

Constraint Satisfaction Problems: Convexity Makes AllDifferent Constraints Tractable

Michael Fellows¹, Tobias Friedrich², Danny Hermelin², Nina Narodytska³, Frances Rosamond¹

¹ Charles Darwin University, Darwin, Northern Territory, Australia. firstname.lastname@cdu.edu.au

² Max-Planck-Institut für Informatik, Saarbrücken, Germany. firstname.lastname@mpi-inf.mpg.de

³ NICTA and University of New South Wales, Sydney, Australia. ninan@cse.unsw.edu.au

Abstract

We examine the complexity of constraint satisfaction problems that consist of a set of AllDiff constraints. Such CSPs naturally model a wide range of real-world and combinatorial problems, like scheduling, frequency allocations and graph coloring problems. As this problem is known to be NP-complete, we investigate under which further assumptions it becomes tractable. We observe that a crucial property seems to be the convexity of the variable domains and constraints. Our main contribution is an extensive study of the complexity of Multiple AllDiff CSPs for a set of natural parameters, like maximum domain size and maximum size of the constraint scopes. We show that, depending on the parameter, convexity can make the problem tractable while it is provably intractable in general.

1 Introduction

Constraint satisfaction is a general framework that allows structure preserving encoding of many real world problems, including formal verification, vehicle routing and scheduling. A CONSTRAINT SATISFACTION PROBLEM (CSP) consists of a set of variables that must be assigned values from their respective domains in such a way that a set of constraints is satisfied. In general, finding a solution of a CSP is an NP-complete problem. Hence, much research has been devoted to finding restricted classes of CSPs that admit polynomial time algorithms. This area of research includes two main directions. The first direction exploits structural properties of relations between variables and constraints of a CSP instance [11, 15]. The second direction investigates limitations on the types of constraints in the CSP constraint language, so called tractable constraint languages [20, 34].

In this work we continue this line of research and investigate CSPs under both structural restrictions and limitations on constraint types. Such combinations allow us to identify interesting classes of CSPs that have expressive constraint languages and are still tractable under some natural conditions on relations between variables and constraints in the framework of parameterized complexity.

We restrict the type of allowed constraints to the most common constraint called the AllDiff constraint [25, 31]. The

AllDiff constraint requires a set of variables to take pairwise distinct values. We call a CSP with the language that consists of only AllDiff constraints the MULTIPLE ALLDIFF CONSTRAINT SATISFACTION PROBLEM (MAD-CSP). Finding a solution of the MAD-CSP is NP-complete due to a straightforward reduction from a graph coloring problem.

Our structural limitations of MAD-CSPs are inspired by a large body of research on global constraints in constraint programming. We consider a class of MAD-CSPs that possess the *convexity of constraints* property. This property means that the variables are linearly ordered so that all AllDiff constraints are defined over intervals of variables in this order. Such MAD-CSPs can be seen as a generalization of the SLIDE constraint where AllDiff is the relation to slide [3]. We also investigate another restriction on CSPs that comes from work on global constraints over variables with interval domains that we call *convex domains*. This means that every domain constitutes an interval of values. It is interesting to consider the convex domain restriction in the context of MAD-CSPs, as it has proved useful for constructing efficient inference algorithms for global constraints. Many global constraints, like NValue [2] or overlapping AllDiff [4], that are NP-hard to reason about become polynomially solvable if the domains of the variables are intervals of values. We further say that a MAD-CSP instance is *biconvex* if it has both convex domains and convex constraints.

To motivate the AllDiff constraint and our notion of convexity, consider the following three examples of real-world and combinatorial problems that can be naturally encoded as MAD-CSP problems:

- The time table of a workshop on CSP might be modeled with one variable per speaker with the domain of the variables corresponding to the availability of the speakers [30, 37]. For example, speakers A and B might be available during the first and third session and speaker C during all sessions. We introduce three variables S_1 , S_2 and S_3 with domains $D(S_1) = \{1, 3\}$, $D(S_2) = \{1, 3\}$ and $D(S_3) = \{1, 2, 3\}$. One (convex) AllDiff constraint

¹We know that this case is FPT for several aggregate parameters like domain size and the treewidth of the constraint graph (Lemma 5.2), domain size and the treewidth of the domain graph (Lemma 5.3), and domain size and the number of constraints (Theorem 5.5), but for the single parameter domain size it remains open. See Section 5 for details.

parameter	$n = \text{number of variables}$	$m = \text{size of the universe}$	$k = \text{maximum size of constraint scopes}$	$\ell = \text{maximum domain size}$
no convexity	FPT (Thm. 3.1)	para-NPC (Lem. 2.1)	para-NPC (Lem. 2.1)	para-NPC (Lem. 2.1)
convex domains	FPT (Thm. 3.1)	para-NPC (Lem. 2.1)	para-NPC (Lem. 2.1)	para-NPC (Lem. 2.1)
convex constraints	FPT (Thm. 3.1)	FPT (Thm. 4.5)	FPT (Thm. 6.2)	para-NPC (Lem. 5.1)
bi-convex	FPT (Thm. 3.1)	FPT (Thm. 4.5)	FPT (Thm. 6.2)	open ¹

Table 1: Overview of our results.

containing all variables encodes that there are no two talks at the same time. If the available times of the speakers are intervals, this MAD-CSP instance is biconvex.

- Scheduling of n jobs with given start and end times on m machines can be modeled as a MAD-CSP with one variable per job and values corresponding to machines on which the job can be run. Each job can be performed by a subset of machines. Suppose we have 3 jobs with the following time intervals: J_1 has to be performed in the interval $[1, 2]$ and can be assigned to one of the machines $\{1, 3\}$, J_2 in the interval $[2, 4]$ on one of the machines $\{1, 3\}$ and J_3 in the interval $[3, 5]$ on one of the machines $\{2, 3\}$. The domains of variables J_i are as follows: $D(J_1) = D(J_2) = \{1, 3\}$ and $D(J_3) = \{2, 3\}$. Convex AllDiff constraints then ensure that each machine only runs one job per time slot: AllDiff $[J_1, J_2]$ and AllDiff $[J_2, J_3]$. Our results demonstrate that this scheduling problem is FPT parameterized by the total number of machines.
- The classical graph coloring problem where two adjacent vertices should get different colors can be described as a MAD-CSP with one variable per vertex and one AllDiff constraint per edge. If there is no restriction on the set of colors used, the domains are convex. If the graph is an interval graph, also the constraints are convex (see Theorem 4.1).

It should be noted that finding a solution of a MAD-CSP with convex domains and constraints is still an NP-complete problem due to a reduction from the list coloring of an interval graph problem (Section 4). Hence, we use the more fine-scaled framework of parameterized complexity [13, 14, 29] to analyze its complexity.

Our Results. We examine MAD-CSPs in several dimensions. First, the domains and/or constraints can be convex as observed in the examples above. Second, there are various natural parameters which might be fixed or bounded for typical problem instances. We selected four commonly used parameters: The number of variables, the number of values (universe size), the maximum size of the constraint scopes (arity), and the maximum domain size. From the theoretical point of view, our parameterizations allow us to reveal the core of the hardness of this problem. From the practical point of view, they can be used to construct efficient fil-

tering algorithms for the Multiple AllDiff constraint. Note that even though some of these parameters, like the number of variables or the number of values, may not be typically bounded in some applications, during the backtracking search of a constraint solver these parameters become bounded due to the solver’s branching and inference mechanisms (see *e.g.* [35, 36]). Hence efficient algorithms for bounded parameters are of high practical relevance.

Depending on the convexity of the domains and constraints, our results for the four chosen parameters are presented in Table 1. For the first parameter (number of variables), we give an FPT algorithm independent of the convexity (Theorem 3.1). For this case we can also show that there is a polynomial-sized problem kernel (Theorem 3.3). For the second and third parameters (size of the universe and maximum size of the constraint scopes), we give an FPT algorithm if the constraints are convex (Theorem 4.5 and 6.2). For most of the remaining cases we can prove that there is no FPT algorithm unless $P=NP$ (Lemmas 2.1 and 5.1). More precisely, we show that these cases are para-NPC, that is, NP-complete for some (small) fixed parameter. There is one case left open. This is the case of bi-convex MAD-CSP instances when parameterized by maximum domain size. We conjecture that this case is in FPT, but we can only prove this for the aggregate parameter domain size and number of constraints (Theorem 5.5). See Section 5 for details.

Our results also extend to another natural notion of convex constraints and domains where the variables and values are arranged on a tree and convexity corresponds to connected subtrees. We defer the discussion of this to the full version of the paper.

Related work. The AllDiff constraint first appeared in the ALICE constraint programming language [25]. The complexity of a single AllDiff constraint is examined in a series of papers by different authors [26, 30, 31]. Recently much research has been devoted to constructing efficient algorithms for conjunctions of AllDiff constraints [4, 23, 24]. Unfortunately, even for the conjunction of two AllDiff constraints a polynomial algorithm does not exist unless $P = NP$ [19, 23]. This motivates looking for tractable subclasses. Bessiere *et al.* [4] give a polynomial-time algorithm for a conjunction of two AllDiff constraints if the domains are convex. We extend this to multiple AllDiff constraints and explore different

parameters and restrictions for which the problem becomes tractable.

In the context of general CSPs, much research has been done on tractable constraint satisfaction problems. One line of this research deals with tractable constraint languages (e.g. [9, 20, 34]), another exploits structural properties of CSPs [11, 15, 27, 28]. Marx [27, 28] investigated general CSPs from the parameterized complexity point of view, and Samer and Szeider [33] identified relevant parameters in general constraint satisfaction problems.

It should be noted that the problem that we consider is a special case of the Same-Relation constraint studied in [21]. Jefferson *et al.* [21] consider a binary constraint satisfaction problem with an identical binary relation over each edge of the primal constraint graph and a set of unary constraints. They showed that such binary CSPs are intractable for many structures of constraint graphs (cf. Definition 2.1), like cliques, bipartite graphs, grids or directed acyclic graphs. In our work, we continue this research and identify additional restrictions on the Same-Relation constraint that make this constraint fixed parameter tractable.

In [1, 10, 22] the similar INTERVAL CONSTRAINED COLORING problem was studied. The objective of this problem is to assign a color to a set of variables such that a set of constraints is satisfied. Each constraint is a convex set of variables which specifies exactly the number of variables that belong to each color class. Despite the similarity, this problem is neither a generalization nor a specialization of MAD-CSPs.

Organization of the paper. We first formally define CSP and MAD-CSP in Section 2. In Sections 3 and 4 we examine the cases of fixed number of variables and values, respectively. Section 5 discusses fixed maximum domain size. In the last section, we study fixed maximum size of the constraint scopes. Note that although each section covers a different parameter, the proofs of Section 6 build on results from Section 5 which in turn depend on Section 4.

2 Formal Model

Formally, a CSP instance is a triple $(\mathcal{X}, \mathcal{U}, \mathcal{C})$, where $\mathcal{X} := \{x_1, \dots, x_n\}$ is a set of variables, $\mathcal{U} \subseteq \mathbb{N}$ is a set of possible values (also called the universe), and \mathcal{C} is a set of constraints. A constraint C is defined over a set of variables, that are called the *scope* of the constraint C , denoted $\text{scope}(C) \subseteq \mathcal{X}$. A constraint C can be represented extensionally as a table of valid (or invalid) assignments to variables in its scope or intensionally, by giving an expression or a formula involving the variables in the constraint scope. We consider only constraints represented extensionally. Each constraint in \mathcal{C} is a triplet (S, R, m) , where S is an m -tuple of variables and R is an arbitrary m -ary relation over \mathcal{U} (constraint relation). We will slightly abuse notation by writing $x \in C$ to indicate that variable $x \in \mathcal{X}$ is in the scope S of some constraint $C \in \mathcal{C}$. Also, we assume w.l.o.g. that every variable occurs in at least one constraint scope, and every domain element occurs in at least one constraint relation. A solution to a CSP instance is a function (assignment) τ from the set of variables \mathcal{X} to the set of values \mathcal{U} satisfying that for each constraint (S, R, m) with $S = (x_{i_1}, x_{i_2}, \dots, x_{i_m})$ the tuple

$(\tau(x_{i_1}), \tau(x_{i_2}), \dots, \tau(x_{i_m}))$ is a member of R . If there is such an assignment we say that the CSP instance is *satisfiable*, and otherwise we say it is *unsatisfiable*.

We will focus on CSP instances which include only AllDiff constraints, henceforth referred to as MULTIPLE ALL-DIFF CSP (MAD-CSP) instances. A MAD-CSP instance is a quadruple $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$, where \mathcal{X} and \mathcal{U} are defined as above, \mathcal{D} is a function assigning domains to variables, and \mathcal{C} is the set of AllDiff constraints. The function \mathcal{D} is defined from \mathcal{X} to $2^{\mathcal{U}}$, restricting variable $x \in \mathcal{X}$ to have values only from $\mathcal{D}(x)$. The set \mathcal{C} is a set of subsets of $2^{\mathcal{X}}$ enforcing that all variables occurring together in one constraint must be assigned different values. We assume w.l.o.g. that all MAD-CSP instances are in a “canonic” form, that is, the universe is $\mathcal{U} = \{1, 2, \dots, |\mathcal{U}|\}$, there are no values which are not used by any variable, there is no variable with an empty domain, there are no empty constraints, and there is no constraint which is a proper subset of another constraint. A solution to a MAD-CSP instance is a function $\tau: \mathcal{X} \rightarrow \mathcal{U}$ such that:

- (i) $\forall x \in \mathcal{X} : \tau(x) \in \mathcal{D}(x)$.
- (ii) $\forall C \in \mathcal{C}$ and $\forall x, y \in C : x \neq y \Rightarrow \tau(x) \neq \tau(y)$.

We are specifically interested in domains and constraints with a particular structure. We say that a domain $\mathcal{D}(x)$ of a variable x is *convex*, if $\mathcal{D}(x) = \{i, i+1, i+2, \dots, j\}$ for some $1 \leq i \leq j \leq m$. Analogously, a constraint $C \in \mathcal{C}$ is called *convex*, if $C = \{x_i, x_{i+1}, x_{i+2}, \dots, x_j\}$ for some $1 \leq i \leq j \leq n$. A MAD-CSP instance has *convex domains (constraints)* if all domains (constraints) are convex, and it is *bi-convex* if it has convex domains and constraints simultaneously.

Our work focuses on analyzing MAD-CSP instances under the framework of parameterized complexity. Readers are referred to [13, 14, 29] for relevant concepts and definitions. We consider the following parameters for MAD-CSP:

- Parameter $n := |\mathcal{X}|$ measuring the number of variables.
- Parameter $m := |\mathcal{U}|$ measuring the size of the universe.
- Parameter $k := \max_{C \in \mathcal{C}} |C|$ measuring the maximum size of the constraint scopes.
- Parameter $\ell := \max_{x \in \mathcal{X}} |\mathcal{D}(x)|$ measuring the maximum domain size.

It is not difficult to see that MAD-CSP instances can naturally model the GRAPH COLORING problem, where we are given a graph $G := (V, E)$, and an integer k , and the goal is to find a function (k -coloring) $f: V \rightarrow \{1, \dots, k\}$ with $f(u) \neq f(v)$ for all $\{u, v\} \in E$. To reduce GRAPH COLORING to MAD-CSP, we let $\mathcal{X} = V$, $\mathcal{U} = \{1, \dots, k\}$, $\mathcal{D}(x) = \mathcal{U}$ for all $x \in \mathcal{X}$, and $\mathcal{C} = E$. Then the MAD-CSP instance is satisfiable iff G has a k -coloring. Since GRAPH COLORING is known to be NP-complete for instances with $k \geq 3$ [16], we have the following hardness result for MAD-CSP.

Lemma 2.1. *MAD-CSP is NP-complete even when restricted to convex domain instances with $m = 3$, $k = 2$, and $\ell = 3$.*

We conclude this section by introducing the notion of a constraint graph, a concept playing an important role in many works regarding CSP [18, 28, 33]. After presenting a formal definition of this notion, we state an important theorem

by Gottlob et al. [18] relating the fixed-parameter tractability of CSP instances to a structural parameter of the constraint graph, namely the *treewidth* parameter (see e.g. [12] for a formal definition).

Definition 2.1. *The constraint graph of a CSP instance $(\mathcal{X}, \mathcal{U}, \mathcal{C})$ is the underlying graph of the hypergraph $(\mathcal{X}, \mathcal{C})$. That is, it is the graph with vertex-set \mathcal{X} , where two variables $x, y \in \mathcal{X}$ are connected by an edge iff there is a constraint $C \in \mathcal{C}$ with $x, y \in C$.*

Theorem 2.2 ([11]). *CSP can be solved in $m^{t+1} \cdot n^{O(1)}$ time and space, where t is the treewidth of the constraint graph.*

3 Fixed Number of Variables

We begin our discussion by considering the instances of MAD-CSP with a fixed number of variables n . We will present two parameterized algorithms for this parameter, both with asymptotically similar dependencies on the parameter, and both working even when neither the domains nor the constraints are convex. Note that general CSP is not FPT when parameterized by n as for example k -Clique can be expressed as a CSP with k variables [33].

Theorem 3.1. *MAD-CSP can be solved in $O(n!nm)$ time, independent of the underlying convexity assumption.*

Proof. We examine the following algorithm:

1. Try all permutations π of $\{1, 2, \dots, n\}$.
2. For i from 1 to n : assign $x_{\pi(i)}$ the smallest valid value from $\mathcal{D}(x_{\pi(i)})$.
3. If this is possible for all variables, return *yes*.
4. If there is no permutation for which a valid assignment is found, return *no*.

We can easily bound the worst-case runtime of this algorithm by $O(n!nm)$ which is FPT for the parameter being the number of variables n . It remains to prove correctness of the algorithm. It is clear that if the algorithm returns “yes”, it has found a valid assignment and the answer is correct. To show that it is also correct if it returns “no”, it is sufficient to prove the following:

- (1) If the given instance is satisfiable, then there is a permutation π of $\{1, 2, \dots, n\}$ such that the algorithm finds a valid assignment.

In order to prove claim (1), we first show the following claim.

- (2) If the given instance is satisfiable, then there is an assignment such that there is a variable x which is assigned to $\min \mathcal{D}(x)$.

To show claim (2), assume the contrary, that is, there is a satisfiable instance where in all valid assignments no variable x is assigned to $\min \mathcal{D}(x)$. In this case, take any valid assignment and observe that the value 1 cannot have been assigned to any variable. By the definition of our problem, there must be a variable x with value 1 in its domain. Reassigning x to 1 gives a valid assignment in which x is assigned to $\min \mathcal{D}(x)$. This proves claim (2).

It remains to show claim (1). Given a satisfiable instance, we use claim (2) and choose the first variable in the permutation π to be a variable x which is assigned to $\min \mathcal{D}(x)$ in

a valid assignment. To choose the second variable in the permutation, we reduce the given instance by the already fixed variable x . More precisely, we remove the variable x , the value $\min \mathcal{D}(x)$, and all values which are solely used by x from the instance. This new instance must also be satisfiable and claim (2) gives the second variable in the required permutation for claim (1). Going on the same way for the third, fourth, etc. variable, gives the desired permutation and proves claim (1). The theorem is thus proven. \square

Our second algorithm shows that the MAD-CSP problem is not only FPT when parameterized by the number of variables, it actually also has a polynomial kernel in this parameter (see e.g. [6] for a discussion on this important concept in parameterized complexity). Our kernelization algorithm is based on a simple reduction rule, which roughly states that if a variable has a large enough domain it can be deleted. The justification of this rule is given in the following simple lemma.

Lemma 3.2. *Let $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ be a MAD-CSP instance and $x \in \mathcal{X}$ be a variable with $|\mathcal{D}(x)| \geq n$. Then $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ is satisfiable iff $(\mathcal{X}', \mathcal{U}', \mathcal{D}', \mathcal{C}')$ is satisfiable, where $\mathcal{X}' := \mathcal{X} \setminus \{x\}$, $\mathcal{U}' := \bigcup_{x \in \mathcal{X}'} \mathcal{D}(x)$, $\mathcal{D}' := \mathcal{D}|_{\mathcal{X}'}$, and $\mathcal{C}' := \{C \setminus \{x\} : C \in \mathcal{C} \text{ and } C \neq \{x\}\}$.*

Using this rule iteratively until it can no longer be applied, we obtain a MAD-CSP instance where each variable has a domain size which is bounded by the total number of remaining variables in the instance. This immediately implies that the problem has a kernel of size $O(n^2)$.

Theorem 3.3. *The MAD-CSP problem has a kernel of size $O(n^2)$, independent of the underlying convexity assumption.*

4 Fixed Number of Values

In this section we consider the MAD-CSP problem when parameterized by the size of the universe m of the instance. We will show that MAD-CSP with convex constraints is FPT in this parameter. In contrast, as previously shown in Lemma 2.1, MAD-CSP with convex domains is para-NPC.

Before presenting our parameterized algorithm, we show that MAD-CSP with convex constraints is equivalent to the classical LIST COLORING PROPER INTERVAL GRAPHS (LCPIG) problem. A graph $G := (V, E)$ is an *interval graph* if there exists a set S of intervals on the real line and a bijection $I: V \rightarrow S$ such that $\{u, v\} \in E \iff I(u) \cap I(v) \neq \emptyset$ (see e.g. [17]). If no interval is properly contained in another in S , then G is a *proper interval graph*. In LCPIG, we are given a proper interval graph G , where each vertex v has a list of colors $L(v)$, and the goal is to assign a color to each vertex from its list such that adjacent vertices are assigned different colors.

To show that MAD-CSP with convex constraints is equivalent to LCPIG, we use an equivalent definition of proper interval graphs due to Roberts [32] who showed that G is a proper interval graph iff the adjacency matrix of G has the *consecutive 1's property*; that is, if there is an ordering of the vertices of G such that the adjacency matrix of G under this ordering has 1's appearing consecutively in each row and column. Due to the classical Booth and Lueker [8] algorithm, this ordering can be computed in linear time. It is easy to verify that

the adjacency matrix of the constraint graph associated with a MAD-CSP instance with convex constraints has the consecutive 1's property. Conversely, any graph whose adjacency matrix has the consecutive 1's property can be interpreted as a constraint graph of a MAD-CSP instance with convex constraints. We therefore obtain the following theorem which will also be useful in Section 5.

Theorem 4.1. *MAD-CSP with convex constraints is equivalent to the LCPIG problem.*

Now using that LCPIG is NP-complete even in the case where each list is an interval of consecutive colors [5, 7], we immediately get from Theorem 4.1 the following corollary.

Corollary 4.2. *MAD-CSP is NP-complete even in the bi-convex case.*

We next proceed to describe our parameterized algorithm for the parameter m . We first observe the following simple lemma that follows directly from the definition of the MAD-CSP problem.

Lemma 4.3. *If there is any constraint $C \in \mathcal{C}$ with scope greater than m , then the given MAD-CSP instance cannot be satisfied.*

We next use Lemma 4.3 above to bound the treewidth of our constraint graph. For this, we will use a slightly different graph invariant called *pathwidth*. The pathwidth of a given graph is bounded from below by its treewidth, and it is well-known that the graph has pathwidth k iff the smallest clique number of an interval graph that contains it as a subgraph is $k+1$ (see e.g. [12]). This result together with Lemma 4.3 and the fact that the constraint graph of a MAD-CSP instance with convex constraints is an interval graph, gives us the following Lemma 4.4.

Lemma 4.4. *If each constraint scope in a given MAD-CSP instance with convex constraints is of size at most k then the constraint graph of the instance has pathwidth at most $k-1$.*

We now can directly use Theorem 2.2 to obtain the main result of this section.

Theorem 4.5. *MAD-CSP with convex constraints can be solved in $m^m \cdot n^{O(1)}$ time.*

5 Fixed Maximum Domain Size

We next consider the parameter ℓ which measures the maximum number of values in any variable domain. Recall that by Lemma 2.1 we know that MAD-CSP with convex domains is para-NPC for this parameter. The following lemma shows that this also holds for convex constraints.

Lemma 5.1. *MAD-CSP with convex constraints is NP-complete for instances with $\ell \geq 3$.*

The proof of Lemma 5.1 uses a result of Jansen [19]. It also implies that MAD-CSP with convex constraints is para-NPC even when parameterized by both the maximum domain size and the number of constraints. We next show that for these two parameters, MAD-CSP turns out to be in FPT when the domains are convex. The complexity of the bi-convex case for the single parameter ℓ remains open. We begin with the following lemma, which we will also need in Section 6. Its proof is based on Theorem 2.2.

Lemma 5.2. *MAD-CSP can be solved in $\ell^{t+1} \cdot n^{O(1)}$, where t is the treewidth of the constraint graph, independent of the underlying convexity assumption.*

We next introduce a graph of MAD-CSP instances which is defined by considering the relationships between the domains of different variables. Following this, we will prove an analogous result to Lemma 5.2 for the case where the treewidth of the so-called domain graph is bounded.

Definition 5.1. *The domain graph of a MAD-CSP instance $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ is a graph with vertex set \mathcal{X} where two variables $x, y \in \mathcal{X}$ are adjacent iff $\mathcal{D}(x) \cap \mathcal{D}(y) \neq \emptyset$.*

Lemma 5.3. *MAD-CSP can be solved in $\ell^{t+1} \cdot n^{O(1)}$, where t is the treewidth of the domain graph, independent of the underlying convexity assumption.*

Our goal now is to use Lemma 5.3 above for showing that MAD-CSP is FPT when parameterized by both ℓ and $|\mathcal{C}|$. For this, we show that unless our given MAD-CSP instance is unsatisfiable, the domain graph of this instance has treewidth bounded by a function of both these parameters. The following lemma gives the first step in this direction.

Lemma 5.4. *Let $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ be an instance of MAD-CSP, and let D be a domain of some variable in \mathcal{X} . If there are more than $\ell \cdot |\mathcal{C}|$ variables $x \in \mathcal{X}$ with $\mathcal{D}(x) = D$, then $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ cannot be satisfied.*

Theorem 5.5. *MAD-CSP with convex domains can be solved in $\ell^{2c\ell^2} \cdot n^{O(1)}$ time, where $c := |\mathcal{C}|$ is the number of constraints in the given instance.*

Proof. We prove the theorem by applying Lemma 5.3. We first observe that in the case of convex domains, the domain graph G of a given MAD-CSP instance is an interval graph. This can be seen by assigning each variable its interval domain. Our algorithm begins by checking whether there is a domain D in our instance where the number of variables with this domain is more than $c\ell$. If this is the case, we report that the instance is unsatisfiable by Lemma 5.4. Otherwise, the number of interval domains which start or end at a certain value s , which is exactly the number of interval domains that are contained inside the interval $[s, s+l]$ or $[s-l, s]$, is at most $c\ell$. From this it follows that any domain intersects less than $2c\ell^2$ other domains, and therefore G has maximum clique-size less than $2c\ell^2$. Thus, G has treewidth at most $2c\ell^2 - 1$, and we can apply Lemma 5.3 to complete the proof. \square

6 Fixed Maximum Size of Constraint Scope

In this section we examine the parameter k which measures the maximum size of constraint scopes. Lemma 2.1 showed that MAD-CSP is para-NPC without convexity of the constraints. To show that convex constraints make the problem FPT with respect to k , we first argue that variables with domain size larger than k^2 can be safely removed.

Lemma 6.1. *Let $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ be an instance of MAD-CSP where all constraints $C \in \mathcal{C}$ are convex and all constraint scopes are of size at most k . Also, let $x \in \mathcal{X}$ be a variable with $\mathcal{D}(x) > k^2$. Then $(\mathcal{X}, \mathcal{U}, \mathcal{D}, \mathcal{C})$ is satisfiable*

iff $(\mathcal{X}', \mathcal{U}', \mathcal{D}', \mathcal{C}')$ is satisfiable, where $\mathcal{X}' := \mathcal{X} \setminus \{x\}$, $\mathcal{U}' := \bigcup_{x \in \mathcal{X}'} \mathcal{D}(x)$, $\mathcal{D}' := \mathcal{D}|_{\mathcal{X}'}$, and $\mathcal{C}' := \{C \setminus \{x\} : C \in \mathcal{C} \text{ and } C \neq \{x\}\}$.

This reduction rule gives us an FPT algorithm for convex constraints as follows.

Theorem 6.2. *MAD-CSP with convex constraints can be solved in $k^{2k} \cdot n^{O(1)}$ time.*

Proof. Given a MAD-CSP instance with maximum size of constraint scopes k , we know from Lemma 4.4 that the pathwidth, and therefore also the treewidth, of the constraint graph is at most $k - 1$. By above Lemma 6.1, we can reduce in polynomial time the given MAD-CSP instance to a MAD-CSP instance with maximum domain size k^2 . As the treewidth of the constraint graph remains bounded by $k - 1$, applying Lemma 5.2 finishes the proof. \square

References

- [1] E. Althaus, S. Canzar, K. M. Elbassioni, A. Karrenbauer, and J. Mestre. Approximating the interval constrained coloring problem. In *11th SWAT*, pages 210–221, 2008.
- [2] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Filtering algorithms for the NValue constraint. In *2nd CPAIOR*, pages 79–93, 2005.
- [3] C. Bessiere, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. SLIDE: A useful special case of the CARD-PATH constraint. In *18th ECAI*, pages 475–479, 2008.
- [4] C. Bessiere, G. Katsirelos, N. Narodytska, C.-G. Quimper, and T. Walsh. Propagating conjunctions of AllDifferent constraints. In *24th AAAI*, pages 27–32, 2010.
- [5] M. Biró, M. Hujter, and Z. Tuza. Precoloring extension. I: Interval graphs. *Discrete Math.*, 100:267–279, 1992.
- [6] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75:423–434, 2009.
- [7] F. Bonomo, G. Durán, and J. Marenco. Exploring the complexity boundary between coloring and list-coloring. *Annals of Operations Research*, 169:3–16, 2009.
- [8] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.
- [9] A. A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *43rd FOCS*, page 649, 2002.
- [10] J. Byrka, A. Karrenbauer, and L. Sanita. The interval constrained 3-coloring problem. In *9th LATIN*, pages 591–602, 2010.
- [11] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artif. Intell.*, 38:353–366, 1989.
- [12] R. Diestel. *Graph Theory*. Springer-Verlag, 2000.
- [13] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
- [14] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
- [15] E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32:755–761, 1985.
- [16] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.
- [17] M. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, 1980.
- [18] G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artif. Intell.*, 138:55–86, 2002.
- [19] K. Jansen. The optimum cost chromatic partition problem. In *3rd CIAC*, pages 25–36, 1997.
- [20] P. Jeavons, D. A. Cohen, and M. Gyssens. Closure properties of constraints. *J. ACM*, 44:527–548, 1997.
- [21] C. Jefferson, S. Kadioglu, K. E. Petrie, M. Sellmann, and S. Zivný. Same-relation constraints. In *15th CP*, pages 470–485, 2009.
- [22] C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Deconstructing intractability: A case study for interval constrained coloring. In *20th CPM*, pages 207–220, 2009.
- [23] M. Kutz, K. M. Elbassioni, I. Katriel, and M. Mahajan. Simultaneous matchings: Hardness and approximation. *J. Comput. Syst. Sci.*, 74:884–897, 2008.
- [24] F. Lardeux, E. Monfroy, F. Saubion, B. Crawford, and C. Castro. SAT encoding and CSP reduction for interconnected alldiff constraints. In *8th MICAI*, pages 360–371, 2009.
- [25] J.-L. Lauriere. ALICE: a language and a program for stating and solving combinatorial problems. *Artif. Intell.*, 10:29–127, 1978.
- [26] M. Leconte. A bounds-based reduction scheme for constraints of difference. In *2nd CONSTRAINT*, pages 19–28, 1996.
- [27] D. Marx. Can you beat treewidth? In *48th FOCS*, pages 169–179, 2007.
- [28] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *42nd STOC*, pages 735–744, 2010.
- [29] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
- [30] J.-F. Puget. A fast algorithm for the bound consistency of alldiff constraints. In *13th AAAI*, pages 359–366, 1998.
- [31] J.-C. Régin. A filtering algorithm for constraints of difference in cps. In *9th AAAI*, pages 362–367, 1994.
- [32] F. Roberts. Indifference graphs. In *Proof Techniques in Graph Theory*, pages 139–146, 1969.
- [33] M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci.*, 76:103–114, 2010.
- [34] T. J. Schaefer. The complexity of satisfiability problems. In *10th STOC*, pages 216–226, 1978.
- [35] C. Schulte and P. J. Stuckey. Efficient constraint propagation engines. *ACM Trans. Program. Lang. Syst.*, 31:1–43, 2008.
- [36] K. Stergiou. Heuristics for dynamically adapting propagation. In *18th ECAI*, pages 485–489, 2008.
- [37] W.-J. van Hoeve. The alldifferent constraint: A survey. In *6th CSCLP*, 2001.