Fanout Limitations on Constraint Systems

Tomás Feder*

Abstract

We examine several families of constraint systems. Constraint satisfaction and network stability are well understood; network stability was introduced in an attempt to understand the role of fanout. We introduce two new families of constraint systems, in a new attempt to limit fanout. These two families are graph constraint satisfaction and bipartite constraint satisfaction. We obtain the following results, (1) an extension of Schaefer's classification argument for boolean satisfiability to boolean graph constraint satisfaction and bipartite constraint satisfaction, (2) a characterization of the graph constraint satisfaction problems that lack fanout in terms of a new problem, the generalized matroid parity problem, which gives rise to many new problems whose complexity is yet to be determined; in particular, the coindependent set problem is shown to be polynomial, and (3) generalized matroid intersection characterizes the lack of fanout in the bipartite case, with a family of matroid intersection problems as a special polynomially solvable subcase.

Key words: constraint satisfaction, closure properties, fanout, matroid parity.

1 Introduction

This paper is the third component of a structural study that attempts to understand what makes problems polynomially solvable or NP-complete. The first obstacle that must be overcome is the presence of strange in-between problems, shown to exist if P≠NP by Ladner [6] using diagonalization. For this purpose, Feder and Vardi [3] introduce the logic class *monotone monadic SNP without inequality*, a subclass of NP. They show that if any one of the three restrictions on SNP is dropped, then the resulting class contains for every problem in NP an equivalent problem, under polynomial time reductions, and in particular for the in-between problems. Thus all three restrictions are adopted, and it turns out that every problem in this logic class has an equivalent problem in *CSP*, the class of *constraint satisfaction* problems, under randomized polynomial time reductions.

The class CSP is then the first component of the study. It is the natural generalization beyond the boolean case of the class of boolean constraint satisfaction problems. The boolean case was studied by Schaefer [9], who showed that it contains three polynomially solvable cases, namely Horn clauses, 2SAT, and linear equations

*Email: tomas@theory.stanford.edu

modulo 2, and that all other problems in the class are NP-complete, in particular 1-in-3 SAT and not-all-equal SAT.

In their study of the general (not necessarily boolean) case of CSP, Feder and Vardi first showed that three seemingly smaller classes, namely *digraph homomorphism*, *graph retract*, and *partial order retract*, which were previously studied, have in fact the same expressive power as all of CSP, up to polynomial time equivalence. Natural structural properties of graphs underlie polynomiality in more restrictive cases. Thus bipartite graphs, interval graphs, and chordal graphs constitute the polynomial cases for *graph homomorphism* or *H-coloring* (Hell and Nešetřil [4]), *reflexive list* problems (where in addition to the reflexive graph that must be mapped, a list of allowed images for each vertex is specified, Feder and Hell [2]), and *reflexive connected list* problems (where the lists must be connected [2]), respectively; the remaining cases are NP-complete.

In an attempt to obtain such a dichotomy for the general CSP case, Feder and Vardi use two tools, namely Datalog and group theory, to define the *bounded width* and *subgroup* subclasses, which are polynomially solvable (the subgroup problems contain *labeled graph isomorphism* as a main example, and the techniques introduced for this problem apply to the general subgroup case). It is then conjectured that this is essentially all there is to polynomiality in CSP, along the following lines: the only way we know how to show that a problem is not of bounded width requires the problem to have a property called the *ability to count*; once a problem has the ability to count, it seems that it must necessarily contain the general subgroup problem for an abelian group as a special case (namely linear equations modulo p); when a new type of subset of a group, called *nearsubgroup* (same as subgroup in the abelian case), is also allowed in a subgroup problem, the resulting problem reduces to subgroup problems, at least for solvable groups; and if an allowed non-subgroup subset is not a nearsubgroup, then the subgroup problem becomes NP-complete.

Three main polynomial cases, namely *width* 1, *bounded strict width*, and *subgroup*, are all characterized by closure properties (and so is convex programming, a polynomially solvable constraint satisfaction problem over the reals). It turns out that Schaefer's three polynomial cases are width 1, strict width 2, and subgroup, respectively. See also [5] for related work on closure properties of constraints.

A more formal presentation of constraint satisfaction is given in section 2; in particular, the closure properties mentioned above are given, because they will be used later in the study of fanout limitations.

The first component of the study is thus general contraint satisfaction. The second and third components of the study examine instances of a restricted type, in an attempt to understand the significance of limiting fanout, that is, limiting the extent to which information can be copied and used in an instance. It is then natural to assume that every element in the input structure participates in only two constraints. Of course, this is no restriction if there is a constraint on three or more elements that can be used, which requires the constrained elements to be assigned any, but the same, value.

The role of fanout in constraint satisfaction was first observed for *network stability*, the second component of the study, by Mayr and Subramanian [8]. A network has functional constraints, that is, many-input, many-output functions, or *gates*. An element of the input structure participates in two constraints, once as an input and once as an output. Mayr and Subramanian showed that when unlimited fanout is available, in the form of a *COPY* gate, then monotone and linear networks are polynomially solvable, all other cases are NP-complete. They then showed that forbidding the COPY gate is equivalent to requiring that the gates be *adjacency-preserving*: changing the value of one input can affect at most one output. They showed that a special case, the *scatter-free* case, is polynomially solvable, and contains stable matching as a main example. Feder [1] showed that the general adjacency-preserving case is polynomially solvable, thus completing the classification of boolean network stability, and furthermore generalized the results to a non-boolean case where values are chosen from a finite metric space, namely *nonexpansive* network stability. Median graphs and their generalizations arise in characterizing the stable solutions. The results are summarized in section 3.

The third component of the study, that we consider in this paper, is *graph constraint satisfaction*. Here we require every element to participate in only two constraints, but we make no input-output assumption as for network stability. The input structure is then a graph whose vertices are the constraints and whose edges join constraints that share a variable. We shall only consider the boolean case.

We first explore Schaefer's approach, by means of closure properties, in this model with limited fanout. We use the closure properties from section 2 to show that if the constraints do not all belong to one of Schaefer's three polynomial cases, then every clause (disjunction of positive and negative literals) can be simulated. In the bipartite case, every monotone clause can be simulated. Of course, in the absence of fanout limitations, this is just the satisfiability problem, which is NP-complete. However, in the graph model, this problem is easily solvable in polynomial time. This is the material in section 4; notice that very little insight is obtained about what graph constraint satisfaction problems might look like.

We then characterize the problems given by graph constraint satisfaction that do not fit in Schaefer's classification. We use here the result from section 4, and assume that all clauses are available; this is done in section 5. Special cases include graph matching and matroid parity. Lovász [7] showed that matroid parity is exponentially hard in an independence oracle model, and polynomially solvable in the case of spanning tree parity. However, in our model, we can assume that the matroid is the disjoint union of constant size matroids, and the complexity of the problem is open.

We characterize fanout freedom in terms of *generalized matroid* parity. A generalized matroid is a subset S of a hypercube such that if $X, Y \in S$ and X, Y differ at least in the ith bit, then either X', obtained from X by changing the ith bit, is in S, or there is some bit $j \neq i$ where X and Y also differ such that X'_j, obtained from X' by changing bit j, is in S. Notice that in the uniform case, with all elements of S having the same number of 0s and 1s, generalized matroids are matroids. In particular the one-in-three relation is a generalized matroid. An interesting class of generalized matroids is the *coindependent sets*, sets whose complement in the hypercube forms an independent set. This includes all clauses (only one excluded vertex), the not-all-equal relation (two opposite excluded vertices), linear equations modulo 2 (even or odd vertices). We show that the graph constraint satisfaction problem for coindependent sets is polynomially solvable.

The basic approach assumes that the constraints do not belong to Schaefer's three polynomial cases, so that all clauses can be represented. If in addition the relation $\{(x, y, z) : x = y = z\}$ can be represented, then this is just satisfiability, which is NP-complete. There is another case shown to exhibit fanout, where the relation accepting three copies 0 or copies of 1 is *not* obtained; the relation obtained is such that a 1 anywhere in the first half of the elements, implies a 0 everywhere in the second half, and viceversa: the two halfs can then be plugged into positive and negative occurences of literals in clauses, respectively, to get NP-completeness. The cases where we cannot get such a proof of NP-completeness are precisely the generalized matroids, and the problem where each variable is constrained twice by the generalized matroids is the parity problem.

We characterize the cases that do not fit in Schaefer's classification in the bipartite case in terms of generalized matroid intersection, where the two sides of the bipartite constraint graph are generalized matroids, in section 6. We then assume that $\{(x,y) : x = y\}$ cannot be simulated, since otherwise bipartiteness can be dropped. If $\{(x,y) : x \leq y\}$ cannot be simulated either, so that the exchange 00 to 11 does not occur in the definition of generalized matroids, then the bipartite case reduces to a matroid intersection problem, which is polynomially solvable.

Thus, a large number of discrete structures and computational problems that have been studied arise naturally in trying to understand the boundary between polynomiality and NP-completeness. That is, none of the problems studied refers explicitly to chordal graphs, Datalog, group theory, stable matching, graph matching, matroid intersection, matroid parity, etc.; these notions arise directly from the study of computational feasibility with and without fanout.

2 Constraint Satisfaction

We start with some definitions. A *vocabulary* is a set $V = \{(R_1, k_1), \ldots, (R_t, k_t)\}$ of relation names and their arities. A *relational structure* over the vocabulary V is a set S together with relations R_i of arity k_i on the set S. An instance of constraint satisfaction is given by a pair I, T of finite relational structures over the same vocabulary. The instance is satisfied if there is a homomorphism from I to T, that is, there exists a mapping h such that for every tuple $(x_1, \ldots, x_k) \in R_i$ in I we have $(h(x_1), \ldots, h(x_k)) \in R_i$ in T. Intuitively, the elements of I should be thought of as variables and the elements of T should be thought of as possible values for the variables. The tuples in the relations of I and T should be viewed as constraints on the set of allowed assignments of values to variables. The set of allowed assignments is nonempty iff there exists a homomorphism from I to T. In what follows, we shall use the homomorphism and variable-value views interchangeably in defining constraint satisfaction problems.

It is well-known that the constraint satisfaction problem is NP-complete. In practice, however, one often encounters the situation where the structure T (which we call the *template*) is fixed and it is only the structure I (which we call the *instance*) that varies. We consider here constraint satisfaction problems with respect to fixed templates.

It will sometimes be convenient to assume that for every element e in the template the relation $\{e\}$ is also in the template. That is, the homomorphism problem is a retract problem; this can be ensured by making the template a core; we define retractions, retracts, and cores as follows. Let S and T be two finite relational structures over the same vocabulary. A *homomorphism* from S to T is a mapping from the elements of S to elements of T such that all elements related by some relation R_i in S map to elements related by R_i in T, as described above. If T is the substructure of S obtained by restricting the relations to a subset of the elements of S, and the homomorphism h from S to T is just the identity mapping when restricted to T, then h is called a *retraction*, and T is called a *retract* of S. If no proper restriction T of S is a retract of S, then S is a *core*, otherwise its core is a retract T that is a core. It is easy to show that the core of a structure S is unique up to isomorphism. Now if a constraint-satisfaction problem has template S, and T is a retract of S, then the constraint-satisfaction problem with template T is polynomially equivalent to the constraint-satisfaction problem with template S. We may thus assume that template T is a core. If an instance I contains a copy of T, then this copy must map to T by a unique mapping up to isomorphism. That is, once the template is a core, the relations $\{e\}$ may be added to the template without affecting the complexity of the problem.

The following result is due to Schaefer.

Theorem 1 *A boolean constraint satisfaction problem whose template is a core is NP-complete unless it belongs to one of the following three categories:*

(1) Horn clauses (problems where every relation in the template can be characterized by a conjunction of disjunctive clauses with at most one positive literal per clause), and similarly anti-Horn clauses (with at most one negative literal per clause);

(2) 2SAT (problems where every relation in the template can be characterized by a conjunction of clauses with two literals per clause);

(3) linear equations modulo 2 *(problems where every relation in the template is the solution set of a system of linear equations modulo* 2*).*

Feder and Vardi attempted to obtain a similar classification when the template is not necessarily boolean. A constraint satisfaction problem is said to have *bounded width* if its complement (i.e., the question of non-existence of a solution) can be expressed in Datalog. More precisely, it is said to have *width* (l, k) if the corresponding Datalog program has rules with at most l variables in the head and at most k variables per rule, and is said to have *width* l if it is has width (l, k) for some k. It turns out that if a constraint satisfaction problem has width (l, k), then there is a canonical Datalog program that solves it. The approach is as follows: from the constraints in the instance, for every subset of k variables, infer constraints; obtain constraints for subsets of l variables, by projecting the constraints on subsets of k variables; obtain again constraints for subsets of k variables by intersecting all the constraints for subsets of l variables contained in k chosen variables; iterate. When this constraint inference procedure terminates, the instance must have no solution if and only if the empty set is inferred. A constraint satisfaction problem is said to have *strict width* $l \geq 2$ if it has width l, all relations in the template involve at most l variables, and after the above inference procedure terminates, if the empty set is not obtained, then a solution can be found by greedily assigning values to the variables in any order while satisfying the inferred constraints on l variables at a time.

Schaefer's approach to the classification is by means of closure properties. Feder and Vardi generalize Schaefer's closure properties beyond the boolean case:

(1) A problem has width 1 if and only if there is a function f that maps nonempty subsets of the template to elements of the template, such that for every relation R in the template, of arity k, the following holds. Let $S_1, S_2, \ldots, S_k$ be nonempty subsets with the property that for every x_i in S_i, there exist x_j in S_j for $j \neq i$ such that

$(x_1, x_2, \ldots, x_k)$ is in R. Then $(f(S_1), f(S_2), \ldots, f(S_k))$ is in R. (There is also a slightly more general case, called *extended width 1*, that has a similar characterization.) The existence of solutions can also be described in terms of a notion called tree duality. For Horn clauses, we have $f(\{0\}) = 0$, $f(\{1\}) = 1$, and $f(\{0,1\}) = 0$. For example, the clause $\overline{x} \vee \overline{y} \vee z$ has $(1,1,1)$ and $(0,1,0)$ among its satisfying assignments, so it must also have $(f(\{0,1\}), f(\{1\}), f(\{0,1\})) = (0,1,0)$ as a satisfying assignment. Similarly, anti-Horn clauses have $f(\{0,1\}) = 1$ instead.

(2) A problem has strict width l if and only if there is a function g that maps $(l+1)$-tuples from the template to elements of the template, such that for every relation R in the template, of arity k, the following holds. First, if all but at most one of some $l+1$ elements x_i are equal to some specific element x, then $g(x_1, x_2, \ldots, x_{l+1}) = x$. Second, let x_{ij} be elements with $(x_{1j}, x_{2j}, \ldots, x_{kj})$ in R for every j. Then $(g(x_{11}, x_{12}, \ldots, x_{1(l+1)}), g(x_{21}, x_{22}, \ldots, x_{2(l+1)}), \ldots, g(x_{k1}, x_{k2}, \ldots, x_{k(l+1)}))$ is in R. The existence of solutions can also be described in terms of a notion called the l-Helly property. For 2SAT, which has strict width 2, we have $g(0,0,0) = g(1,0,0) = g(0,1,0) = g(0,0,1) = 0$ and $g(1,1,1) = g(0,1,1) = g(1,0,1) = g(1,1,0) = 1$. For example, the clause $x \vee y$ has $(0,1)$, $(1,1)$, and $(1,0)$ among its satisfying assignments, so it must also have $(g(0,1,1), g(1,1,0)) = (1,1)$ as a satisfying assignment.

(3) A subroup problem is a problem with a group operation on the elements of the template T such that every relation R, of arity k, is a coset in the direct product T^k. A problem is a subgroup problem if and only if we can define a group operation on the elements of the template such that for every relation R in the template, of arity k, the following holds. If the three tuples $(b_1, b_2, \ldots, b_k)$, $(b_1x_1, b_2x_2, \ldots, b_kx_k)$, $(b_1y_1, b_2y_2, \ldots, b_ky_k)$ are in R, then the tuple $(b_1x_1y_1, b_2x_2y_2, \ldots, b_kx_ky_k)$ is also in R. If we write $a = bx$ and $c = by$, then $bxy = ab^{-1}c$. For linear equations modulo 2, the group is Z_2, and the operation $ab^{-1}c$ is just $a + b + c$ modulo 2. For example, the equation $x + y + z = 1$ modulo 2 has $(0,0,1)$, $(0,1,0)$, and $(1,1,1)$ among its satisfying assingments, so it must also have $(0+0+1, 0+1+1, 1+0+1) = (1,0,0)$ as a satisfying assignment.

It is worth noting that convex programming, a constraint-satisfaction problem over the reals, also has a componentwise closure property, namely the mappings $h_\alpha(x,y) = \alpha x + (1-\alpha)y$ for $0 \leq \alpha \leq 1$.

3 Network Stability and Limiting Fanout

To study the effect of fanout limitations, Mayr and Subramanian defined network stability. We describe the work on network stability in the context of constraint satisfaction in more detail here. Say that a template is *functional* if all relations have some arity $k + l$ with $k, l \geq 0$ and are described by a function $f(x_1, x_2, \ldots, x_k) = (y_1, y_2, \ldots, y_l)$, called a *gate*, where the x_i are called *inputs* and the y_i are called *outputs*. A *network stability* problem is a constraint-satisfaction problem where the template is functional, and where the input structure has the property that every element participates in exactly two relation occurrences, one as an input and one as an output. The input structure is then called a *network*. For e in the template, call $f() = e$ the *constant* gates, call $f(x) = ()$ the *absorption* gate, and call $f(x) = (x, x)$ the COPY gate.

Theorem 2 *Every network stability problem over a boolean functional template containing the constant and absorption gates is NP-complete, with the exception of the following polynomially solvable cases:*

(1) Monotone functional templates, where every output y_j of every gate is a monotone function of the inputs x_i. For the general case with AND, OR, and COPY gates, the problem is P-complete and determining whether there is a solution other than the zero-most and one-most solutions is NP-complete.

(2) Linear functional templates, where every output y_i of every gate is a linear function of the inputs x_i modulo 2.

(3) Adjacency-preserving functional templates, where every gate f has the property that changing the value of just one of the x_i inputs can affect at most one of the y_j outputs. Here the set of solutions can be described by a 2SAT instance because the median of three solutions, obtained by taking coordinate-wise majority, is also a solution.

The above classification was obtained by Mayr and Subramanian, with the exception of the polynomiality of the adjacency-preserving case, which is due to Feder. Mayr and Subramanian showed polynomiality for a subcase of the adjacency-preserving case, the *scatter-free* case, which contains stable matching as a main example. The NP-completeness remark in (1) and the 2SAT remark in (3) are due to Feder.

Mayr and Subramanian showed that the adjacency-preserving property is equivalent to the inability to simulate the COPY gate; thus this case limits fanout.

Feder attempted to generalize these three cases beyond the boolean case. The resulting polynomial cases are as follows:

(1) Monotone functional templates, where there is a partial order $\leq$ with a bottom element on the elements of the template, so that for tuples X and Y of elements in the template, if $X \leq Y$, then $f(X) \leq f(Y)$, componentwise.

(2) All constraint-satisfaction problems can be encoded as network stability problems. The case of linear functional templates is such an encoding.

(3) Nonexpansive functional templates, by assuming that the template has an associated distance function on the elements satisfying the triangle inequality, such that for every gate f, if $f(x_1, x_2, \ldots, x_k) = (y_1, y_2, \ldots, y_l)$ and $f(x'_1, x'_2, \ldots, x'_k) = (y'_1, y'_2, \ldots, y'_l)$, then $\sum d(y_j, y'_j) \leq \sum d(x_i, x'_i)$. The functional template is then called nonexpansive and the associated network is metric; this case is also polynomially solvable. Here the structure of the set of solutions is a strict width 2 problem because the solutions form a 2-isometric subspace, where the corresponding g mapping is obtained with the imprint function, yielding the 2-Helly property, thus generalizing 2SAT and medians.

4 Schaefer's approach

We saw in the last section that Mayr and Subramanian, Feder limited fanout by assuming that each variable participates in two gates, one as an input, one as an output. We begin here the study of the case where fanout is limited by assuming that each variable participates in exactly two relations (no input-output restrictions). We call this *graph constraint satisfaction*. If the constraints can be partitioned into two sets so that a variable participates in only one constraint from each set, we call this a *bipartite constraint satisfaction* problem. We generalize Schaefer's Theorem 3.0. For e an element of the template, call $x = e$ the *constant* relation.

It will sometimes be convenient to assume that variables can be constrained only by a single relation, instead of two relations. This can be achieved by making two copies of an instance, and linking the corresponding variables that are constrained by a single relation.

Given the relations in a template, we can represent other relations by giving an instance of the problem, with every variable in at most two constraints, finding all solutions and then dropping the variables that are constrained twice, thus giving a relation on the variables that are constrained only once. In the bipartite case, we require that the variables that are kept have their only constraint in the same side of the bipartite constraint graph.

Theorem 3 *Suppose that a boolean graph constraint satisfaction problem containing the constant relations does not fall into one of Schaefer's three polynomial cases. Then all clauses can be represented with this template. In the bipartite case, all monotone clauses can be represented.*

Proof. We establish a series of facts based on Schaefer's proof.

Fact 1: Suppose there is a relation that is not closed under the f mapping $f(\{0\}) = 0$, $f(\{1\}) = 1$, $f(\{0,1\}) = 0$. Then either $x \vee y$ or $x \neq y$ can be represented. If we use instead $f(\{0,1\}) = 1$, then either $\overline{x} \vee \overline{y}$ or $x \neq y$ can be represented.

We prove the first statement; the proof of the second statement is similar. Let V be the set of variables, and for each $v \in V$, let S_v be the corresponding set. So $v = f(S_v)$ for all $v \in V$ is not a satisfying assignment. If $S_v = \{0\}$, we set $v = 0$ and drop v. Similarly, if $S_v = \{1\}$, we set $v = 1$ and drop v. So we may assume $S_v = \{0,1\}$ for all v. Then the assignment that sets $v = 0$ for all $v \in V$ is not a satisfying assignment. We thus have a collection of satisfying assignments X_i, such that if W_i is the set of variables set to 0 in X_i, then the union of the W_i is V. There is no W_i that contains all W_j, because otherwise W_i would set all $v = 0$, and this is not a satisfying assignment. We may thus assume that there are two sets W_1 and W_2 which do not contain each other and which are maximal, in the sense that no W_i properly contains W_1 or W_2. Set the variables that are in both W_1 and W_2 to 0 and drop them, and similarly set the variables that are neither in W_1 nor in W_2 to 1 and drop them. Now W_1 and W_2 are nonempty and disjoint, with $W_1 \cup W_2 = V$. If one of W_1 and W_2 has more than one variable, say W_1, then we may drop one of the variables in W_1 and reduce the number of variables in V; setting all $v = 0$ is still not a satisfying assignment by maximality of W_2. If $W_1 = \{x\}$ and $W_2 = \{y\}$, then the condition on x, y is either $x \vee y$ or $x \neq y$, proving fact 1.

Fact 2: Suppose $x \neq y$ can be represented. Suppose also that there is a relation that is not closed under the mapping $h(x,y,z) = x + y + z$ modulo 2. Then every two-literal clause can be represented.

It suffices to represent some two-literal clause, since all other two-literal clauses can then be obtained using $x \neq y$ for complementation. Since $x \neq y$ is available, we may assume by complementation that the assignment obtained by applying h that is not a satisfying assignment sets $v = 0$ for all $v \in V$. We thus have three satisfying assignments X, Y, Z on V such that $X + Y + Z = 0$, the all-zero assignment, is not a satisfying assignment. Set the variables that equal 0 in all three X, Y, Z to 0 and drop these variables. We may thus partition V into three sets W_1, W_2, W_3 where the values for the assignments X, Y, Z respectively are $0, 1, 1$ for W_1, $1, 0, 1$ for W_2, and $1, 1, 0$ for W_3. Notice that at most one W_i is empty. Setting one variable $x = 1$ and all other variables $v = 0$ can be assumed to be a satisfying assignment, otherwise we could drop x and reduce the number of variables. There are necessarily two W_i, say W_1 and W_2, such that $|W_1| + |W_2|$ is even. We may assume inductively that there is no counterexample to the h closure property involving fewer variables. As a result, setting an odd number of variables in $W_1 \cup W_2$ to 1 is a satisfying assignment. Set the variables in W_3 to 0 and drop them. The assignment $v = 1$ for all $v \in V = W_1 \cup W_2$ is the satisfying assignment Z. Since $|V|$ is even, setting one variable x in V to $x = 0$ and all other variables to $v = 1$ is also a satisfying assignment, by the condition on an odd number of variables. If the assignment T that sets two $x, y = 0$ and all other $v = 1$ is not a satisfying assignment, then setting these $v = 1$ and dropping them gives $x \vee y$. We may thus assume that T is a satisfying assignment. Notice that T has an even number of 1s but two 0s. Iterating this construction, the number of 0s in satisfying assignments becomes $2, 4, 6, \ldots$ until the all-zeros assignment is obtained as a satisfying assignment, a contradiction. This proves fact 2.

Fact 3: If there are relations that are not closed for the two f's of fact 1 and for the h of fact 2, then every two-literal clause can be represented.

This is immediate. Either $x \neq y$ can be represented and the claim follows from fact 2, or it cannot, in which case by fact 1 both $x \vee y$ and $\overline{y} \vee \overline{z}$ can be represented, and from these two $x \vee \overline{z}$ can be represented.

Fact 4: If every two-literal clause can be represented, and there is a relation that is not closed under the majority mapping $g(x, y, z)$, then every clause can be represented.

It suffices to represent a single three-literal clause, since all other clauses can be obtained by resolution using the two-literal clauses. For notational convenience, we complement some of the variables so that the three satisfying assignments X, Y, Z such that T is not a satisfying assignment for $t_i = g(x_i, y_i, z_i)$ are such that all $t_i = 0$. If $x_i = y_i = z_i = 0$, then set variable i to 0 and drop it. We thus have three nonempty and disjoint sets of variables W_1, W_2, W_3 such that $W_1 \cup W_2 \cup W_3 = V$, X sets W_1 to 1, Y sets W_2 to 1, and Z sets W_3 to 1. Suppose that some W_i has more than one element, say W_1. Then choose x in W_1, and consider the assignment that sets $x = 1$ and all other $v = 0$. If this assignment is not a satisfying assignment, then we may drop x and reduce the number of variables. If it is a satisfying assignment, then we may set all variables in $W_1 \setminus \{x\}$ to 0 and drop them, reducing again the number of variables. We may thus assume W_1, W_2, W_3 each contain only one variable, say x, y, z respectively. For (x, y, z), we have three satisfying assignments $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, but $(0, 0, 0)$ is not a satisfying assignment. Then writing the two-literal clauses $x \leq x'$, $y \leq y'$, $z \leq z'$ gives the three-literal clause $x' \vee y' \vee z'$, proving fact 4 and completing the proof of the nonbipartite case.

In the bipartite case, as in fact 2, some two-literal clause can be represented. If this is $x \vee \overline{y}$, then as in fact 4, some three-literal clause can be represented, so $x \vee y$ or $\overline{x} \vee \overline{y}$ can be represented. So one of these two clauses can be represented, say $x \vee y$. Then as in fact 1, $\overline{x} \vee \overline{y}$ or $x \neq y$ can be represented, so in either case $\overline{x} \vee \overline{y}$ can be represented as well. As in fact 3, some three-literal clause can then be represented, which is either monotone or can simulate $x \vee \overline{y}$ to make it monotone, so all monotone clauses can be simulated. $\square$

5 Graph constraint satisfaction: matroid parity

Recall the definition of generalized matroids and coindependent sets from the introduction.

Theorem 4 *All relations represented by a generalized matroid parity instance are also generalized matroids (we view this as evidence that the fanout needed for NP-completeness cannot be obtained). A graph constraint satisfaction problem that contains a constraint that is not a generalized matroid falls into Schaefer's polynomial cases or is NP-complete. Thus the only new problems are the generalized matroid parity problems. The special case of coindependent sets is polynomially solvable.*

Proof. We prove the first statement. All generalized matroids in an instance can be viewed as a single generalized matroid, by taking their disjoint union. We just need to show that matching a pair of elements gives a generalized matroid on the remaining elements. So suppose R is a generalized matroid with elements $(x_1, x_2, \ldots, x_k)$. New relations are defined by matching elements, that is, requiring disjoint pairs of elements to be equal. Suppose x_1 and x_2 are matched. Suppose we have some X, Y in R with $x_1 = x_2$, that differ at least in some x_i for $i \geq 3$. Either changing x_i in X gives an X' in R, by the first case of the definition of generalized matroids, in which case we are done, or X' is not in R. In the second case, there is some other x_j where X and Y differ such that X'_j obtained from X' by changing x_j is also in R. If $j \geq 3$, we are again done. If $j \leq 2$, say $j = 1$, then X, Y also differ in x_2, so we can change x_2 in X'_1, to obtain X'_{12}. If X'_{12} is in R, we are done. Otherwise there is some $j \geq 3$ where X'_1 differ such that X'_{12j}, obtained by changing x_j, is in R, and we are done.

We prove next the second statement. Recall from theorem 3 that we can assume that all clauses (disjunctions of positive and negative literals) are present. This will justify complementing variables as needed. We can therefore assume that the counter-example to the generalized matroid property is given by the all-zeros X and the all-ones Y in a relation R, which differ in the first bit, and we are trying to change that bit from X, but neither X_1, obtained by changing the first bit, nor any X_{1j} obtained by changing also the jth bit, with $j \geq 2$, is in R. Thus changing x_1 to a 1 forces at least two more 1's. Let K be a minimal set of x_j containing $j = 1$ such that the Z with 1's in K and 0s elsewhere is in R. Notice that $|K| \geq 3$. We may then set the x_j not in K to 0 and and assume that K is all the variables. Then setting x_1 to 1 implies all x_j are 1, to be in R. We then drop all but the first three x_i.

This gives R as follows. 000 and 111 are in R, 100, 110, and 101 are not in R. Possibly $A = 010$, $B = 001$, and $C = 011$ are in R. Suppose none of A, B, C are in R. Then R is the relation $x = y = z$, giving the fanout needed for NP-completeness, when combined with the clauses available. Suppose only one of A, B, C is in R. Then R (up to complementation and permutation) is the relation $x \leq y = z$. Adding two-literal clauses, we get $x_1 \leq y_1 = z_1 \leq x_2 \leq y_2 = z_2 \leq \cdots \leq x_k \leq y_k = z_k \leq x_1$, with the y_i participating in only one constraint. This forces all y_i to be equal, giving fanout again. Suppose next that B and C are present, or A and C are present. Then R is $x \leq y \leq z$ and the preceding argument making equal y_i gives fanout again.

We are thus left with the case where just A and B are present, or all three A, B, C are present. In the second case, R is the relation $x \leq y, z$. In the first case, R is the relation $x = \min(y, z)$. This can be reduced also to $x \leq y, z$ by writing $x = \min(y', z')$ and $y' \leq y$ and $z' \leq z$.

So R is given by $x \leq y, z$. Recall that we have all clauses. We claim that we can make variables participate in three clauses. We may assume that such a variable appears both as a positive and as a negative literal. Suppose the clauses for variable x are $x \vee t$, $\overline{x} \vee u$, and $\overline{x} \vee v$, where t, u, v are disjunctions of literals. We may then write instead $x \vee t$, $\overline{y} \vee u$, and $\overline{z} \vee v$, with $x \leq y, z$; the equivalence of this formulation is immediate. So every variable can occur three times; to make it occur k times, introduce k variables x_i, with three copies of each, and write $x_1 \leq x_2 \leq \cdots \leq x_k \leq x_1$, forcing all x_i to be equal with only two occurences of each x_i, so the k variables x_i can be used as the same variable in an instance. This gives fanout and NP-completeness.

For the coindependent set problem, suppose all constraints have been assigned satisfying values, treating all occurences of variables as separate but paired up. Suppose some pair is unmatched, it has different values in its two occurences. This value can be fixed, but may require changing any other value in the same constraint, creating a different unmatched pair. Iterating, the problem reduces to finding a path in a graph, and only how the final constraint in the path is visited matters. □

6 Bipartite constraint satisfaction: matroid intersection

Theorem 5 *In the bipartite case, the problems that are not among Schaefer's polynomial or NP-complete cases, are characterized by generalized matroid intersection. If neither $x = y$ nor $x \leq y$ can be simulated, the problem can be solved in polynomial time by matroid intersection.*

Proof. The proof follows the lines of the graph constraint satisfaction problem. We assume by theorem 3 that all monotone clauses can be simulated. If $x = y$ can be simulated, then this reduces to the non-bipartite case. Complement variables as needed, just to describe the possible cases. As in theorem 4, in the cases where only 000 and 111 appear, or $x \leq y = z$, or $x \leq y \leq z$ is obtained, then $x \leq y$ or $x = y$ can be simulated, even without complementation, and the proof goes through. We are thus left with the case $x \leq \overline{y}, \overline{z}$, and the proof of NP-completeness goes through by simulating $s \vee \overline{t}$ with $s \vee x$ and $\overline{x} \vee \overline{t}$, making all clauses monotone, with positive clauses in one side and negative clauses in the other, proving the first statement.

When neither $x = y$ nor $x \leq y$ can be simulated, we show that if $X = 0^k 1^l$ and $Y = 1^k 0^l$ with $k \leq l$, are in the generalized matroid, then after changing the first 0 bit in X to 1, it must be possible to change some 1 bit to 0 and stay in the matroid; furthermore, when $k < l$ some 1 bit in Y can be changed to 0 and stay in the generalized matroid. As a result, adding additional bits to ensure that the number of 0s and 1s is always the same, the generalized matroid becomes a matroid, giving matroid intersection which is polynomially solvable. We prove the first observation. If no 1 bit can be changed, then $X' = 10^{k-1}1^l$ is in the matroid but none of the $10^{k-1}1^{j-1}01^{l-j}$ is in the generalized matroid. Since $x = y$ cannot be simulated, no $10^{k-1}x$ with $x \neq 1^l$ is in the matroid. However, if from X', we change the k 0s into 1s, then at most k 1s change to 0s, a contradiction. We prove the second observation. From X, by changing the first $k-1$ 0s to 1s, some $k-1$ 1s change to 0s, giving some Z which compared to Y completes the proof. □

7 Conclusion

We have shown that when fanout is limited by requiring that each variable appear in only two constraints, then the only new problems that arise, beyond Schaefer's classification of the boolean case without fanout limitations, have all relations given by generalized matroids. That is, we have the parity problem for generalized matroids in the case of graph constraint satisfaction, and the intersection problem for generalized matroids in the bipartite case.

We have only been able to show that these new problems are polynomially solvable in restricted cases, so their complexity is in general open. We do not know if the assumption that the generalized matroids are the disjoint union of constant size matroids, because the constraint types are given by a fixed template, will make the problems more easily solved with polynomial time algorithms.

References

[1] T. Feder, "Stable Networks and Product Graphs," doctoral dissertation, Stanford University (1991), also Memoirs of the Amer. Math. Soc. vol 555 (1995), 223 pp.

[2] T. Feder and P. Hell, "List homomorphisms to reflexive graphs," J. Comb. Theory Series B 72 (1998), 236–250.

[3] T. Feder and M. Vardi, "The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory," SIAM J. on Computing 28 (1998), 57–104.

[4] P. Hell and J. Nešetřil, "On the complexity of H-coloring," J. Comb. Theory, Series B 48 (1990), 92–110.

[5] P.Jeavons, D.Cohen, and M. Gyssens, "Closure properties of constraints," J. Assoc. Comput. Mach. 44 (1997), 527–548.

[6] R. E. Ladner, "On the structure of polynomial time reducibility," J. Assoc. Comput. Mach. 22 (1975), 155–171.

[7] L. Lovász, "Matroid matching and some applications," J. Combinatorial Theory Ser. B 28 (1980), 208–236.

[8] E. Mayr and A. Subramanian, "The complexity of circuit value and network stability," J. Comput. Syst. Sci. 44 (1992), 302–323.

[9] T. J. Schaefer, "The complexity of satisfiability problems," Proc. 10th ACM Symp. on Theory of Computing (1978), 216–226.