Electronic Colloquium on Computational Complexity, Report No. 71 (2011)

The Parameterized Complexity of Local Consistency*

Serge Gaspers and Stefan Szeider
Institute of Information Systems
Vienna University of Technology, Vienna, Austria
gaspers@kr.tuwien.ac.at, stefan@szeider.net

Abstract

We investigate the parameterized complexity of deciding whether a constraint network is k-consistent. We show that, parameterized by k, the problem is complete for the complexity class co-W[2]. As secondary parameters we consider the maximum domain size d and the maximum number ℓ of constraints in which a variable occurs. We show that parameterized by $k + d$, the problem drops down one complexity level and becomes co-W[1]-complete. Parameterized by $k + d + \ell$ the problem drops down one more level and becomes fixed-parameter tractable. We further show that the same complexity classification applies to strong k-consistency, directional k-consistency, and strong directional k-consistency.

Our results establish a super-polynomial separation between input size and time complexity. Thus we strengthen the known lower bounds on time complexity of k-consistency that are based on input size.

1 Introduction

Local consistency is one of the oldest and most fundamental concepts of constraint solving and can be traced back to Montanari's 1974 paper [24]. If a constraint network is locally consistent, then consistent instantiations to a small number of variables can be consistently extended to an additional variable. Hence local consistency avoids certain dead-ends in the search tree, in some cases it even guarantees backtrack-free search [1, 20]. The simplest and most widely used form of local consistency is arc-consistency, introduced by Mackworth [23], and later generalized to k-consistency by Freuder [19]. A constraint network is k-consistent if each consistent assignment to $k-1$ variables can be consistently extended to any additional k-th variable.

Consider a constraint network of *input size* s where the constraints are given as relations. It is easy to see that k-consistency can be checked by brute force in time $O(s^k)$ [10]. Hence, if k is a fixed constant, the check is polynomial. However, the algorithm runs in "nonuniform" polynomial time in the sense that the order of the polynomial depends on k, hence the running time scales poorly in k and becomes impractical already for $k \geq 3$. Also more sophisticated algorithms for k-consistency achieve only a nonuniform polynomial running time [8].

In this paper we investigate the possibility of a uniform polynomial-time algorithm for k-consistency, i.e., an algorithm of running time $O(f(k)s^c)$ where f is an arbitrary function and c is a constant independent of k. We carry out our investigations in the theoretical framework of *parameterized complexity* [15, 17, 25] which allows to distinguish between uniform and nonuniform polynomial time. Problems that can be solved in uniform polynomial time are called *fixed-parameter tractable* (FPT), problems that can be solved in nonuniform polynomial time are further classified within a hierarchy of parameterized complexity classes forming the chain FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq$ W[3] $\subseteq \cdots$, where all inclusions are believed to be strict.

*This research was funded by the ERC (COMPLEX REASON, 239962).

ISSN 1433-8092

Results We pinpoint the exact complexity of k-consistency decision in general and under restrictions on the given constraint network in terms of domain size d and the maximum number ℓ of constraints in which a variable occurs.

We show that deciding k-consistency is co-W[2]-complete for parameter k, co-W[1]-complete for parameter $k+d$, and fixed-parameter tractable for parameter $k+d+\ell$. Hence, subject to complexity theoretic assumptions, k-consistency cannot be decided in uniform polynomial-time in general, but admits a uniform polynomial-time solution if domain size and variable occurrence are bounded. The hardness results imply a super-polynomial separation between input size and running time for k-consistency algorithms.

We further show that all three complexity results also hold for deciding *strong* k-consistency, for deciding *directional* k-consistency, and for deciding *strong directional* k-consistency. A constraint network is strongly k-consistent if it is j-consistent for all $1 \leq j \leq k$. Directional local consistency takes a fixed ordering of the variables into account, the variable to which a local instantiation is extended is ordered higher than the previously instantiated variables [12].

Known Lower Bounds In previous research, lower bounds on the running time of k-consistency algorithms have been obtained [8, 10]. These lower bounds are based on instances of large input size, and the observation that any k-consistency algorithm needs to read the entire input. For instance, to decide whether a given constraint network on n variables is k-consistent one needs to check each constraint of arity $r \leq k$ at least once (the arity of a constraint is the number of variables that occur in the constraint). Since there can be $\sum_{i=1}^{k} \binom{n}{i}$ such constraints, $\Omega(n^k)$ provides a lower bound on the running time of any k-consistency algorithm. Taking the domain size d into account, this lower bound can be improved to $\Omega((dn)^k)$ [10]. However, the constraint networks to which this lower bound applies are of size $s = \Omega((dn)^k)$. Therefore the known lower bounds do not provide a separation between input size and running time.

2 Preliminaries

2.1 Constraint Networks and Local Consistency Problems

A *constraint network* (or *CSP instance*) N is a triple (X, D, C), where X is a finite set of *variables*, D is a finite set of *values*, and C is a finite set of *constraints*. Each constraint $c \in C$ is a pair (S, R), where $S = var(C)$, the *constraint scope*, is a finite sequence of distinct variables from X, and R, the *constraint relation*, is a relation over D whose arity matches the length of S, i.e., $R \subseteq D^r$ where r is the length of S. The size of N is $s = |N| = |X| + |D| + \sum_{(S,R)\in C} |S| \cdot (1 + |R|)$.

Let $N = (X, D, C)$ be a constraint network. A *partial instantiation* of N is a mapping $\alpha : X' \to D$ defined on some subset $var(\alpha) = X' \subseteq X$. We say that α *satisfies* a constraint $c = ((x_1, \ldots, x_r), R) \in C$ if $var(c) \subseteq var(\alpha)$ and $(\alpha(x_1), \ldots, \alpha(x_r)) \in R$. If α satisfies all constraints of N then it is a *solution* of N; in this case, N is satisfiable. We say that α is consistent with a constraint $c \in C$ if either $var(c)$ is not a subset of $var(\alpha)$ or α satisfies c. If α is consistent with all constraints of N we call it consistent. The restriction of a partial assignment α to a set of variables Y is denoted $\alpha|_Y$. It has scope $var(\alpha) \cap Y$ and $\alpha|_Y(x) = \alpha(x)$ for all $x \in var(\alpha|_Y)$.

Let $k > 0$ be an integer. A constraint network $N = (X, D, C)$ is *k-consistent* if for all consistent partial instantiations α of N with $|var(\alpha)| = k-1$ and all variables $x \in X \setminus var(\alpha)$ there is a consistent partial instantiation α' such that $var(\alpha') = var(\alpha) \cup \{x\}$, and $\alpha'|_{var(\alpha)} = \alpha$. In such a case we say that α' *consistently extends* α to x. A constraint network is *strongly k-consistent* if it is j-consistent for all $j = 1, \ldots, k$.

For further background on local consistency we refer to other sources [2, 11].

We consider the following decision problem.

k-Consistency	
Input:	A constraint network $N = (X, D, C)$ and an integer $k > 0$.
Question:	Is N k-consistent?

The problem Strong k-Consistency is defined analogously, asking whether N is strongly k-consistent.

It is easy to see that k-Consistency is co-NP-hard if k is unbounded. Take an arbitrary constraint network $N = (X, D, C)$ and form a new network N' from N by adding a new variable x, and $|X| + 1$ new constraints with empty relations, namely the constraint whose scope contains all variables, and all possible constraints of arity $|X|$ having x in their scope. Let $k = |X| + 1$. Now N' is k-consistent if and only if N is not satisfiable. Since k is large this reduction seems somehow unnatural and breaks down for bounded k. This suggests to "deconstruct" this hardness proof (in the sense of [22]) and to parameterize by k.

The constraint network N is *directionally k-consistent* with respect to a total order $\leq$ on its variables if every consistent partial instantiation α of $k-1$ variables of N can be consistently extended to every variable that is higher in the order $\leq$ than any variable of $var(\alpha)$. The corresponding decision problem is defined as follows.

Directional k-Consistency	
Input:	A constraint network $N = (X, D, C)$, a total order $\leq$ on X, and an integer $k > 0$.
Question:	Is N directionally k-consistent with respect to $\leq$?

A constraint network is *strongly directionally k-consistent* if and only if it is directionally j-consistent for all $j = 1, \ldots, k$. The strong counterpart of the Directional k-Consistency problem is called Strong Directional k-Consistency.

We will consider parameterizations of these four problems by k, by $k + d$, and by $k + d + \ell$, where $d = |D|$ and ℓ denotes the maximum number of constraints in which a variable occurs.

2.2 Parameterized Complexity

We define the basic notions of Parameterized Complexity and refer to other sources [15, 17] for an in-depth treatment. A parameterized problem can be considered as a set of pairs (I, k), the instances, where I is the main part and k is the parameter. The parameter is usually a non-negative integer. A parameterized problem is *fixed-parameter tractable* if there exists an algorithm that solves any instance (I, k) of size n in time $f(k)n^{O(1)}$, where f is a computable function. FPT denotes the class of all fixed-parameter tractable decision problems.

Parameterized complexity offers a completeness theory, similar to the theory of NP-completeness, that allows the accumulation of strong theoretical evidence that some parameterized problems are not fixed-parameter tractable. This theory is based on a hierarchy of complexity classes

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \text{W}[3] \subseteq \cdots.$$

where all inclusions are believed to be strict. Each class W[i] contains all parameterized decision problems that can be reduced to a canonical parameterized satisfiability problem P_i under *parameterized reductions*. These are many-to-one reductions where the parameter for one problem maps into the parameter for the other. More specifically, a parameterized problem L reduces to a parameterized problem L' if there is a mapping R from instances of L to instances of L' such that

1. (I, k) is a Yes-instance of L if and only if $(I', k') = R(I, k)$ is a Yes-instance of L',
2. there is a computable function g such that $k' \leq g(k)$, and
3. there is a computable function f and a constant c such that R can be computed in time $O(f(k) \cdot n^c)$, where n denotes the size of (I, k).

A parameterized problem L is then in W[i], $i \in \mathbb{N}$, if it has a parameterized reduction to the problem of deciding whether a Boolean decision circuit (a decision circuit is a circuit with exactly one output), with AND, OR, and NOT gates, of constant depth such that on each path from an input to the output, all but i gates have a constant number of inputs, parameterized by the number of ones in a satisfying assignment to the inputs of the circuit [15].

A parameterized problem is in co-W[i], $i \in \mathbb{N}$, if its complement is in W[i], where the *complement* of a parameterized problem is the parameterized problem resulting from reversing the YES and NO answers.

If any co-W[i]-complete problem is fixed-parameter tractable, then co-W[i] = FPT = co-FPT = W[i] follows, which causes the Exponential Time Hypothesis to fail [17]. Hence co-W[i]-completeness provides strong theoretical evidence that a problem is not fixed-parameter tractable.

2.3 Tries, Turing Machines, and Gaifman Graphs

Tries A *trie* [9, 18] is a tree for storing strings in which there is one node for every prefix of a string. Let T be a trie that stores a set S of strings on an alphabet Σ. At a given node v of T, corresponding to the prefix $p(v)$, there is an array with one entry for every character c of Σ. If $p(v).c$ is a prefix of a string of S, the entry corresponding to c has a pointer to the node corresponding to the prefix $p(v).c$ (the dot denotes a concatenation). If $p(v).c$ is not a prefix of a string of S, the entry corresponding to c has a null pointer. Thus, a trie uses space $O(|S| \cdot |\Sigma|)$, while inserting or searching a string s can be done in time $O(|s|)$ using the ordinal values for characters as array indices.

Turing Machines A *nondeterministic Turing Machine* (*NTM*) [4, 17] with t tapes is an 8-tuple $M = (Q, \Gamma, \beta, \$, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of *states*,
- the *tape alphabet* Γ is a finite set of symbols,
- $\beta \in \Gamma$ is the *blank symbol*, the only symbol allowed to occur on the tape(s) infinitely often,
- $\$ \in \Gamma$ is a delimiter marking the (left) end of a tape,
- $\Sigma \subseteq \Gamma$ is the set of *input symbols*,
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the set of *final* states,
- $\sigma \subseteq Q \setminus F \times \Gamma^t \times Q \times \Gamma^t \times \{L, N, R\}^t$ is the *transition relation*. A transition $(q, (a_1, \ldots, a_t), q', (a'_1, \ldots, a'_t), (d_1, \ldots, d_t)) \in \sigma$ allows the machine, when it is in state q and the head of each tape T_i is positioned on a cell containing the symbol a_i, to transition in one computation step into the state q', writing the symbol a'_i into the cell on which the head of each tape T_i is positioned, and shifting this head one position to the left if $d_i = L$, one position to the right if $d_i = R$, or not at all if $d_i = N$. On each tape, \$ cannot be overwritten and allows only right transitions, which is formally achieved by imposing that whenever $(q, (a_1, \ldots, a_t), q', (a'_1, \ldots, a'_t), (d_1, \ldots, d_t)) \in \sigma$, then for all $i \in \{1, \ldots, t\}$ we have $a_i = \$$ if and only if $a'_i = \$$, and $a_i = \$$ implies $d_i = R$.

Initially, the first tape contains $\$w\beta\beta\ldots$, where $w \in \Sigma^*$ is the input word, all other tapes contain $\$\beta\beta\ldots$, M is in state q_0, and all heads are positioned on the first cell to the right of the \$ symbol. We speak of a *single-tape* NTM if $t = 1$ and of a *multi-tape* NTM if $t > 1$. M accepts the input word w in k steps if there exists a transition path that takes M with input word w to a final state in k computation steps.

Graphs The *Gaifman graph* $\mathcal{G}(N)$ of a constraint network $N = (X, D, C)$ has the vertex set $V(\mathcal{G}(N)) := X$ and its edge set $E(\mathcal{G}(N))$ contains an edge $\{u, v\}$ if u and v occur together in the scope of a constraint of C. In a graph $G = (V, E)$, the *(open) neighborhood* of a vertex v is the subset of vertices sharing an edge with v and is denoted $\Gamma(v)$, its *closed neighborhood* is $\Gamma[v] := \Gamma(v) \cup \{v\}$, and the *degree* of v is $d(v) := |\Gamma(v)|$. The maximum vertex degree of G is denoted $\Delta(G)$. For a vertex set S, $\Gamma[S] := \bigcup_{v \in S} \Gamma[v]$. S is *independent* in G if no two vertices of S are adjacent in G. S is *dominating* in G if $\Gamma[S] = V$.

3 k-Consistency Parameterized by k

In this section, we consider the most natural parameterization of k-Consistency. Theorem 1 shows that the problem is co-W[2]-hard, parameterized by k, and Theorem 2 shows that it is in co-W[2]. These results are also extended to the strong and directional versions of the problem, resulting in Corollary 1, which says that all four problems are co-W[2]-complete when parameterized by k.

Theorem 1. *Parameterized by k, the following problems are* co-W[2]*-hard: k-Consistency, Strong k-Consistency, Directional k-Consistency, and Strong Directional k-Consistency.*

Proof. We show a parameterized reduction from Independent Dominating Set to the complement of k-Consistency. The Independent Dominating Set problem was shown to be W[2]-hard by Downey and Fellows [13] (see also [7] where W[2]-completeness is established).

Independent Dominating Set	
Input:	A graph $G = (V, E)$ and an integer $k \geq 0$.
Parameter:	k.
Question:	Is there a set $S \subseteq V$ of size k that is independent and dominating in G?

Let $G = (V, E)$ and $k \geq 0$ be an instance of Independent Dominating Set. We construct a constraint network $N = (X, D, C)$ as follows. We take $k + 1$ variables and put $X = \{x_1, \dots, x_{k+1}\}$. For $1 \leq i \leq k + 1$ we put $D(x_i) = V$. The set C contains $\binom{k+1}{2}$ constraints $c_{i,j} = ((x_i, x_j), R_E)$, $1 \leq i < j \leq k + 1$, where $R_E = \{\, (v, u) \in V \times V \mid u \neq v,\ \{u, v\} \notin E \,\}$. This completes the definition of the constraint network N.

Claim 1. *G has an independent dominating set of size k if and only if N is not $(k + 1)$-consistent.*

To show the ($\Rightarrow$)-direction, suppose $S = \{v_1, \dots, v_k\}$ is an independent dominating set of G. Consider the partial instantiation α with $var(\alpha) = \{x_1, \dots, x_k\}$ and $\alpha(x_i) = v_i$, $1 \leq i \leq k$.

First we show that α is consistent. Consider an arbitrary constraint $c_{i,j}$ with $var(c_{i,j}) \subseteq var(\alpha)$. It follows that $1 \leq i < j \leq k$. Since S is an independent set, $\{v_i, v_j\} \notin E$, and so $(\alpha(x_i), \alpha(x_j)) = (v_i, v_j) \in R_E$. Hence α is consistent.

Second, we show that α cannot be consistently extended to x_{k+1}. Let α' be an arbitrarily chosen partial instantiation of N with $var(\alpha') = \{x_1, \dots, x_{k+1}\}$ extending α. Let $v_{k+1} = \alpha'(x_{k+1})$. Since S is a dominating set of G, there must be some $1 \leq i \leq k$ with $\{v_i, v_{k+1}\} \in E$. Consequently $(\alpha'(x_i), \alpha'(x_{k+1})) = (v_i, v_{k+1}) \notin R_E$, hence α' is not consistent with $c_{i,k+1}$. Since α' was chosen arbitrarily, we conclude that α cannot be consistently extended to x_{k+1}. Hence N is not $(k + 1)$-consistent.

It remains to show the ($\Leftarrow$)-direction. Assume that N is not $(k + 1)$-consistent. Hence there is a partial instantiation α on k variables that cannot be consistently extended to a further variable x. Without loss of generality, assume $var(\alpha) = \{x_1, \dots, x_k\}$ and $x = x_{k+1}$. Let $v_i = \alpha(x_i)$, $1 \leq i \leq k$. Since α is consistent, it follows that $S = \{v_1, \dots, v_k\}$ is an independent set of G. Furthermore, $v_i \neq v_j$ for $1 \leq i < j \leq k$, since R_E does not contain any pair of the form (v, v). Hence $|S| = k$. It remains to show that S is dominating. Assume to the contrary that some $v_{k+1} \in V \setminus S$ is not dominated by S, i.e., v_{k+1} is not a neighbor of any vertex in S. This however, implies that the extension α' of α with $\alpha(x_{k+1}) = v_{k+1}$ is consistent, contradicting our assumption. Hence S is indeed an independent dominating set of size k, and Claim 1 is shown true.

Evidently N can be obtained from G in polynomial time. Thus we have established a parameterized reduction from Independent Dominating Set to the complement of k-Consistency. The co-W[2]-hardness of k-Consistency, parameterized by k, now follows from the W[2]-hardness of Independent Dominating Set.

The co-W[2]-hardness of Strong k-Consistency, parameterized by k, is proved analogously by reducing from the variant of Independent Dominating Set which asks for an independent dominating set of size *at most* k. This variant is also W[2]-hard, as shown by Downey et al. [16].

To show that the directional versions of the problem are co-W[2]-hard, parameterized by k, we use the same reductions and additionally specify a total ordering of the vertices. We use the total order by increasing indices of the variables, and observe that the variable to which the partial order α cannot be extended is the last variable in this order in both directions of the proof of Claim 1. Thus, this modification of the reductions shows that DIRECTIONAL k-CONSISTENCY and STRONG DIRECTIONAL k-CONSISTENCY are also co-W[2]-hard parameterized by k. □

The reductions of Theorem 1 actually show somewhat stronger results, namely that the four problems are co-W[2]-hard when parameterized by $k + \ell$. This follows from the observation that the number of variables in the target problems is $k + 1$. From Theorem 2, the co-W[2]-membership of this parameterization will follow. Thus, the problems are co-W[2]-complete when parameterized by $k + \ell$.

For the co-W[2]-membership proof, we build a multi-tape nondeterministic Turing machine that reaches a final state in $f(k)$ steps, for some function f, if and only if N is not k-consistent. As this reduction needs to be a parameterized reduction, we need avoid that the size of the Turing machine (and the time needed to compute it) depends on $O(|X|^k)$ or $O(d^k)$ terms, which would have been very handy to model constraint scopes and constraint relations. We counter this issue via organizing the states of the NTM in tries. There is a first level of tries to determine whether a certain subset of variables is the scope of some constraint. There is a second level of tries to find out whether a certain partial instantiation is allowed by a constraint relation. A second issue that needs particular attention is the size of the transition table. The number of tapes of the NTM is $d + 4$, and we cannot afford a transition for each combination of characters that the head of each tape might be positioned on. We use Cesati's information hiding trick [4] to avoid this issue, which means that the machine does the computations in such a way that in each state, it knows for most tapes (i.e., all, except a constant number of tapes) which characters are in the cell on which the corresponding head is positioned.

Theorem 2. *Parameterized by k, the following problems are in* co-W[2]*: k-CONSISTENCY, STRONG k-CONSISTENCY, DIRECTIONAL k-CONSISTENCY, and STRONG DIRECTIONAL k-CONSISTENCY.*

Proof. Cesati [4] showed that the following parameterized problem is in W[2].

SHORT MULTI-TAPE NTM COMPUTATION	
Input:	A multi-tape NTM M, a word w on the input alphabet of M, and an integer $k > 0$.
Parameter:	k.
Question:	Does M accept w in at most k steps?

We reduce the complement of k-CONSISTENCY to SHORT MULTI-TAPE NTM COMPUTATION. Let $(N = (X, D, C), k)$ be an instance for k-CONSISTENCY. We will construct an instance (M, w, k') which is a YES-instance for SHORT MULTI-TAPE NTM COMPUTATION if and only if (N, k) is a NO-instance for k-CONSISTENCY.

Let us describe how $M = (Q, \Gamma, \beta, \$, \Sigma, q_0, F, \sigma)$ operates. M has $d + 4$ tapes, named $Gx, Gd, Gx_k, S, d_1, \ldots, d_d$, and the input word w is empty. Thus, all the information about N is encoded in the states and transitions of M. The tape alphabet of M is $\Gamma = \{\beta, \$\} \cup X \cup D \cup \{T, F, 1, 0\}$.

In the initialization phase, M writes a 'T' symbol on the tapes $d_1, \ldots, d_d$ and it positions the head of each tape on the first blank symbol of this tape. This can be done in one computation step.

In the guess phase, M nondeterministically guesses $x(1), \ldots, x(k) \in X$ such that $x(i) < x(i + 1)$ for all $i \in \{1, \ldots, k - 2\}$, and it guesses $d(1), \ldots, d(k - 1) \in D$. Here, $\leq$ is an arbitrary order on the variables, and $a < b$ means $a \leq b$ and $a \neq b$. It appends $x(1), \ldots, x(k - 1)$ to the tape Gx, it appends $d(1), \ldots, d(k - 1)$ to the tape Gd, and it appends $x(k)$ to the tape Gx_k. The goal is to make M halt in a final state after a number of steps only depending on k if and only if the partial instantiation α, with $\alpha(x(i)) = d(i), 1 \leq i \leq k - 1$, is consistent, but α cannot be consistently extended to $x(k)$. See Figure 1 for a typical content of the tapes during the execution of M.

The remaining states of M are partitioned into $|X|$ parts, one part for each choice of $x(k)$. M reads $x(k)$ on the tape Gx_k and moves to the initial state in the part corresponding to $x(k)$.

Gx:	\$	$x(1)$	$x(2)$	$x(3)$	$\cdots$	$x(k-1)$
Gd:	\$	$d(1)$	$d(2)$	$d(3)$	$\cdots$	$d(k-1)$
Gx_k:	\$	$x(k)$				
S:	\$	0	0	1	$\cdots$	0
d_1:	\$	T	F	F		
d_2:	\$	T				
d_3:	\$	T	F			
$\cdots$						
d_d:	\$	T	F	F		

Figure 1: A typical content of the tapes during an execution of M (blank symbols are omitted).

On the S tape, M now enumerates all binary 0/1 strings of length $k-1$. The strings in $\{0,1\}^{k-1}$ represent subsets of $\{x(1),\ldots,x(k-1)\}$, i.e., all possible scopes of the constraints that could be violated by the partial instantiation α. For each such binary string, representing a subset X' of $\{x(1),\ldots,x(k-1)\}$, M moves to a state representing X' if there is a constraint with scope X', otherwise it moves to a state calculating the next subset X'. This is achieved by a trie of states; each node of this trie represents a subset X'' of X which is the subset of the first few variables of the scope of some constraint (i.e., X'' represents the prefix of a constraint scope, if we imagine all constraint scopes to be strings of increasing variable names). Thus, the size of this trie does not exceed $O(|C| \cdot |X|)$, and the node corresponding to X' (or the evidence that there is no node corresponding to X') is found in $O(|X'|) = O(k)$ steps. Without loss of generality, we may assume that for each subset of X, there is at most one constraint with that scope; otherwise merge constraints with the same scope. If there is a node representing X', there is a constraint c with scope X'. A trie of states starting at this node represents all tuples that belong to the constraint relation R of c. This trie has size $O(|R| \cdot |X'|)$. Moreover, M can determine in time $O(|X'|)$ whether the tuple t, setting $x(i)$ to $d(i)$ for each i such that $x(i) \in X'$, is in R. If so, it moves to a state representing t, otherwise it moves to a non-accepting state where is loops forever (as the selected partial instantiation α is not consistent). At the state representing t, it appends 'F' to all tapes d_j such that there exists a constraint with scope $X' \cup \{x(k)\}$ and its constraint relation does not contain the tuple setting $x(i)$ to $d(i)$ for each $x(i) \in X'$ and setting $x(k)$ to d_j. Then, it moves to the state computing the next set X'. The machine can only move to a final state if the last symbol on each d_i-tape is 'F', meaning that the calculated partial instantiation $\alpha(x(i)) = d(i), 1 \leq i \leq k-1$ is consistent (otherwise the machine loops forever in a non-accepting state), but cannot be consistently extended to $x(k)$ (otherwise some d_i-tape does not end in 'F'), which certifies that (N,k) is a No-instance for k-Consistency.

The number of states of M is clearly polynomial in $|N| + k$. The transition relation has also polynomial size as we use Cesati's information hiding trick [4], and place the head of the tapes $d_1,\ldots,d_d$ always on the first blank symbol, except for the final check of whether M moves into a final state. If the machine can reach a final state, it can reach one in a number of steps which is a function of k only. This proves the co-W[2]-membership of k-Consistency, parameterized by k.

Checking whether a network is a No-instance for Strong k-Consistency can be done by checking whether it is a No-instance for j-Consistency for some $j \in \{1,\ldots,k\}$. Thus, it is sufficient to build k NTMs as we described, one for each value of $j \in \{1,\ldots,k\}$, nondeterministically guess the integer j for which N is not j-consistent in case N is a No-instance, and move to the initial state of the j^{th} NTM checking whether N is a No-instance for j-Consistency. Thus, Strong k-Consistency parameterized by k is in co-W[2].

For the directional variants of the problem, the order $\leq$ is the one given in the input. It is sufficient to additionally require $x(k)$ to represent a variable that is higher in the order $\leq$ than all variables $x(1),\ldots,x(k-1)$. Thus, our condition that $x(i) < x(i+1)$ for all $i \in \{1,\ldots,k-2\}$ is extended to $i \in \{1,\ldots,k-1\}$. We conclude that the parameterizations of Directional k-Consistency and Strong Directional k-Consistency by k are in co-W[2] as well. □

From Theorems 1 and 2, we obtain the following corollary.

Corollary 1. *Parameterized by k, the following problems are* co-W[2]*-complete:* k-Consistency, Strong k-Consistency, Directional k-Consistency, *and* Strong Directional k-Consistency.

As mentioned before, the corollary also holds for the parameterization by $k + \ell$.

4 k-Consistency Parameterized by $k + d$

In our quest to find parameterizations that make local consistency problems tractable, we augment the parameter by the domain size d. We find that, with this parameterization, the problems become co-W[1]-complete. The co-W[1]-hardness follows from a parameterized reduction from Independent Set.

Theorem 3. *Parameterized by $k + d$, the following problems are hard for* co-W[1]*:* k-Consistency, Strong k-Consistency, Directional k-Consistency, *and* Strong Directional k-Consistency.

Proof. To show that the complement of k-Consistency is W[1]-hard, we reduce from Independent Set, which is well-known to be W[1]-hard [14].

Independent Set	
Input:	A graph $G = (V, E)$ and an integer $k \geq 0$.
Parameter:	k.
Question:	Is there an independent set of size k in G?

Let $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and $k \geq 0$ be an instance of Independent Set. We construct a constraint network $N = (X, D, C)$ as follows.

The set of variables is $X = \{x_1, \dots, x_n, c\}$. The set of values is $D = \{0, \dots, k\}$. The constraint set C contains the constraints

(a) $((x_i, x_j), \{(a, b) : a, b \in \{0, \dots, k\} \text{ and } (a = 0 \text{ or } b = 0)\})$, for all $v_i v_j \in E$, constraining at least one of x_i and x_j to take the value 0 if $v_i v_j \in E$,

(b) $((x_i, c), \{(a, b) : a, b \in \{0, \dots, k\} \text{ and } (a = 0 \text{ or } a \neq b)\})$, for all $i \in \{1, \dots, n\}$, constraining c to be set to a value different from j if any x_i is set to $j > 0$, and

(c) $((c), \{(1), \dots, (k)\})$, restricting the domain of c to $\{1, \dots, k\}$.

This completes the definition of the constraint network N. See Figure 2 for an illustration of N.

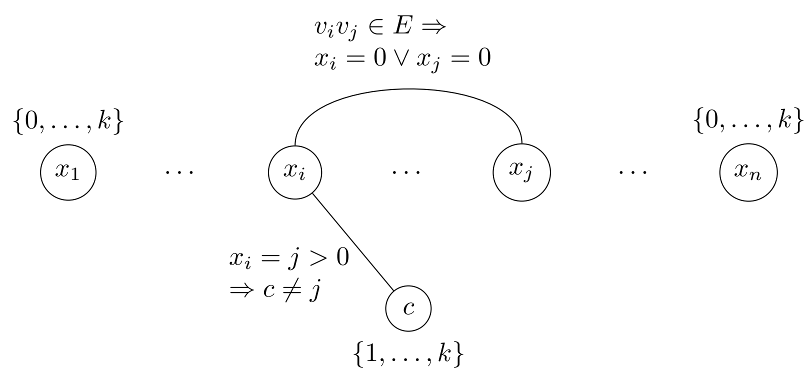

Figure 2: The target constraint network in the parameterized reduction from Independent Set.

Claim 2. *G has an independent set of size k if and only if N is not $(k+1)$-consistent.*

To show the ($\Rightarrow$)-direction, suppose $S = \{v_{s(1)}, \ldots, v_{s(k)}\}$ is an independent set in G. Consider the partial instantiation α such that $\alpha(x_{s(i)}) = i$, $i = 1, \ldots, k$. This partial instantiation is consistent, but cannot be consistently extended to c.

To show the ($\Leftarrow$)-direction, suppose α is a consistent partial instantiation of k variables and x is a variable such that α cannot be consistently extended to x. As the only constraint preventing a variable to be set to 0 is the constraint (c) restricting the domain of c to $\{1, \ldots, k\}$, we have that $x = c$. Now, that c cannot take any of the values in $\{1, \ldots, k\}$ is achieved by the constraints of type (b) by having α bijectively map k variables $x_{s(1)}, \ldots, x_{s(k)}$ to the set $\{1, \ldots, k\}$ without violating any constraint. As two distinct vertices can only be assigned values different from 0 each if they are not adjacent, by the constraints of type (a), we have that $\{x_{s(1)}, \ldots, x_{s(k)}\}$ is an independent set of size k. Hence Claim 2 is shown true.

Evidently N can be obtained from G in polynomial time. Thus we have established a parameterized reduction from Independent Set to the complement of k-Consistency with $d = k + 1$. The co-W[1]-hardness of k-Consistency, parameterized by $k + d$, now follows from the W[1]-hardness of Independent Set.

For the co-W[1]-hardness of Strong k-Consistency, parameterized by $k + d$, just observe that any partial instantiation of fewer than k variables can be extended to any other variable. Thus, G has an independent set of size k if and only if N is not strongly k-consistent, and the co-W[1]-hardness of Strong k-Consistency, parameterized by $k + d$, follows analogously.

For the directional versions of the problem, we use the same reduction and define the ordering in the target problem to be some ordering which has c as its last element. Observing that c is the variable to which the partial instantiation α cannot be extended in both directions of the proof of Claim 2, the co-W[1]-hardness of Directional k-Consistency and Strong Directional k-Consistency, parameterized by $k + d$, follows. □

It remains to show co-W[1]-membership, which easily follows from the parameterized reduction from Theorem 2 (we designed the proof of Theorem 2 in such a way that the same parameterized reduction shows co-W[1]-membership for the parameterization by $k + d$).

Theorem 4. *Parameterized by $k + d$, the following problems are in* co-W[1]*: k-Consistency, Strong k-Consistency, Directional k-Consistency, and Strong Directional k-Consistency.*

Proof. Cesati and Di Ianni [6] showed that the following parameterized problem is in W[1] (see also [3] where W[1]-completeness is established for the single-tape version of the problem).

Short Bounded-tape NTM Computation	
Input:	A t-tape NTM M, a word w on the input alphabet of M, and an integer $k > 0$.
Parameter:	$k + t$.
Question:	Does M accept w in at most k steps?

Now, the proof follows from the proof of Theorem 2, which gives a parameterized reduction from the four problems to Short Multi-tape NTM Computation where the number of tapes is bounded by $d + 4$. □

From Theorems 3 and 4, we obtain the following corollary.

Corollary 2. *Parameterized by $k+d$, the following problems are* co-W[1]*-complete: k-Consistency, Strong k-Consistency, Directional k-Consistency, and Strong Directional k-Consistency.*

5 k-Consistency Parameterized by $k + d + \ell$

We further augment the parameter by ℓ, the maximum number of constraints in which a variable occurs. For this parameterization, we are able to show that the considered problems are fixed-parameter tractable. Bounding both d and ℓ is a reasonable restriction, as it still admits constraint networks whose satisfiability is NP-complete. For instance, determining whether a graph with maximum degree 4 is 3-colorable is an NP-complete problem [21] that can be naturally expressed as a constraint network with $d = 3$ and $\ell = 4$.

For checking whether there is a partial assignment that cannot be extended to a variable x, our FPT algorithm uses the fact that the number of constraints involving x is bounded by a function of the parameter. As constraints with a scope on more than k variables are irrelevant, it follows that the number of variables whose instantiation could prevent x from taking some value can also be bounded by a function of the parameter. For strong k-consistency, these observations are already sufficient to obtain an FPT algorithm as all instantiations of subsets of size at most $k-1$ of the relevant variables can be enumerated. For (non-strong) k-consistency, the algorithm tries to select some independent variables to complete the consistent partial assignment, which must be of size exactly $k-1$. If such a set of independent variables does not exist, the size of the considered constraint network is actually bounded by a function of the parameter and can be solved by a brute-force algorithm.

Theorem 5. *Parameterized by $k+d+\ell$, the following problems are fixed-parameter tractable: k-*Consistency*,* Strong k-Consistency*,* Directional k-Consistency*, and* Strong Directional k-Consistency*.*

Proof. Consider an input instance $N = (X, D, C)$ for k-Consistency. In a first step, discard all constraints c with $|var(c)| > k$, as they cannot influence whether N is k-consistent. The algorithm goes over all $|X|$ possibilities for choosing the vertex x to which a consistent partial instantiation α on $k-1$ variables cannot be extended. If $|X| \leq k \cdot (1 + k \cdot \ell)$, then the number of constraints is at most $|X| \cdot \ell \leq k \cdot (1 + k \cdot \ell) \cdot \ell$ and each constraint has size at most $k \cdot (1 + d^k)$. It follows that

$$|N| \leq k \cdot (1 + k \cdot \ell) + d + (1 + k \cdot \ell) \cdot k^2 \cdot \ell \cdot (1 + d^k).$$

Thus, N is a kernel, i.e., its size is a function of the parameter, and any algorithm solving k-Consistency for N (brute-force search or Cooper's algorithm [8]) has a running time that can be bounded by a function of the parameter only.

Therefore, suppose $|X| > k \cdot (1 + k \cdot \ell)$. Let $G := \mathcal{G}(N)$ be the Gaifman graph of N. The algorithm chooses a set S of $k-1$ variables for the scope of α. To do this, it goes over all $\delta = 0, \ldots, k-1$, where δ represents the number of variables in $S \cap \Gamma(x)$. The number of possibilities for choosing these δ variables is at most $\binom{k \cdot \ell}{\delta}$ as $d(x) \leq k \cdot \ell$. The remaining $k - 1 - \delta$ variables of S need to be chosen from $V \setminus \Gamma[S \cup \{x\}]$. Note that these variables do not influence whether α can be extended to x as they do not occur in a constraint with x. So, it suffices to choose them such that α remains consistent if $\alpha|_{\Gamma(x)}$ was consistent. To do this, the algorithm chooses an independent set of size $k - 1 - \delta$ in $G \setminus \Gamma[S \cup \{x\}]$, which exists and can be obtained greedily due to the lower bound on $|X|$ and because every variable has degree at most $k \cdot \ell$. This terminates the selection of the $k-1$ variables for the scope of α. The algorithm then goes over all d^{k-1} partial instantiations with scope S. For each such partial instantiation α, check in polynomial time whether it is consistent, and if so, whether it can be consistently extended to x. If any such check finds that α is consistent, but cannot be consistently extended to x, answer No, otherwise answer Yes. This part of the algorithm takes time $2^{k \cdot \ell} \cdot d^{k-1} \cdot |N|^{O(1)}$. We conclude that k-Consistency, parameterized by $k + d + \ell$, is fixed-parameter tractable.

The algorithm for the Strong k-Consistency problem is simpler. After having chosen x, there is no need to consider variables that do not occur in a constraint with x. To choose S, it goes over all subsets of $\Gamma(x)$ of size at most $k-1$, and proceeds as described above.

To solve the Directional k-Consistency and Strong Directional k-Consistency problems, after having chosen x, the algorithm deletes all variables from N that occur after x in the ordering $\leq$, and it also removes the constraints whose scope contains at least one of the deleted variables. Then, the algorithm proceeds as above. □

Once a local inconsistency in a constraint network is detected, one can add a new (redundant) constraint to the network that excludes this local inconsistency. More specifically, if we detect that a constraint network $N = (X, D, C)$ is not k-consistent because some partial instantiation α to a set $S = \{x_1, \ldots, x_{k-1}\}$ of variables cannot be extended to some variable x, we add the redundant constraint $((x_1, \ldots, x_{k-1}), D^{k-1} \setminus \{(\alpha(x_1), \ldots, \alpha(x_{k-1}))\})$ to the network. We repeat this process until we end up with a network N^* that is k-consistent. One says that N^* is obtained from N by *enforcing k-consistency* [2]. Similar notions can be defined for strong/directional k-consistency.

It is obvious that the computational task of enforcing k-consistency is at least as hard as deciding k-consistency. Hence, by Theorems 1 and 3, enforcing (strong/directional) k-consistency is co-W[1]-hard when parameterized by $k + d$ and co-W[2]-hard when parameterized by k.

The fixed-parameter tractability result of Theorem 5 does not directly apply to enforcing, since one can construct instances with small d and ℓ that require the addition of a large number of redundant constraints that exceeds any fixed-parameter bound. However, we can obtain fixed-parameter tractability by restricting the enforced network N^*. Let ℓ^* denote the maximum number of constraints in which a variable occurs after k-consistency is enforced. The proof of Theorem 5 shows that enforcing k-consistency is fixed-parameter tractable when parameterized by $k + d + \ell^*$.

6 Conclusion

In recent years numerous computational problems from various areas of computer science have been identified as fixed-parameter tractable or complete for a parameterized complexity class W[i] or co-W[i]. The list includes fundamental problems from combinatorial optimization, logic, and reasoning (see, e.g., Cesati's compendium [5]). Our results place fundamental problems of constraint satisfaction within this complexity hierarchy.

It is perhaps not surprising that the general local consistency problems are fixed-parameter intractable. The drop in complexity from co-W[2] to co-W[1] when we include the domain size as a parameter shows that domain size is of significance for the complexity of local consistency. Somewhat surprising to us is Theorem 5 which shows that under reasonable assumptions there is still hope for fixed-parameter tractability. This result suggests to look for other less restricted cases for which local consistency checking or even enforcing is fixed-parameter tractable. For instance, it would be interesting to see if Theorem 5 still holds if we replace ℓ with the average number of constraints in which a variable occurs.

References

[1] A. Atserias, A. A. Bulatov, and V. Dalmau. On the power of k-consistency. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 279–290. Springer Verlag, 2007.

[2] C. Bessiere. Constraint propagation. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 3. Elsevier, 2006.

[3] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic*, 36(4–5):321–337, 1997.

[4] M. Cesati. The Turing way to parameterized complexity. *J. of Computer and System Sciences*, 67:654–685, 2003.

[5] M. Cesati. Compendium of parameterized problems. http://bravo.ce.uniroma2.it/home/cesati/research/compendium.pdf, Sept. 2006.

[6] M. Cesati and M. D. Ianni. Computation models for parameterized complexity. *Mathematical Logic Quarterly*, 43:179–202, 1997.

[7] Y. Chen and J. Flum. The parameterized complexity of maximality and minimality problems. *Annals of Pure and Applied Logic*, 151(1):22–61, 2008.

[8] M. C. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41(1):89–95, 1989.

[9] R. De La Briandais. File searching using variable length keys. In *Papers presented at the the March 3-5, 1959, Western Joint Computer Conference*, IRE-AIEE-ACM '59 (Western), pages 295–298, New York, NY, USA, 1959. ACM.

[10] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–107, 1992.

[11] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[12] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38, 1987.

[13] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. In *Proceedings of the Twenty-first Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, MB, 1991)*, volume 87, pages 161–178, 1992.

[14] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. II. On completeness for $W[1]$. *Theoretical Computer Science*, 141(1-2):109–131, 1995.

[15] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer Verlag, New York, 1999.

[16] R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation problems. In *Parameterized and Exact Computation, Second International Workshop, IWPEC 2006*, volume 4169 of *Lecture Notes in Computer Science*, pages 121–129. Springer Verlag, 2006.

[17] J. Flum and M. Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

[18] E. Fredkin. Trie memory. *Communications of the ACM*, 3:490–499, 1960.

[19] E. C. Freuder. Synthesizing constraint expressions. *Communications of the ACM*, 21(11):958–966, 1978.

[20] E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. of the ACM*, 32(4):755–761, 1985.

[21] M. R. Garey and D. R. Johnson. *Computers and Intractability*. W. H. Freeman and Company, New York, San Francisco, 1979.

[22] C. Komusiewicz, R. Niedermeier, and J. Uhlmann. Deconstructing intractability - a multivariate complexity analysis of interval constrained coloring. *J. Discrete Algorithms*, 9(1):137–151, 2011.

[23] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[24] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

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

ECCC ISSN 1433-8092

http://eccc.hpi-web.de