most two. (In fact all the variables p_i, except for p_0 and p_{n+m}, will have single-valued domains.)

Consider the patterns shown in Fig. 10, where each part is labelled with a variable of I. Using these patterns we build a complete pattern corresponding to the instance I, as follows:

- For each variable x_i in I_{SAT} we include a pattern P_{x_i} of the form shown in Fig. 10(a).
- For each clause C_r in I_{SAT} we include a pattern P_{C_r} of the form shown in Fig. 10(b), where the choice of variables for the three central parts depends on the literals in the clause C_r in the following way: variable v_{ir} corresponds to $\neg x_i$ occurring in clause C_r and variable $\overline{v}_{ir}$ corresponds to x_i occurring in clause C_r. That is, the example shown in Fig. 10(b) would correspond to the clause $x_j \vee \neg x_k \vee x_\ell$.
- We also include the pattern shown in Fig. 10(c) and the pattern shown in Fig. 10(d).
- Finally, we complete the resulting pattern to obtain $\mathrm{Patt}(I)$ by adding negative edges between all pairs of points in distinct parts that are not already directly connected by a positive or negative edge.

The only pairs of parts in $\mathrm{Patt}(I)$ that are connected by more than one positive edge are $\{u, p_0\}$ and $\{p_{n+m}, w\}$. So, if M occurs as a topological minor in $\mathrm{Patt}(I)$, then the points of M must map injectively to these two pairs of parts. Therefore, deciding whether M occurs as a topological minor in $\mathrm{Patt}(I)$ is equivalent to deciding whether there is a path π of positive edges from p_0 to p_{n+m} in $\mathrm{Patt}(I)$ which passes through each part at most once.

Any such path π must pass through the points $p_0, p_1, \dots, p_{n+m}$ in this order, because the positive edges in P_{x_i} ($1 \le i \le n$) use different points in each part (shown as the bottom of the two points in Fig. 10) from the positive edges in P_{C_r} ($1 \le r \le m$) (which use the top points), so there are no short-cuts.

If such a path π exists, then for each variable x_i of I_{SAT}, the path π must select in P_{x_i} either the upper path through variables v_{ir} ($r = 1, \dots, m$) or the lower path through variables $\overline{v}_{ir}$ ($r = 1, \dots, m$). Thus π selects a truth value for each variable x_i: TRUE if π follows the upper of these two paths, FALSE otherwise.

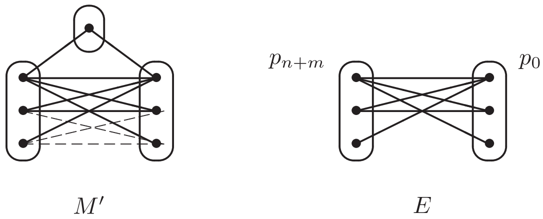

Fig. 11. The pattern M' and one of the building blocks for the globally-consistent instance I' in which detecting it is coNP-complete.

Moreover, for each clause C_r in I_{SAT} the path π must pass from p_{n+r-1} to p_{n+r} by one of the three paths in P_{C_r} without passing through parts that have been already used by π. Thus, for π to exist it must have already assigned TRUE to one of the literals of the clause C_r.

It follows that M occurs as a topological minor of $\mathrm{Patt}(I)$ if and only if $\mathrm{Patt}(I)$ has an appropriate path of positive edges, which occurs if and only if I_{SAT} is satisfiable. $\square$

The instance I in the proof of Theorem 6.2 is clearly inconsistent since there are some constraint relations which are empty. An instance is said to be *globally consistent* if each variable-value assignment (v_i, a) can be extended to a solution. We now give another example of a pattern which is coNP-complete to detect as a topological minor even in globally-consistent instances.

Theorem 6.3. *The problem of deciding* $I \in \mathrm{CSP}_{\overline{\mathrm{TM}}}(M')$ *for globally-consistent instances* I *is* coNP-*complete.*

Proof. We use a very similar construction to the one used in the proof of Theorem 6.2. Let I be the instance constructed in that proof. Let I' be identical to I except that:

- we replace the sub-instances obtained from the patterns shown in Fig. 10(c) and Fig. 10(d) with a single sub-instance obtained from the pattern E shown in Fig. 11;
- for each variable-value assignment (v, a) of I, we create a solution which is an extension of (v, a), by adding a new value $b(v, a, v')$ to the domain of each variable $v' \neq v$ which is compatible with (v, a) and with all such values $b(v, a, v'')$ $(v'' \notin \{v, v'\})$, but incompatible with all other variable-value assignments.

By construction, I' is clearly globally-consistent. If M' occurs as a topological minor of $\mathrm{Patt}(I')$, then the points of M' must map injectively to the points of E, and so again the question is whether there is a path (of length greater than 1) of positive edges linking p_0 to p_{n+m}. As in the proof of Theorem 6.2, this path exists if and only if the instance I_{SAT} is satisfiable. Hence, the decision problem $I \in \mathrm{CSP}_{\overline{\mathrm{TM}}}(P_X)$ for globally-consistent instances I is coNP-complete. $\square$

Theorems 6.2 and 6.3 show that not all classes defined by forbidding topological minors can be recognized in polynomial time. Certain uses of tractable classes require polynomial-time recognition: in particular, the automatic recognition and resolution of easy instances within general-purpose solvers. On the other hand, polynomial-time recognition of a tractable class $\mathcal{C}$ is not required for the construction of a polynomial-time solvable relaxation in $\mathcal{C}$, nor in the proof (by a human being) that a subproblem of CSP encountered in practice falls in $\mathcal{C}$.

7. Augmented patterns

For some CSP instances we have extra information such as an ordering on the variables or on the domains (or both). In this section we introduce the idea of adding an additional relation to a pattern to allow us to capture information of this kind. A pattern P, together with an additional relation on the points of P will be called an *augmented pattern*. We will demonstrate that augmented patterns can be used to define new hybrid tractable classes that extend those described in earlier sections.

Definition 7.1. An *augmented pattern* is a pair (P, R) where P is a pattern and R is a relation (of any arity) over the points of P. The augmented pattern (P, R) will be denoted P_R.

Obvious examples of relations that could be added to a pattern are disequality relations or partial orders on points, and this idea has been explored in a number of papers [7,13,16].

Definition 7.2. A homomorphism between augmented patterns P_R and $P'_{R'}$ is a homomorphism h from P to P' such that for all tuples $(x_1, x_2, \ldots, x_k) \in R$, the tuple $(h(x_1), h(x_2), \ldots, h(x_k)) \in R'$.

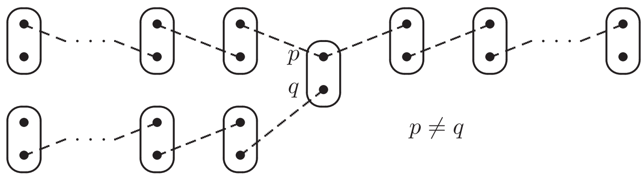

Fig. 12. The augmented pattern Pivot$_{\neq}(k)$.

Using this extended definition of homomorphism, we can extend the notion of occurring as a sub-pattern (Definition 3.2) and occurring as a topological minor (Definition 3.4) to augmented patterns in the natural way.

Now we can extend Definitions 3.8 and 3.14, as follows, to define restricted classes of CSP instances and associated relations by forbidding the occurrence of certain augmented patterns.

Definition 7.3. Let m be a constant, and let $\mathcal{S}$ be a set of augmented patterns such that for each $P_R \in \mathcal{S}$ the relation R has arity m. Let Rel be a partial function that maps an instance I to a relation R_I of arity m over the points of Patt(I).

We denote by $\mathrm{CSP}_{\overline{\mathrm{SP}}}(\mathcal{S}, \mathrm{Rel})$ the set of all binary CSP instances I such that Rel(I) is defined and for all $P_R \in S$ it is not the case that $P_R \overset{SP}{\rightarrow} \mathrm{Patt}(I)_{\mathrm{Rel}(I)}$.

We denote by $\mathrm{CSP}_{\overline{\mathrm{TM}}}(\mathcal{S}, \mathrm{Rel})$ the set of all binary CSP instances I such that Rel(I) is defined and for all $P_R \in S$ it is not the case that $P_R \overset{TM}{\rightarrow} \mathrm{Patt}(I)_{\mathrm{Rel}(I)}$.

One of the simplest ways to augment a pattern P is by adding a binary disequality relation, $\neq$, to specify that some points of P are distinct. A homomorphism from an augmented pattern $P_{\neq}$ to an augmented pattern $Q_{\neq}$ must map points that are specified to be distinct in P to points that are specified to be distinct in Q. In the next three theorems, we shall assume that for any instance I, *all* points in Patt$(I)_{\neq}$ are specified to be distinct. In other words, we shall assume that for any instance I the function Rel introduced in Definition 7.3 always returns the binary relation $\neq$ containing all pairs of distinct points of I. We will denote this function by Rel$_{\neq}$.

Now consider the augmented pattern Pivot$_{\neq}(k)$ which is obtained from the pattern Pivot(k) defined in Definition 3.11 by adding a disequality relation specifying that the two points in the central node are distinct, as shown in Fig. 12. Forbidding this pattern from occurring as a sub-pattern results in a larger class of instances than forbidding the pattern Pivot(k), but our next result shows that this larger class is still tractable.

Theorem 7.4. *The augmented pattern Pivot$_{\neq}(k)$, shown in Fig. 12, is sub-pattern tractable.*

Proof. Let $I \in \mathrm{CSP}_{\overline{\mathrm{SP}}}(Pivot_{\neq}(k), \mathrm{Rel}_{\neq})$ for some constant k. If Patt(I) has a point $x_{v,a}$ which belongs to no negative edge (i.e., it is compatible with all assignments to all other variables), then we can clearly remove all points in the same part as $x_{v,a}$ without introducing the pattern or affecting the existence of a solution. Thus we can assume without loss of generality that Patt(I) contains no such points. A similar remark holds if Patt(I) has any parts containing just a single point.

We can also assume without loss of generality that the constraint graph of I is connected. A variable v is called an *articulation variable* of I if removing v from I disconnects the constraint graph of I. Any instance can be decomposed into a tree of components which only intersect at articulation variables. It therefore suffices to show that any instance I without articulation variables can be solved in polynomial time, so we shall assume that I has no articulation variables.

If Pivot$(2k)$ does not occur as a sub-pattern in Patt(I) then, by Theorem 3.12 we have that I is tractable.

To deal with the remaining case, assume that Pivot$(2k)$ occurs as a sub-pattern in Patt(I) with the central part U of Pivot$(2k)$ mapping to part V of Patt(I). Let S_{2k} be the set of parts of Patt(I) to which the parts of Pivot$(2k)$ are mapped.

Since Pivot$_{\neq}(k)$ does not occur as a sub-pattern in Patt$(I)_{\neq}$ (and hence neither does Pivot$_{\neq}(2k)$), the two points in the central part U of Pivot$(2k)$ must map to the *same* point in Patt(I), which we denote by $x_{v,a}$.

By our assumptions, we know that there is another (distinct) value b in the domain of v which belongs to a negative edge in Patt(I), connecting part V to some other part W. If W is only connected to S_{2k} in the constraint graph of Patt(I) via V, then v is an articulation variable of I, which contradicts our assumption. Hence, there is a path π in the constraint graph of Patt(I) linking W to some part $Y \in S_{2k}$ such that $Y \neq V$.

By choosing π to be minimal, we can assume that no other parts on the path π belong to S_{2k}. Now, since Y must lie on one of the three branches of the occurrence of Pivot$(2k)$ in Patt(I), we can extend π by following this branch from Y either towards or away from the central part V, in order to obtain a path of length at least k. This length-k path, together with the first k variables of the other two branches of Pivot$(2k)$, gives an occurrence of the pattern Pivot$_{\neq}(k)$ in Patt$(I)_{\neq}$, which contradicts our choice of I, so we are done. $\square$

Now consider the augmented pattern $K_{\neq}$, shown in Fig. 13, which is obtained from the pattern K shown in Fig. 6 by adding a disequality relation to specify that any two points in the same part are distinct. We now show that forbidding $K_{\neq}$

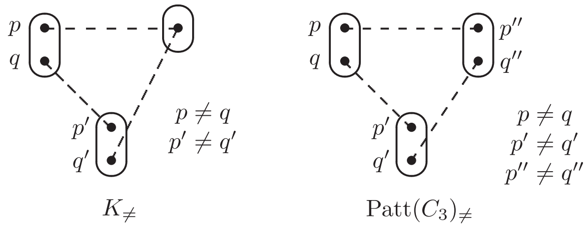

Fig. 13. Two augmented patterns which are topological-minor tractable.

from occurring as a topological minor results in a tractable class (which is larger than the class obtained by forbidding the pattern K as a topological minor discussed in Theorem 5.1).

Theorem 7.5. *The augmented pattern $K_{\neq}$, shown in Fig. 13, is sub-pattern NP-complete but topological-minor tractable.*

Proof. By Theorem 3.12, the (negative) pattern K shown in Fig. 6 is sub-pattern NP-complete. Since $\text{CSP}_{\overline{\text{SP}}}(K) \subseteq \text{CSP}_{\overline{\text{SP}}}(K_{\neq}, \text{Rel}_{\neq})$, we have that $K_{\neq}$ is also sub-pattern NP-complete.

To show that $K_{\neq}$ is topological-minor tractable we will show that establishing arc-consistency is sufficient to decide the existence of a solution for any instance in $\text{CSP}_{\overline{\text{TM}}}(K_{\neq}, \text{Rel}_{\neq})$.

By Lemma 3.7, without loss of generality we need consider only arc consistent instances. We will show, by induction on the number of variables, that in any arc-consistent instance $I \in \text{CSP}_{\overline{\text{TM}}}(K_{\neq}, \text{Rel}_{\neq})$, any assignment to a single variable can be extended to a solution of I. This is certainly true for instances on up to two variables, by the definition of arc consistency.

Now assume that I has more than two variables, and consider the assignment of the value a to the variable v. Let $I[v=a]$ be the instance obtained from I by making this assignment, eliminating variable v and eliminating from the domain of all other variables w all values b such that $(a,b) \notin R_{vw}$. By arc consistency, none of the resulting domains in $I[v=a]$ is empty, i.e., for each variable w there is a value c_w in the domain of w such that $(a, c_w) \in R_{vw}$. By the absence of $K_{\neq}$ as a topological minor in $\text{Patt}(I)_{\neq}$, we can deduce that all variables w that were connected to v in the constraint graph of I are not connected in the constraint graph of $I[v=a]$.

Let $S_1, \ldots, S_m$ be the connected components of the constraint graph of $I[v=a]$. For any $k = 1, \ldots, m$, consider the subinstance $I[S_k]$ of the original instance I on the variables of S_k. Clearly, each $I[S_k] \in \text{CSP}_{\overline{\text{TM}}}(K_{\neq}, \text{Rel}_{\neq})$ and each $I[S_k]$ is arc-consistent. Furthermore, since at least the variable v has been eliminated from the original set of variables, we know that each $I[S_k]$ has strictly fewer variables than I (even if $m = 1$). Hence, by our inductive hypothesis, the assignment of any value c_w to any variable w in $I[S_k]$ can be extended to a solution s_k to $I[S_k]$. The solutions s_k ($k = 1, \ldots, m$) together with the assignment of a to v then form a solution to I and the result follows by induction. □

Now consider the augmented pattern $\text{Patt}(C_3)_{\neq}$, shown in Fig. 13, which is obtained from the pattern $\text{Patt}(C_3)$ shown in Fig. 2 by adding a disequality relation specifying that any two points in the same part are distinct. We now show that forbidding $\text{Patt}(C_3)_{\neq}$ from occurring as a topological minor results in a tractable class (which is larger than the class of acyclic instances obtained by forbidding the pattern $\text{Patt}(C_3)$ as a topological minor discussed in Proposition 4.3).

Theorem 7.6. *The augmented pattern $\text{Patt}(C_3)_{\neq}$, shown in Fig. 13, is sub-pattern NP-complete but topological-minor tractable.*

Proof. By Theorem 3.12, the (negative) pattern $\text{Patt}(C_3)$ shown in Fig. 2 is sub-pattern NP-complete. Since $\text{CSP}_{\overline{\text{SP}}}(\text{Patt}(C_3)) \subseteq \text{CSP}_{\overline{\text{SP}}}(\text{Patt}(C_3)_{\neq}, \text{Rel}_{\neq})$, we have that $\text{Patt}(C_3)_{\neq}$ is also sub-pattern NP-complete.

Singleton arc consistency (SAC) is an operation which consists in applying the following operation on an instance I until convergence: if the instance $I[v=a]$ obtained by making the assignment of the value a to the variable v and establishing arc consistency is empty, then eliminate a from the domain of v in I. To show that $\text{Patt}(C_3)_{\neq}$ is topological-minor tractable we will show that SAC is a decision procedure for $\text{CSP}_{\overline{\text{TM}}}(\text{Patt}(C_3)_{\neq}, \text{Rel}_{\neq})$.

Since establishing SAC cannot introduce any occurrence of the pattern, we need only consider instances that are singleton-arc-consistent (i.e., where no more eliminations are possible by SAC). We will show, by induction on the number of variables, that in any singleton-arc-consistent instance $I \in \text{CSP}_{\overline{\text{TM}}}(\text{Patt}(C_3)_{\neq}, \text{Rel}_{\neq})$, any assignment to a single variable can be extended to a solution to I. This is certainly true for instances on up to two variables, by the definition of arc consistency.

Now assume that I has more than two variables, and consider the assignment of the value a to the variable v. Let N be the set of parts of $\text{Patt}(I)$ that are connected by a negative edge to $x_{v,a}$. We can assume that $N \neq \emptyset$, otherwise we could make the assignment a to variable v without affecting the rest of the instance I, and thus reduce I to an instance on fewer variables (which by our inductive hypothesis would have a solution).

Now let $I[N]$ be the subinstance of I on the variables corresponding to parts in N, with the domain of each variable w of $I[N]$ reduced to those values c such that $(a,c) \in R_{vw}$. Since I is singleton arc-consistent, $I[N]$ is arc-consistent.

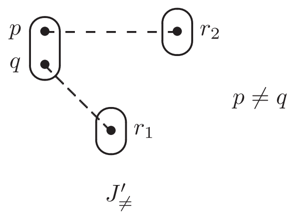

Fig. 14. The augmented pattern $J'_{\neq}$ used in the proof of Theorem 7.6.

Let $J'_{\neq}$ be the augmented pattern shown in Fig. 14. Note that $J'_{\neq} \xrightarrow{SP} \mathrm{Patt}(C_3)_{\neq}$. Now, since $\mathrm{Patt}(C_3)_{\neq}$ does not occur as a topological minor in $\mathrm{Patt}(I)_{\neq}$, we can deduce that $J'_{\neq}$ does not occur as a topological minor in $\mathrm{Patt}(I[N])$. Hence, $K_{\neq}$ does not occur as a topological minor in $\mathrm{Patt}(I[N])$ either, since $J'_{\neq} \xrightarrow{SP} K_{\neq}$. By the proof of Theorem 7.5, any arc-consistent instance in $\mathrm{CSP}_{\overline{\mathrm{TM}}}(K_{\neq}, \mathrm{Rel}_{\neq})$ has a solution, so $I[N]$ has a solution which we denote by s_N.

Let u be a variable of $I[N]$ and denote by a_u the value assigned to u by s_N. Let I_u be the subinstance of I on all variables of I except $\{v\} \cup (N \setminus \{u\})$.

Let S_u be the set of variables w of I_u which are either (1) u itself, (2) directly constrained by the assignment of a_u to u (i.e., variables w such that $(a_u, b) \notin R_{uw}$ for some b in the domain of w), or (3) such that the pattern $J'_{\neq}$ occurs as a topological minor in $\mathrm{Patt}(I_u)_{\neq}$ with the point r_1 of $J'_{\neq}$ mapping to x_{u,a_u} and the point r_2 of $J'_{\neq}$ mapping to some point $x_{w,b}$ for some b.

Let $I[S_u]$ be the subinstance of I on the set of variables S_u. Clearly $I[S_u]$ is singleton arc-consistent (since I is), and has fewer variables than I (since $v \notin S_u$). Hence, by our inductive hypothesis, the assignment of value a_u to variable u can be extended to a solution s_u of $I[S_u]$.

Now let $u' \in N \setminus \{u\}$. By the absence of $\mathrm{Patt}(C_3)_{\neq}$ as a topological minor in $\mathrm{Patt}(I)$, we can deduce that no assignment in s_u can be incompatible with any assignment to a variable y in $S_{u'} \setminus S_u$, except possibly in the case that the assignment to y is directly incompatible with both the assignment of a_u to u and $a_{u'}$ to u'. In this latter case, the solution $s_{u'}$ projected onto $S_{u'} \setminus S_u$ is necessarily consistent with s_u.

Hence, by a simple inductive argument, we can create a consistent partial assignment composed of the assignment of a to v, and the assignments specified by s_N and each s_u (projected onto the not-yet-assigned variables).

The rest of the instance I, if it is non-empty, is not constrained by this partial assignment and by our inductive hypothesis has a solution; combining these partial solutions gives a solution to I. $\square$

Classes of the CSP that are defined by specifying a restricted set of constraint relations over some fixed domain D are known as *language classes* [32,24]. Every known tractable language class [32,2] of CSP instances is characterised by an operation $f : D^k \to D$ with the property that for all constraints R_{uv}, and all pairs $(p_1, q_1), (p_2, q_2), \dots, (p_k, q_k) \in R_{uv}$, the pair $(f(p_1, p_2, \dots, p_k), f(q_1, q_2, \dots, q_k)) \in R_{uv}$; such an operation is known as a *polymorphism* of the constraint relations [2,32].

We now show that using augmented patterns we can characterise every known tractable language class using a single forbidden augmented sub-pattern.

Theorem 7.7. *Every tractable language class of binary CSP instances that is characterised by a polymorphism f is equal to $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P_R, \mathrm{Rel}_f)$ for some augmented pattern P_R and function Rel_f.*

Proof. The k-ary operation $f : D^k \to D$ can be specified by a $(k+1)$-ary relation R_f over D where $R_f = \{(a_1, \dots, a_{k+1}) \mid a_{k+1} = f(a_1, \dots, a_k)\}$. Define Rel_f to be the function that maps any CSP instance I over D to the relation R over the points of $\mathrm{Patt}(I)$, where $R = \{(x_{v,a_1}, \dots, x_{v,a_{k+1}}) \mid (a_1, \dots, a_{k+1}) \in R_f\}$.

The class of all instances I over domain D for which all constraint relations admit f as a polymorphism, is precisely the class of instances defined by $\mathrm{CSP}_{\overline{\mathrm{SP}}}(P_R, \mathrm{Rel}_f)$ where $P = (X, E^{\sim}, E^{+}, E^{-})$ with

- $X = U \cup V$, where $U = \{p_1, p_2, \dots, p_{k+1}\}$ and $V = \{q_1, q_2, \dots, q_{k+1}\}$;
- $E^{\sim} = (U \times U) \cup (V \times V)$;
- $E^{+} = \{(p_i, q_i) \mid p_i \in U, q_i \in V, i = 1, 2, \dots, k\}$;
- $E^{-} = \{(p_{k+1}, q_{k+1})\}$;

and $R = \{(p_1, p_2, \dots, p_{k+1}), (q_1, q_2, \dots, q_{k+1})\}$, as illustrated in Fig. 15. $\square$

We remark that the algebraic dichotomy conjecture [5], which is a refinement of the dichotomy conjecture of Feder and Vardi [24], implies that *every* tractable language is characterised by a single polymorphism, and thus under this conjecture Theorem 7.7 applies to *all* tractable language classes of binary CSP instance over a fixed domain.

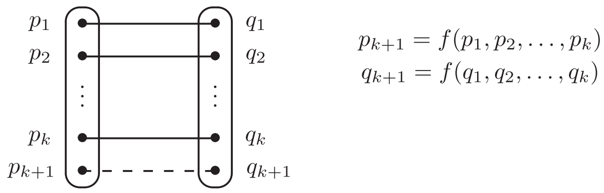

Fig. 15. The augmented pattern P_R used in the proof of Theorem 7.7.

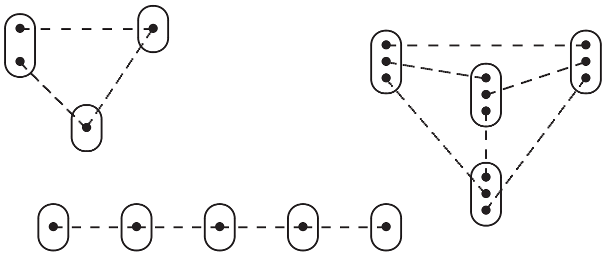

Fig. 16. Three patterns whose topological-minor tractability is open.

8. Conclusions and open problems

The notion of a pattern occurring as a topological minor, introduced here, allows a new approach to the definition of tractable classes of CSP instances. We have shown that this approach, together with the notion of augmented patterns, can unify the description of all tractable structural and language classes, as well as allowing new and more general tractable classes to be identified. We therefore believe that it has great potential for systematically identifying all tractable classes of the CSP.

One long-term goal is to characterise precisely which patterns P are topological-minor tractable and for which such patterns P, $\mathrm{CSP}_{\overline{\mathrm{TM}}}(P)$ is recognisable in polynomial time. For example, Fig. 16 shows three simple patterns whose topological minor tractability is currently open.

Another avenue of future research is the discovery of other applications for topological minors, such as in variable elimination [8]. Indeed, perhaps the most interesting open question is whether the notion of topological minor, introduced in this paper, will find applications other than the definition of tractable classes of the CSP. We have seen that certain classic results from graph theory can lead to results concerning topological minors of CSP instances. An intriguing avenue for future research is to build bridges in the other direction. For example, a corollary of the proof of Theorem 6.2 is that finding a path linking two given vertices and which passes at most once through each part of an n-partite graph is NP-hard. Another way of expressing this is that finding a *heterochromatic path* linking two given vertices in a vertex-coloured graph is NP-hard [35,3].

To achieve further progress it may well be necessary to further refine or modify the definition of a topological minor given here. We regard this work as simply a first step towards a general topological theory of complexity for constraint satisfaction problems.

References

[1] S. Arnborg, A. Proskurowski, D.G. Corneil, Forbidden minors characterization of partial 3-trees, Discrete Math. 80 (1) (1990) 1–19, https://doi.org/10.1016/0012-365X(90)90292-P.
[2] L. Barto, Constraint satisfaction problem and universal algebra, ACM SIGLOG News 1 (2) (2014) 14–24, https://doi.org/10.1145/2677161.2677165.
[3] H. Broersma, X. Li, G. Woeginger, S. Zhang, Paths and cycles in colored graphs, Australas. J. Comb. 31 (2005) 299–311, http://www.combinatorics.cn/publications/papers/2004/LiXL-04A5.pdf.
[4] A. Bulatov, A dichotomy theorem for nonuniform CSP, in: Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS'17), IEEE, 2017, pp. 319–330.
[5] A. Bulatov, A. Krokhin, P. Jeavons, Classifying the complexity of constraints using finite algebras, SIAM J. Comput. 34 (3) (2005) 720–742, https://doi.org/10.1137/S0097539700376676.
[6] D. Cohen, A new hybrid class for which arc-consistency is a decision procedure, in: Proceedings of 9th International Conference on Principles and Practice of Constraint Programming (CP'03), in: Lecture Notes in Computer Science, vol. 2833, Springer-Verlag, 2003, pp. 807–811.
[7] D.A. Cohen, M.C. Cooper, P. Creed, D. Marx, A.Z. Salamon, The tractability of CSP classes defined by forbidden patterns, J. Artif. Intell. Res. 45 (2012) 47–78, https://doi.org/10.1613/jair.3651.
[8] D.A. Cohen, M.C. Cooper, G. Escamoche, S. Živný, Variable and value elimination in binary constraint satisfaction via forbidden patterns, J. Comput. Syst. Sci. 81 (7) (2015) 1127–1143, https://doi.org/10.1016/j.jcss.2015.02.001.
[9] D.A. Cohen, M.C. Cooper, P. Jeavons, S. Živný, Tractable classes of binary CSPs defined by excluded topological minors, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15), AAAI Press, 2015, pp. 1945–1951, http://ijcai.org/Abstract/15/276.
[10] D.A. Cohen, P.G. Jeavons, The power of propagation: when GAC is enough, Constraints 22 (1) (2017) 3–23, https://doi.org/10.1007/s10601-016-9251-0.
[11] M.C. Cooper, An optimal k-consistency algorithm, Artif. Intell. 41 (1989) 89–95, https://doi.org/10.1016/0004-3702(89)90080-5.

[12] M.C. Cooper, G. Escamocher, Characterising the complexity of constraint satisfaction problems defined by 2-constraint forbidden patterns, Discrete Appl. Math. 184 (2015) 89–113, https://doi.org/10.1016/j.dam.2014.10.035.
[13] M.C. Cooper, P.G. Jeavons, A.Z. Salamon, Generalizing constraint satisfaction on trees: hybrid tractability and variable elimination, Artif. Intell. 174 (9–10) (2010) 570–584, https://doi.org/10.1016/j.artint.2010.03.002.
[14] M.C. Cooper, S. Živný, Hybrid tractability of valued constraint problems, Artif. Intell. 175 (9–10) (2011) 1555–1569, https://doi.org/10.1016/j.artint.2011.02.003.
[15] M.C. Cooper, S. Živný, Hybrid tractable classes of constraint problems, in: A. Krokhin, S. Živný (Eds.), Complexity and Approximability of Constraint Satisfaction Problems, in: Dagstuhl Follow-Ups, vol. 7, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017, pp. 113–135.
[16] M.C. Cooper, S. Živný, The power of arc consistency for CSPs defined by partially-ordered forbidden patterns, Log. Methods Comput. Sci. 13 (4) (2017), https://doi.org/10.23638/LMCS-13(4:26)2017.
[17] V. Dalmau, P.G. Kolaitis, M.Y. Vardi, Constraint satisfaction, bounded treewidth, and finite-variable logics, in: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02), in: Lecture Notes in Computer Science, vol. 2470, Springer, 2002, pp. 310–326.
[18] R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.
[19] R. Dechter, J. Pearl, Tree clustering for constraint networks, Artif. Intell. 38 (1989) 353–366, https://doi.org/10.1137/S0097539794266766.
[20] R. Diestel, Graph Theory, 4th ed., Springer, 2010.
[21] G.A. Dirac, Short proof of Menger's graph theorem, Mathematika 13 (1) (1966) 42–44, https://doi.org/10.1112/S0025579300004162.
[22] R. Downey, M. Fellows, Parametrized Complexity, Springer, 1999.
[23] G. Escamocher, Forbidden Patterns in Constraint Satisfaction Problems, Ph.D thesis, IRIT, University of Toulouse, 2014.
[24] T. Feder, M.Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory, SIAM J. Comput. 28 (1) (1998) 57–104, https://doi.org/10.1137/S0097539794266766.
[25] J. Flum, M. Grohe, Parametrized Complexity Theory, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2006.
[26] E. Freuder, A sufficient condition for backtrack-free search, J. ACM 29 (1) (1982) 24–32, https://doi.org/10.1145/322290.322292.
[27] E. Freuder, A sufficient condition for backtrack-bounded search, J. ACM 32 (4) (1985) 755–761, https://doi.org/10.1145/4221.4225.
[28] M. Grohe, The complexity of homomorphism and constraint satisfaction problems seen from the other side, J. ACM 54 (1) (2007) 1–24, https://doi.org/10.1145/1206035.1206036.
[29] M. Grohe, K. Kawarabayashi, D. Marx, P. Wollan, Finding topological subgraphs is fixed-parameter tractable, in: Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC'11), 2011, pp. 479–488.
[30] P. Hell, J. Nešetřil, Colouring, constraint satisfaction, and complexity, Comput. Sci. Rev. 2 (3) (2008) 143–163, https://doi.org/10.1016/j.cosrev.2008.10.003.
[31] J.E. Hopcroft, R.E. Tarjan, Dividing a graph into triconnected components, SIAM J. Comput. 2 (3) (1973) 135–158, https://doi.org/10.1137/0202012.
[32] P. Jeavons, D.A. Cohen, M. Gyssens, Closure properties of constraints, J. ACM 44 (4) (1997) 527–548, https://doi.org/10.1145/263867.263489.
[33] P. Jégou, Decomposition of domains based on the micro-structure of finite constraint-satisfaction problems, in: Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93), AAAI Press, 1993, pp. 731–736, http://www.aaai.org/Library/AAAI/1993/aaai93-109.php.
[34] G. Kun, J. Nešetřil, Forbidden lifts (NP and CSP for combinatorialists), Eur. J. Comb. 29 (4) (2008) 930–945, https://doi.org/10.1016/j.ejc.2007.11.027.
[35] X. Li, S. Zhang, H. Broersma, Paths and cycles in colored graphs, Electron. Notes Discrete Math. 8 (2001) 128–132, https://doi.org/10.1016/S1571-0653(05)80098-8.
[36] F.R. Madelaine, I.A. Stewart, Constraint satisfaction, logic and forbidden patterns, SIAM J. Comput. 37 (1) (2007) 132–163, https://doi.org/10.1137/050634840.
[37] D. Marx, Tractable hypergraph properties for constraint satisfaction and conjunctive queries, J. ACM 60 (6) (2013) 42, https://doi.org/10.1145/2535926.
[38] N. Robertson, P.D. Seymour, Graph minors. V. Excluding a planar graph, J. Comb. Theory, Ser. B 41 (1) (1986) 92–114, https://doi.org/10.1016/0095-8956(86)90030-4.
[39] N. Robertson, P.D. Seymour, Graph minors. XX. Wagner's conjecture, J. Comb. Theory, Ser. B 92 (2) (2004) 325–357, https://doi.org/10.1016/j.jctb.2004.08.001.
[40] F. Rossi, P. van Beek, T. Walsh (Eds.), The Handbook of Constraint Programming, Elsevier, 2006.
[41] F. Rossi, V. Dahr, C. Petrie, On the equivalence of constraint satisfaction problems, in: Proceedings of the European Conference on Artificial Intelligence (ECAI90), 1990, pp. 550–556.
[42] W.T. Tutte, Connectivity in Graphs, University of Toronto Press, 1966.
[43] D. Zhuk, The proof of CSP dichotomy conjecture, in: Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS'17), IEEE, 2017, pp. 331–342.