Constraint Tightness and Looseness versus Local and Global Consistency

Peter van Beek
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
vanbeek@cs.ualberta.ca

Rina Dechter
Department of Computer and Information Science
University of California, Irvine
Irvine, California, USA 92717
dechter@ics.uci.edu

Published in: *J. ACM,* 44:549–566, 1997.

Abstract

Constraint networks are a simple representation and reasoning framework with diverse applications. In this paper, we identify two new complementary properties on the restrictiveness of the constraints in a network—*constraint tightness* and *constraint looseness*—and we show their usefulness for estimating the level of local consistency needed to ensure global consistency, and for estimating the level of local consistency present in a network. In particular, we present a sufficient condition, based on constraint tightness and the level of local consistency, that guarantees that a solution can be found in a backtrack-free manner. The condition can be useful in applications where a knowledge base will be queried over and over and the preprocessing costs can be amortized over many queries. We also present a sufficient condition for local consistency, based on constraint looseness, that is straightforward and inexpensive to determine. The condition can be used to estimate the level of local consistency of a network. This in turn can be used in deciding whether it would be useful to preprocess the network before a backtracking search, and in deciding which local consistency conditions, if any, still need to be enforced if we want to ensure that a solution can be found in a backtrack-free manner. Two definitions of local consistency are employed in characterizing the conditions: the traditional variable-based notion and a recently introduced definition of local consistency called relational consistency[1].

[1]Some of the results reported in this paper have previously appeared at KR-94 [28] and

1 Introduction

Constraint networks are a simple representation and reasoning framework. A problem is represented as a set of variables, a domain of values for each variable, and a set of constraints between the variables. A central reasoning task is then to find an instantiation of the variables that satisfies the constraints. In spite of the simplicity of the framework, many interesting problems can be formulated as constraint networks, including graph coloring, scene labeling, natural language parsing, and temporal reasoning.

In general, what makes constraint networks hard to solve is that they can contain many local inconsistencies. A local inconsistency is an instantiation of some of the variables that satisfies the relevant constraints that cannot be extended to an additional variable and so cannot be part of any global solution. If we are using a backtracking search to find a solution, such an inconsistency can lead to a dead end in the search. This insight has led to the definition of conditions that characterize the level of local consistency of a network [12, 18, 21] and to the development of algorithms for enforcing local consistency conditions by removing local inconsistencies (e.g., [2, 7, 10, 18, 21]).

Local consistency has proven to be an important concept in the theory and practice of constraint networks for primarily two reasons. First, a common method for finding solutions to a constraint network is to first preprocess the network by enforcing local consistency conditions, and then perform a backtracking search. The preprocessing step can reduce the number of dead ends reached by the backtracking algorithm in the search for a solution. With a similar aim, local consistency techniques can be interleaved with backtracking search. The effectiveness of using local consistency techniques in these two ways has been studied empirically (e.g., [4, 6, 13, 14, 23]). Second, much previous work has identified conditions for when a certain level of local consistency is sufficient to guarantee that a network is globally consistent or that a solution can be found in a backtrack-free manner (e.g., [5, 7, 11, 12, 21, 29]).

In this paper, we identify two new complementary properties on the restrictiveness of the constraints in a network—*constraint tightness* and *constraint looseness*—and we show their usefulness for estimating the level of local consistency needed to ensure global consistency, and for estimating the level of local consistency present in a network. In particular, we present the following results.

We show that, in any constraint network where the constraints have arity r or less and tightness of m or less, if the network is strongly $((m+1)(r-1)+1)$-consistent, then it is globally consistent. Informally, a constraint network is strongly k-consistent if any instantiation of any $k-1$ or fewer variables that satisfies the constraints can be extended consistently to any additional variable. Also informally, given an r-ary constraint and an instantiation of $r-1$ of the variables that participate in the constraint, the parameter m is an upper bound

AAAI-94 [27].

on the number of instantiations of the rth variable that satisfy the constraint. In general, such sufficient conditions, bounding the level of local consistency that guarantees global consistency, are important in applications where constraint networks are used for knowledge base maintenance and there will be many queries against the knowledge base. Here, the cost of preprocessing will be amortized over the many queries. They are also of interest for their explanatory power, as they can be used for characterizing the difficulty of problems formulated as constraint networks.

We also show that, in any constraint network where the domains are of size d or less, and the constraints have looseness of m or greater, the network is strongly $(\lceil d/(d-m) \rceil)$-consistent[2]. Informally, given an r-ary constraint and an instantiation of $r-1$ of the variables that participate in the constraint, the parameter m is a lower bound on the number of instantiations of the rth variable that satisfy the constraint. The bound is straightforward and inexpensive to determine. In contrast, all but low-order local consistency is expensive to verify or enforce as the optimal algorithms to enforce k-consistency are $O(n^k d^k)$, for a network with n variables and domains of size at most d [2, 24].

The condition based on constraint looseness is useful in two ways. First, it can be used in deciding which low-order local consistency techniques will *not* change the network and thus are not useful for processing a given constraint network. For example, we use our results to show that the n-queens problem, a widely used test-bed for comparing backtracking algorithms, has a high level of inherent local consistency. As a consequence, it is generally fruitless to preprocess such a network. Second, it can be used in deciding which local consistency conditions, if any, still need to be enforced if we want to ensure that a solution can be found in a backtrack-free manner.

Two definitions of local consistency are employed in characterizing the conditions: the traditional variable-based notion and a recently introduced definition of local consistency called relational consistency [9, 29].

2 Definitions and Preliminaries

A *constraint network* $\mathcal{R}$ is a set of n variables $\{x_1, \ldots, x_n\}$; a domain D_i of possible values for each variable x_i, $1 \leq i \leq n$; and a set of t *constraint relations* $\{R_{S_1}, \ldots, R_{S_t}\}$, where $S_i \subseteq \{1, \ldots, n\}$, $1 \leq i \leq t$. Each constraint relation R_{S_i}, $1 \leq i \leq t$, is a subset of a Cartesian product of the form $\bigotimes_{j \in S_i} D_j$. We say that R_{S_i} *constrains* the variables $\{x_j \mid j \in S_i\}$. The *arity* of a constraint relation is the number of variables that it constrains. The set $\{S_1, \ldots, S_t\}$ is called the *scheme* of $\mathcal{R}$. We assume that $S_i \neq S_j$, for all $1 \leq i, j \leq t$, $i \neq j$. Because we assume that variables have names, the order of the variables constrained by a relation is not important (see [26, pp. 43–45]). We use subsets of the integers $\{1, \ldots, n\}$ and subsets of the variables $\{x_1, \ldots, x_n\}$ interchangeably.

[2] $\lceil x \rceil$, the ceiling of x, is the smallest integer greater than or equal to x.

Let $Y \subseteq \{1, \ldots, n\}$ be a subset of the variables in a constraint network. An *instantiation* of the variables in Y is an element of $\bigotimes_{j \in Y} D_j$. Given an instantiation $\bar{a}$ of the variables in Y, an *extension* of $\bar{a}$ to a variable x_i, $x_i \notin Y$, is the tuple $(\bar{a}, a_i)$, where a_i is in the domain of x_i.

We now introduce a needed operation on relations adopted from the relational calculus (see [26] for details).

Definition 1 (projection) *Let R_S be a relation, let S be the set of variables constrained by R_S, and let $S' \subseteq S$ be a subset of the variables. The* projection *of R_S onto the set of variables S', denoted $\Pi_{S'}(R_S)$, is the relation which constrains the variables in S' and contains all tuples $u \in \bigotimes_{j \in S'} D_j$ such that there exists an instantiation $\bar{a}$ of the variables in $S - S'$ such that the tuple $(u, \bar{a}) \in R_S$.*

Let $Y \subseteq \{1, \ldots, n\}$ be a subset of the variables in a constraint network. An instantiation $\bar{a}$ of the variables in Y *satisfies* or is *consistent* with a relation R_{S_i}, $S_i \subseteq Y$, if the tuple $\Pi_{S_i}(\{\bar{a}\}) \in R_{S_i}$. An instantiation $\bar{a}$ of the variables in Y is *consistent* with a network $\mathcal{R}$ if and only if for all S_i in the scheme of $\mathcal{R}$ such that $S_i \subseteq Y$, $\bar{a}$ satisfies R_{S_i}. A *solution* to a constraint network is an instantiation of all n of the variables that is consistent with the network.

Definition 2 (m-tight) *A constraint relation R of arity k is called* m-tight *if, for any variable x_i constrained by R and any instantiation $\bar{a}$ of the remaining $k-1$ variables constrained by R, either there are at most m extensions of $\bar{a}$ to x_i that satisfy R, or there are exactly $|D_i|$ such extensions.*

Definition 3 (m-loose) *A constraint relation R of arity k is called* m-loose *if, for any variable x_i constrained by R and any instantiation $\bar{a}$ of the remaining $k-1$ variables constrained by R, there are at least m extensions of $\bar{a}$ to x_i that satisfy R.*

The tightness and looseness properties are complementary properties on the constraints in a network. Tightness is an upper bound on the number of extensions and looseness is a lower bound. With the exception of the universal relation, a constraint relation that is m_1-loose and m_2-tight has $m_1 \leq m_2$ (the case where there are exactly $|D_i|$ extensions to variable x_i is handled specially in the definition of tightness). Every constraint relation is 0-loose and a constraint relation is d-loose if and only if it is the universal relation, where the domains of the variables constrained by the relation are of size at most d. Every constraint relation is $(d-1)$-tight and a constraint relation is 0-tight if and only if it is either the null relation or the universal relation.

In what follows, we are interested in the least upper bound and greatest lower bound on the number of extensions of the relations of a constraint network. It is easy to check that:

Observation 1 *Given a constraint network with t constraint relations each with arity at most r and each with at most e tuples in the relation, determining the*

least m such that all t of the relations are m-tight requires $O(tre)$ time. The same bound applies to determining the greatest m such that the relations are m-loose.

Example 1. We illustrate the definitions using the following network $\mathcal{R}$ with variables $\{x_1, x_2, x_3, x_4\}$, domains $D_i = \{\text{a,b,c}\}$, $1 \leq i \leq 4$, and relations,

$$\begin{aligned} R_{S_1} &= \{(\text{a,a,a}),\ (\text{a,a,c}),\ (\text{a,b,c}),\ (\text{a,c,b}),\ (\text{b,a,c}),\\ &\qquad (\text{b,b,b}),\ (\text{b,c,a}),\ (\text{c,a,b}),\ (\text{c,b,a}),\ (\text{c,c,c})\},\\ R_{S_2} &= \{(\text{a,b}),\ (\text{b,a}),\ (\text{b,c}),\ (\text{c,a}),\ (\text{c,c})\},\\ R_{S_3} &= \{(\text{a,b}),\ (\text{a,c}),\ (\text{b,b}),\ (\text{c,a}),\ (\text{c,b})\}, \end{aligned}$$

where $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_2, x_4\}$, and $S_3 = \{x_3, x_4\}$. The projection of R_{S_1} onto $\{x_1, x_3\}$ is given by,

$$\Pi_{\{x_1,x_3\}}(R_{S_1}) = \{(\text{a,a}),\ (\text{a,c}),\ (\text{a,b}),\ (\text{b,c}),\ (\text{b,b}),\ (\text{b,a}),\ (\text{c,b}),\ (\text{c,a}),\ (\text{c,c})\}.$$

The instantiation $\bar{a} = (\text{a,c,b})$ of the variables in $Y = \{x_2, x_3, x_4\}$ is consistent with $\mathcal{R}$ since $\Pi_{S_2}(\{\bar{a}\}) = (\text{a,b})$ and $(\text{a,b}) \in R_{S_2}$, and $\Pi_{S_3}(\{\bar{a}\}) = (\text{c,b})$ and $(\text{c,b}) \in R_{S_3}$. The instantiation $\bar{a} = (\text{c,a,b})$ of the variables in Y is not consistent with $\mathcal{R}$ since $\Pi_{S_2}(\{\bar{a}\}) = (\text{c,b})$, and $(\text{c,b}) \notin R_{S_2}$.

Let $\bar{a} = (\text{a,a,a,b})$ be an instantiation of all of the variables $\{x_1, x_2, x_3, x_4\}$. The tuple $\bar{a}$ is consistent with $\mathcal{R}$ and is therefore a solution of the network. The set of all solutions of the network is given by,

$$\begin{aligned} \{&(\text{a,a,a,b}),\ (\text{a,a,c,b}),\ (\text{a,b,c,a}),\ (\text{b,a,c,b}),\\ &(\text{b,c,a,c}),\ (\text{c,a,b,b}),\ (\text{c,b,a,c}),\ (\text{c,c,c,a})\}. \end{aligned}$$

It can be verified that all of the constraints are 2-tight and 1-loose. As a partial verification of the binary constraint R_{S_3}, consider the extensions to variable x_3 given instantiations of the variable x_4. For the instantiation $\bar{a} = (\text{a})$ of x_4 there is 1 extension to x_3; for $\bar{a} = (\text{b})$ there are 3 extensions (but $|D_3| = 3$ so the definition of 2-tightness is still satisfied); and for $\bar{a} = (\text{c})$ there is 1 extension.

Local consistency has proven to be an important concept in the theory and practice of constraint networks. We now review previous definitions of local consistency, which we characterize as variable-based and relation-based.

2.1 Variable-based local consistency

Mackworth [18] defines three properties of networks that characterize local consistency: *node*, *arc*, and *path consistency*. Freuder [10] generalizes this to k-consistency, which can be defined as follows:

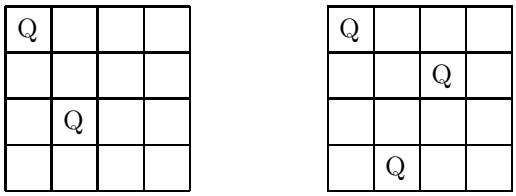
Definition 4 (k-consistency, global consistency) *A constraint network $\mathcal{R}$ is* k-consistent *if and only if given*

1. *any $k-1$ distinct variables, and*

2. *any instantiation $\bar{a}$ of the $k-1$ variables that is consistent with $\mathcal{R}$,*

there exists an extension of $\bar{a}$ to any kth variable such that the k-tuple is consistent with $\mathcal{R}$. A network is strongly k-consistent *if and only if it is j-consistent for all $j \leq k$. A strongly n-consistent network is called* globally consistent*, where n is the number of variables in the network.*

Node, arc, and path consistency correspond to one-, two-, and three-consistency, respectively. Globally consistent networks have the property that a solution can be found without backtracking [11].



(a) (b)

Figure 1: (a) not 3-consistent; (b) not 4-consistent

Example 2. We illustrate the definition of k-consistency using the well-known n-queens problem. The problem is to find all ways to place n-queens on an $n \times n$ chess board, one queen per column, so that each pair of queens does not attack each other. One possible constraint network formulation of the problem is as follows: there is a variable for each column of the chess board, $x_1, \ldots, x_n$; the domains of the variables are the possible row positions, $D_i = \{1, \ldots, n\}$, $1 \leq i \leq n$; and the binary constraints are that two queens should not attack each other. Consider the constraint network for the 4-queens problem. It can be seen that the network is 2-consistent since, given that we have placed a single queen on the board, we can always place a second queen such that the queens do not attack each other. However, the network is not 3-consistent. For example, given the consistent placement of two queens shown in Figure 1a, there is no way to place a queen in the third column that is consistent with the previously placed queens. Similarly the network is not 4-consistent (see Figure 1b).

2.2 Relation-based local consistency

In [29], we extended the notions of arc and path consistency to non-binary relations. The new local consistency conditions were called relational arc- and path-consistency. In [9], we generalized relational arc- and path-consistency to *relational m-consistency*. In the definition of *relational m-consistency*, the relations rather than the variables are the primitive entities. As we shall see in

subsequent sections, this allows expressing the relationships between the restrictiveness of the constraints and local consistency in a way that avoids an explicit reference to the arity of the constraints. The definition below is slightly weaker than that given in [9].

Definition 5 (relational m-consistency) *A constraint network $\mathcal{R}$ is* relationally m-consistent *if and only if given*

1. *any m distinct relations $R_{S_1}, \ldots, R_{S_m}$, and*
2. *any $x \in \bigcap_{i=1}^{m} S_i$*
3. *any instantiation $\bar{a}$ of the variables in $(\bigcup_{i=1}^{m} S_i) - \{x\}$ that is consistent with $\mathcal{R}$,*

there exists an extension of $\bar{a}$ to x such that the extension is consistent with the m relations. A network is strongly relationally m-consistent if it is relationally j-consistent for every $j \leq m$.

Example 3. Consider the constraint network with variables $\{x_1, x_2, x_3, x_4, x_5\}$; domains $D_i = \{\text{a, b, c}\}$, $i = 1, \ldots, 5$; and relations,

$R_{S_1} = \{$(a,a,a,a), (b,a,a,a), (a,b,a,a), (a,a,b,a), (a,a,a,b)$\}$,
$R_{S_2} = \{$(b,a,b), (c,b,c), (b,a,c)$\}$.

where $S_1 = \{x_2, x_3, x_4, x_5\}$ and $S_2 = \{x_1, x_2, x_5\}$. The constraints are not relationally 1-consistent. For example, the instantiation (a,b,b) of the variables in $Y = \{x_2, x_3, x_4\}$ is consistent with the network (trivially so, since $S_1 \not\subseteq Y$ and $S_2 \not\subseteq Y$), but it does not have an extension to x_5 that satisfies R_{S_1}. Similarly, the constraints are not relationally 2-consistent. For example, the instantiation (c,b,a,a) of the variables in $\{x_1, x_2, x_3, x_4\}$ is consistent with the network (again, trivially so), but it does not have an extension to x_5 that satisfies both R_{S_1} and R_{S_2}. If we add the constraints $R_{\{x_2\}} = R_{\{x_3\}} = R_{\{x_4\}} = \{\text{a}\}$ and $R_{\{x_1\}} = R_{\{x_5\}} = \{\text{b}\}$, the set of solutions of the network does not change, and it can be verified that the network is both relationally 1- and 2-consistent.

When the constraints are all binary, relational m-consistency is identical to variable-based $(m + 1)$-consistency; otherwise the conditions are different. While enforcing *variable-based* m-consistency can be done in polynomial time, it is unlikely that relational m-consistency can be achieved tractably since even for $m = 2$ it solves the NP-complete problem of propositional satisfiability (see Example 6). A more direct argument suggesting an increase in time and space complexity is the fact that an algorithm may need to record relations of arbitrary arity. As with variable-based local-consistency, we can improve the efficiency of enforcing relational consistency by enforcing it only along a certain direction or linear ordering of the variables. Algorithms for enforcing relational consistency and directional relational consistency are given in [9, 28].

3 Constraint Tightness vs Global Consistency

In this section, we present relationships between the tightness of the constraints and the level of local consistency sufficient to ensure that a network is globally consistent.

Much work has been done on identifying relationships between properties of constraint networks and the level of local consistency sufficient to ensure global consistency. This work falls into two classes: identifying topological properties of the underlying graph of the network (e.g., [7, 8, 11, 12]) and identifying properties of the constraints (e.g., [3, 15, 16, 21, 29]). Dechter [5] identifies the following relationship between the size of the domains of the variables, the arity of the constraints, and the level of local consistency sufficient to ensure the network is globally consistent.

Theorem 1 (Dechter [5]) *If a constraint network with domains that are of size at most d and relations that are of arity at most r is strongly $(d(r-1)+1)$-consistent, then it is globally consistent.*

For some networks, Dechter's theorem is tight in that the level of local consistency specified by the theorem is really required (graph coloring problems formulated as constraint networks are an example). For other networks, Dechter's theorem overestimates. Our results should be viewed as an improvement on Dechter's theorem. By taking into account the tightness of the constraints, our results always specify a level of strong consistency that is less than or equal to the level of strong consistency required by Dechter's theorem.

The following lemma is needed in the proof of the main result.

Lemma 1 *Let $R_{S_1}, \ldots, R_{S_l}$ be l relations that constrain a variable x, let d be the size of the domain of variable x, and let $\bar{a}$ be an instantiation of all of the variables except for x that are constrained by the l relations (i.e., $\bar{a}$ is an instantiation of the variables in $(S_1 \cup \cdots \cup S_l) - x$). If*

1. *each relation is m-tight, for some $1 \leq m < d$, and*
2. *for every subset of $m+1$ or fewer relations from $\{R_{S_1}, \ldots, R_{S_l}\}$, there exists at least one extension of $\bar{a}$ to x that satisfies each of the relations in the subset,*

then there exists at least one extension of $\bar{a}$ to x that satisfies all l relations.

Proof. Let $a_1, a_2, \ldots, a_d$ be the d elements in the domain of x. We say that a relation allows an element a_i if the extension $(\bar{a}, a_i)$ of $\bar{a}$ to x satisfies the relation. Assume to the contrary that an extension of $\bar{a}$ to x that satisfies all of the l relations does not exist. Then, for each element a_i in the domain of x there must exist at least one relation that does not allow a_i. Let c_i denote one of the relations that does not allow a_i. By construction, the set $c = \{c_1, c_2, \ldots, c_d\}$

is a set of relations for which there does not exist an extension of $\bar{a}$ to x that satisfies each of the relations in the set (every candidate a_i is ruled out since c_i does not allow a_i). For every possible value of m, this leads to a contradiction.

Case 1 ($m = d - 1$): The contradiction is immediate as $c = \{c_1, c_2, \ldots, c_d\}$ is a set of relations of size $m + 1$ for which there does not exist an extension to x that satisfies every relation in the set. This contradicts condition (2).

Case 2 ($m = d - 2$): The nominal size of the set $c = \{c_1, c_2, \ldots, c_d\}$ is $m + 2$. We claim, however, that there is a repetition in c and that the true size of the set is $m+1$. Assume to the contrary that $c_i \neq c_j$ for $i \neq j$. Recall c_i is a relation that does not allow a_i, $i = 1, \ldots, d$ and consider $\{c_1, c_2, \ldots, c_{d-1}\}$. This is a set of $m + 1$ relations so by condition (2) there must exist an a_i that every relation in the set allows. The only possibility is a_d. Now consider $\{c_1, \ldots, c_{d-2}, c_d\}$. Again, this is a set of $m + 1$ relations so there must exist an a_i that every relation in the set allows. This time the only possibility is a_{d-1}. Continuing in this manner, we can show that relation c_1 must allow $a_d, a_{d-1}, \ldots, a_2$; that is, c_1 must allow exactly $m+1$ extensions. This contradicts condition (1). Therefore, it must be the case that $c_i = c_j$ for some $i \neq j$. Thus, the set c is of size $m + 1$ and this contradicts condition (2).

Case 3 ($m = d - 3$), ..., **Case d-1** ($m = 1$): The remaining cases are similar. In each case we argue that (i) there are repetitions in the set $c = \{c_1, c_2, \ldots, c_d\}$, (ii) the true size of the set c is $m + 1$, and (iii) a contradiction is derived by appealing to condition (2).

Thus, there exists at least one extension of $\bar{a}$ to x that satisfies all of the relations. □

We first state the result using variable-based local consistency and then state the result using relation-based local consistency.

Theorem 2 *If a constraint network with relations that are m-tight and of arity at most r is strongly $((m+1)(r-1)+1)$-consistent, then it is globally consistent.*

Proof. Let $k = (m + 1)(r - 1) + 1$. We show that any network with relations that are m-tight and of arity at most r that is strongly k-consistent is $(k + i)$-consistent for any $i \geq 1$.

Without loss of generality, let $X = \{x_1, \ldots, x_{k+i-1}\}$ be a set of $k + i - 1$ variables, let $\bar{a}$ be an instantiation of the variables in X that is consistent with the constraint network, and let x_{k+i} be an additional variable. Using Lemma 1, we show that there exists an extension of $\bar{a}$ to x_{k+i} that is consistent with the constraint network. Let $R_{S_1}, \ldots, R_{S_l}$ be l relations which are all and only the relations which constrain only x_{k+i} and a subset of variables from X. To be consistent with the constraint network, the extension of $\bar{a}$ to x_{k+i} must satisfy each of the l relations. Now, condition (1) of Lemma 1 is satisfied since each of the l relations is m-tight. It remains to show that condition (2) is satisfied. By

definition, the requirement of strong $((m+1)(r-1)+1)$-consistency ensures that any instantiation of any $(m+1)(r-1)$ or fewer variables that is consistent with the network, has an extension to x_{k+i} such that the extension is also consistent with the network. Note, however, that since each of the l relations is of arity at most r and constrains x_{k+i}, each relation can constrain at most $r-1$ variables that are not constrained by any of the other relations. Therefore, the requirement of strong $((m+1)(r-1)+1)$-consistency also ensures that for any subset of $m+1$ or fewer *relations* from $\{R_{S_1}, \ldots, R_{S_l}\}$, there exists an extension of $\bar{a}$ to x_{k+i} that satisfies each of the relations in the subset. Thus, condition (2) of Lemma 1 is satisfied. Therefore, from Lemma 1 it follows that there is an extension of $\bar{a}$ to x_{k+i} that is consistent with the constraint network. □

Theorem 2 always specifies a level of strong consistency that is less than or equal to the level of strong consistency required by Dechter's theorem (Theorem 1). The level of required consistency is equal only when $m = d-1$ and is less when $m < d-1$. As well, the theorem can sometimes be usefully applied if $d \geq n-1$, whereas Dechter's theorem cannot.

As the following example illustrates, both r, the arity of the constraints, and m can change if the level of consistency required by the theorem is not present and must be enforced. The parameter r can only increase; m can decrease, as shown below, but also increase. The parameter m will increase if all of the following hold: (i) there previously was no constraint between a set of variables, (ii) enforcing a certain level of consistency results in a new constraint being recorded between those variables and, (iii) the new constraint has a larger m value than the previous constraints.

Example 4. Nadel [22] introduces a variant of the n-queens problem called confused n-queens. The problem is to find all ways to place n-queens on an $n \times n$ chess board, one queen per column, so that each pair of queens *does* attack each other. One possible constraint network formulation of the problem is as follows: there is a variable for each column of the chess board, $x_1, \ldots, x_n$; the domains of the variables are the possible row positions, $D_i = \{1, \ldots, n\}$, $1 \leq i \leq n$; and the binary constraints are that two queens should attack each other. The constraint relation between two variables x_i and x_j, $1 \leq i < j \leq n$, is given by,

$$R_{\{i,j\}} = \{(a,b) \mid (a = b) \vee (|a-b| = |i-j|)\}, \quad a, b = 1, \ldots, n.$$

The problem is worth considering, as Nadel [22] uses confused n-queens in an empirical comparison of backtracking algorithms for solving constraint networks. Thus it is important to analyze the difficulty of the problems to set the empirical results in context. As well, the problem is interesting in that it provides an example where Theorem 2 can be applied but Dechter's theorem can not (since $d \geq n-1$). Independently of n, the constraint relations are all 3-tight. Hence, the theorem guarantees that if the network for the confused n-queens problem is strongly 5-consistent, the network is globally consistent.

First, suppose that n is even and we attempt to either verify or achieve this level of strong consistency by applying successively stronger local consistency algorithms. Kondrak [17] has shown that the following analysis holds for all n, n even.

1. Applying an arc consistency algorithm results in no changes as the network is already arc consistent.

2. Applying a path consistency algorithm does tighten the constraints between the variables. Once the network is made path consistent, the constraint relations are all 2-tight. Now the theorem guarantees that if the network is strongly 4-consistent, it is globally consistent.

3. Applying a 4-consistency algorithm results in no changes as the network is already 4-consistent. Thus, the network is strongly 4-consistent and therefore also globally consistent.

Second, suppose that n is odd. This time, after applying path consistency, the networks are still 3-tight and it can be verified that the networks are not 4-consistent. Enforcing 4-consistency requires 3-ary constraints. Adding the necessary 3-ary constraints does not change the value of m; the networks are still 3-tight. Hence, by Theorem 2, if the networks are strongly 9-consistent, the networks are globally consistent. Kondrak [17] has shown that recording 3-ary constraints is sufficient to guarantee that the networks are strongly 9-consistent for all n, n odd. Hence, independently of n, the networks are globally consistent once strong 4-consistency is enforced.

Recall that Nadel [22] uses confused n-queens problems to empirically compare backtracking algorithms for finding all solutions to constraint networks. Nadel states that these problems provide a "non-trivial test-bed" [22, p.190]. We believe the above analysis indicates that these problems are quite easy and that any empirical results on these problems should be interpreted in this light. Easy problems potentially make even naive algorithms for solving constraint networks look promising. To avoid this potential pitfall, backtracking algorithms should be tested on problems that range from easy to hard. In general, hard problems are those that require a high level of local consistency to ensure global consistency. Note also that these problems are trivially satisfiable.

Example 5. The graph k-colorability problem can be viewed as a problem on constraint networks: there is a variable for each node in the graph, the domains of the variables are the k possible colors, and the binary constraints are that two adjacent nodes must be assigned different colors. Graph k-colorability provides examples of networks where both Theorems 1 and 2 give the same bound on the sufficient level of local consistency since the constraints are $(k-1)$-tight.

We now show how the concept of relation-based local consistency can be used to alternatively describe Theorem 2.

Theorem 3 *If a constraint network with relations that are m-tight is strongly relationally $(m+1)$-consistent, then it is globally consistent.*

Proof. We show that any network with relations that are m-tight that is strongly relationally $(m+1)$-consistent is k-consistent for any $k \geq 1$ and is therefore globally consistent.

Without loss of generality, let $X = \{x_1, \ldots, x_{k-1}\}$ be a set of $k-1$ variables, let $\bar{a}$ be an instantiation of the variables in X that is consistent with the constraint network, and let x_k be an additional variable. Using Lemma 1, we show that there exists an extension of $\bar{a}$ to x_k that is consistent with the constraint network. Let $R_{S_1}, \ldots, R_{S_l}$ be l relations which are all and only the relations which constrain only x_k and a subset of variables from X. To be consistent with the constraint network, the extension of $\bar{a}$ to x_k must satisfy each of the l relations. Now, condition (1) of Lemma 1 is satisfied since each of the l relations is m-tight. Further, condition (2) of Lemma 1 is satisfied since, by definition, the requirement of strong relational $(m+1)$-consistency ensures that for any subset of $m+1$ or fewer relations, there exists an extension of $\bar{a}$ to x_k that satisfies each of the relations in the subset. Therefore, from Lemma 1 it follows that there is an extension of $\bar{a}$ to x_k that is consistent with the constraint network. □

As an immediate corollary of Theorem 3, if we know that the result of applying an algorithm for enforcing strong relational $(m+1)$-consistency will be that all of the relations will be m-tight, we can guarantee *a priori* that the algorithm will return an equivalent, globally consistent network.

Example 6. Consider networks where the domains of the variables are of size two. For example, the satisfiability of propositional CNFs provide an example of networks with domains of size two. Relations which constrain variables with domains of size two are 1-tight and any additional relations that are added to the network as a result of enforcing strong relational 2-consistency will also be 1-tight. Thus, the consistency of such networks can be decided by an algorithm that enforces strong relational 2-consistency. A different derivation of the same result is already given by [5, 29].

A backtracking algorithm constructs and extends partial solutions by instantiating the variables in some linear order. Global consistency implies that for any ordering of the variables the solutions to the constraint network can be constructed in a backtrack-free manner; that is, a backtracking search will not encounter any dead-ends in the search. Dechter and Pearl [7] observe that it is often sufficient to be backtrack-free along a particular ordering of the variables and that local consistency can be enforced with respect to that ordering only. Frequently, if the property of interest (in our case tightness and looseness) is satisfied along that ordering we can conclude global consistency restricted to that ordering as well. Enforcing relational consistency with respect to an ordering of the variables can be done by a general elimination algorithm called

DIRECTIONAL-RELATIONAL-CONSISTENCY, presented in [9]. Such an algorithm has the potential of being more effective in practice and in the worst-case as it requires weaker conditions. Directional versions of the tightness and looseness properties and of the results presented in this paper are easily formulated using the ideas presented in [7, 9].

The results of this section can be used as follows. Mackworth [19] shows that constraint networks can be viewed as a restricted knowledge representation and reasoning framework. In this context, solutions of the constraint network correspond to models of the knowledge base. Our results which bound the level of local consistency needed to ensure global consistency, can be useful in applications where constraint networks are used as a knowledge base and there will be many queries against the knowledge base. Preprocessing the constraint network so that it is globally consistent means that queries can be answered in a backtrack-free manner.

An equivalent globally consistent representation of a constraint network is highly desirable since it compiles the answers to many queries and it can be shown that there do exist constraint networks and queries against the network for which there will be an exponential speed-up in the worst case. As an example, consider a constraint network with no solutions. The equivalent globally consistent network would contain only null relations and an algorithm answering a query against this constraint network would quickly return "yes." Of course, of more interest are examples where the knowledge base is consistent. Queries which involve determining if a value for a variable is feasible—can occur in a model of the network—can be answered from the globally consistent representation by looking only at the domain of the variable. Queries which involve determining if the values for a pair of variables is feasible—can both occur in a single model of the network—can be answered by looking only at the binary relations which constrain the two variables. It is clear that a general algorithm to answer a query against the original network, such as backtracking search, can take an exponential amount of time to answer the above queries. In general, a globally consistent representation of a network will be useful whenever it is more compact than the set of all solutions to the network. With the globally consistent representation we can answer any query on a subset of the variables $Y \subseteq \{1, \ldots, n\}$ by restricting our attention to the smaller network which consists of only the variables in Y and only the relations which constrain the variables in Y. The global consistency property ensures that a solution for all of the variables can also be created in a backtrack-free manner. However, how our results will work in practice is an interesting empirical question which remains open.

The results of this section are also interesting for their explanatory power, as they can be used for characterizing the difficulty of problems formulated as constraint networks (see the discussion at the end of the next section).

4 Constraint Looseness vs Local Consistency

In this section, we present a sufficient condition, based on the looseness of the constraints and on the size of the domains of the variables, that gives a lower bound on the inherent level of local consistency of a constraint network.

It is known that some classes of constraint networks already possess a certain level of local consistency and therefore algorithms that enforce this level of local consistency will have no effect on these networks. For example, Nadel [22] observes that an arc consistency algorithm never changes a constraint network formulation of the n-queens problem, for $n > 3$. Dechter [5] observes that constraint networks that arise from the graph k-coloring problem are inherently strongly k-consistent. Our results characterize what it is about the structure of the constraints in these networks that makes these statements true.

The following lemma is needed in the proof of the main result.

Lemma 2 *Let $R_{S_1}, \ldots, R_{S_l}$ be l relations that constrain a variable x, let d be the size of the domain of variable x, and let $\bar{a}$ be an instantiation of all of the variables except for x that are constrained by the l relations (i.e., $\bar{a}$ is an instantiation of the variables in $(S_1 \cup \cdots \cup S_l) - x$). If*

1. *each relation is m-loose, for some $0 \leq m < d$, and*
2. *$l \leq \left\lceil \frac{d}{d-m} \right\rceil - 1$*

there exists at least one extension of $\bar{a}$ to x that satisfies all l relations.

Proof. Let $a_1, a_2, \ldots, a_d$ be the d elements in the domain of x. We say that a relation allows an element a_i if the extension $(\bar{a}, a_i)$ of $\bar{a}$ to x satisfies the relation. Now, the key to the proof is that, because each of the l relations is m-loose, at least m elements from the domain of x are allowed by each relation. Thus, each relation does not allow at most $d - m$ elements, and together the l relations do not allow at most $l(d-m)$ elements from the domain of x. Thus, if

$$l(d - m) < d,$$

it cannot be the case that every element in the domain of x is not allowed by some relation. Thus, if

$$l \leq \left\lceil \frac{d}{d-m} \right\rceil - 1,$$

there exists at least one extension of $\bar{a}$ to x that satisfies all l relations. □

We first state the result using variable-based local consistency and then state the result using relation-based local consistency. Let binomial(k, r) be the binomial coefficients, the number of possible choices of r different elements from a collection of k objects. If $k < r$, then binomial$(k, r) = 0$.

Theorem 4 *A constraint network with domains that are of size at most d and relations that are m-loose and of arity at least r, $r \geq 2$, is strongly k-consistent, where k is the minimum value such that the following inequality holds,*

$$\text{binomial}(k-1, r-1) \geq \left\lceil \frac{d}{d-m} \right\rceil - 1.$$

Proof. Without loss of generality, let $X = \{x_1, \ldots, x_{k-1}\}$ be a set of $k-1$ variables, let $\bar{a}$ be an instantiation of the variables in X that is consistent with the constraint network, and let x_k be an additional variable. To show that the network is k-consistent, we must show that there exists an extension of $\bar{a}$ to x_k that is consistent with the constraint network. Let $R_{S_1}, \ldots, R_{S_l}$ be l relations which are all and only the relations which constrain only x_k and a subset of variables from X. To be consistent with the constraint network, the extension of $\bar{a}$ to x_k must satisfy each of the l relations. From Lemma 2, such an extension exists if $l \leq \lceil d/(d-m) \rceil - 1$.

Now, the level of strong k-consistency is the minimum number of distinct variables that can be constrained by the l relations. In other words, k is the minimum number of variables that can occur in $S_1 \cup \cdots \cup S_l$. We know that each of the relations constrains the variable x_k. Thus, $k = c + 1$, where c is the minimum number of variables in $(S_1 - \{x\}) \cup \cdots \cup (S_l - \{x\})$. The minimum value of c occurs when all of the relations have arity r and thus each $(S_i - \{x\})$, $1 \leq i \leq l$, is a set of $r-1$ variables. Further, we know that each of the l relations constrains a different subset of variables; i.e., if $i \neq j$, then $S_i \neq S_j$ for all $1 \leq i, j \leq l$. The binomial coefficients $\text{binomial}(c, r-1)$ tell us the number of distinct subsets of cardinality $r-1$ which are contained in a set of size c. Thus, $\text{binomial}(c, r-1)$ tells us the minimum number of variables c that are needed in order to specify the remaining $r-1$ variables in each of the l relations subject to the condition that each relation must constrain a different subset of variables. □

Constraint networks with relations that are all binary are an important special case of Theorem 4.

Corollary 1 *A constraint network with domains that are of size at most d and relations that are binary and m-loose is strongly $\left(\left\lceil \frac{d}{d-m} \right\rceil\right)$-consistent.*

Proof. All constraint relations are of arity $r = 2$ and $\text{binomial}(k-1, r-1) = k-1$ when $r = 2$. Hence, the minimum value of k such the inequality in Theorem 4 holds is when $k = \lceil d/(d-m) \rceil$. □

Theorem 4 always specifies a level of local consistency that is less than or equal to the actual level of inherent local consistency of a constraint network. That is, the theorem provides a lower bound. However, given only the looseness of the constraints and the size of the domains, Theorem 4 gives as strong an

estimation of the inherent level of local consistency as possible as examples can be given for all $m < d$ where the theorem is exact. Graph coloring problems provide an example where the theorem is exact for $m = d-1$, whereas n-queens problems provide an example where the theorem underestimates the true level of local consistency.

Example 7. Consider again the well-known n-queens problem discussed in Example 2. The problem is of historical interest but also of theoretical interest due to its importance as a test problem in empirical evaluations of backtracking algorithms and heuristic repair schemes for finding solutions to constraint networks (e.g., [13, 14, 20, 22]). For n-queens networks, each of the domains is of size n and each of the constraints is binary and $(n-3)$-loose. Hence, Theorem 4 predicts that n-queens networks are inherently strongly $(\lceil n/3 \rceil)$-consistent. Thus, an n-queens constraint network is inherently arc-consistent for $n \geq 4$, inherently path consistent for $n \geq 7$, and so on, and we can predict where it is fruitless to apply a low-order consistency algorithm in an attempt to simplify the network (see Table 1). The actual level of inherent consistency is $\lfloor n/2 \rfloor$ for $n \geq 7$. Thus, for the n-queens problem, the theorem underestimates the true level of local consistency.

Table 1: Predicted ($\lceil n/3 \rceil$) and actual ($\lfloor n/2 \rfloor$, for $n \geq 7$) level of strong local consistency for n-queens networks

n	4	5	6	7	8	9	10	11	12	13	14	15
pred.	2	2	2	3	3	3	4	4	4	5	5	5
actual	2	2	2	3	4	4	5	5	6	6	7	7

Example 8. Graph k-colorability provides an example where Theorem 4 is exact in its estimation of the inherent level of local consistency (see Example 5 for the constraint network formulation of graph coloring). As Dechter [5] states, graph coloring networks are inherently strongly k-consistent but are not guaranteed to be strongly $(k+1)$-consistent. Each of the domains is of size k and each of the constraints is binary and $(k-1)$-loose. Hence, Theorem 4 predicts that graph k-colorability networks are inherently strongly k-consistent.

Example 9. Consider a formula in 3-CNF which can be viewed as a constraint network where each variable has the domain {true, false} and each clause corresponds to a constraint defined by its models. The domains are of size two and all constraints are of arity 3 and are 1-loose. The minimum value of k such that the inequality in Theorem 4 holds is when $k = 3$. Hence, the networks are strongly 3-consistent.

We now show how the concept of relation-based local consistency can be used to alternatively describe Theorem 4.

Theorem 5 *A constraint network with domains that are of size at most d and relations that are m-loose is strongly relationally $\left(\left\lceil \frac{d}{d-m} \right\rceil - 1\right)$-consistent.*

Proof. Follows immediately from Lemma 2. □

The results of this section can be used in two ways. First, they can be used to estimate whether it would be useful to preprocess a constraint network using a local consistency algorithm, before performing a backtracking search (see, for example, [6] for an empirical study of the effectiveness of such preprocessing). Second, they can be used in conjunction with previous work which has identified conditions for when a certain level of local consistency is sufficient to ensure a solution can be found in a backtrack-free manner (see, for example, the brief review of previous work at the start of Section 3 together with the new results presented there). Sometimes the level of inherent strong k-consistency guaranteed by Theorem 4 is sufficient, in conjunction with these previously derived conditions, to guarantee that the network is globally consistent and therefore a solution can be found in a backtrack-free manner without preprocessing. Otherwise, the estimate provided by the theorem gives a starting point for applying local consistency algorithms.

The results of this section are also interesting for their explanatory power. We conclude this section with some discussion on what Theorem 2 and Theorem 4 contribute to our intuitions about hard classes of problems (in the spirit of, for example, [1, 30]). Hard constraint networks are instances which give rise to search spaces with many dead ends. The hardest networks are those where many dead ends occur deep in the search tree. Dead ends, of course, correspond to partial solutions that cannot be extended to full solutions. Networks where the constraints are m_1-loose and m_2-tight for values m_1 and m_2, $m_1 \leq m_2$, that are close to d, the size of the domains of the variables, are good candidates to be hard problems. The reasons are two-fold. First, networks that have high looseness values have a high level of inherent strong consistency and strong k-consistency means that all partial solutions are of at least size k. Second, networks that have high tightness values require a high level of preprocessing to be backtrack-free.

Computational experiments we performed on random problems with binary constraints provide evidence that networks with constraints with high looseness values can be hard. Random problems were generated with $n = 50$, $d = 5, \ldots 10$, and $p, q = 1, \ldots, 100$, where $p/100$ is the probability that there is a binary constraint between two variables, and $q/100$ is the probability that a pair in the Cartesian product of the domains is in the constraint. The time to find one solution was measured. In the experiments we discovered that, given that the number of variables and the domain size were fixed, the hardest problems were found when the constraints were as loose as possible without degenerating into the trivial constraint where all tuples are allowed. In other words, we found that the hardest region of loose constraints is harder than the hardest

region of tight constraints. That networks with loose constraints would turn out to be the hardest of these random problems is somewhat counter-intuitive, as individually the constraints are easy to satisfy. These experimental results run counter to Tsang's [25, p.50] intuition that a single solution of a loosely constrained problem "can easily be found by simple backtracking, hence such problems are easy," and that tightly constrained problems are "harder compared with loose problems." As well, these hard loosely-constrained problems are not amenable to preprocessing by low-order local consistency algorithms, since, as Theorem 4 states, they possess a high level of inherent local consistency. This runs counter to Williams and Hogg's [30, p.476] speculation that preprocessing will have the most dramatic effect in the region where the problems are the hardest.

5 Conclusions

We identified two new complementary properties on the restrictiveness of the constraints in a network: *constraint tightness* and *constraint looseness*. Constraint tightness was used, in conjunction with the level of local consistency, in a sufficient condition that guarantees that a solution to a network can be found in a backtrack-free manner. The condition can be useful in applications where a knowledge base will be queried over and over and the preprocessing costs can be amortized over many queries. Constraint looseness was used in a sufficient condition for local consistency. The condition is inexpensive to determine and can be used to estimate the level of strong local consistency of a network. This in turn can be used in deciding whether it would be useful to preprocess the network before a backtracking search, and in deciding which local consistency conditions, if any, still need to be enforced if we want to ensure that a solution can be found in a backtrack-free manner.

We also showed how constraint tightness and constraint looseness are of interest for their explanatory power, as they can be used for characterizing the difficulty of problems formulated as constraint networks and for explaining why some problems that are "easy" locally, are difficult globally. We showed that when the constraints have low tightness values, networks may require less preprocessing in order to guarantee that a solution can be found in a backtrack-free manner and that when the constraints have high looseness values, networks may require much more search effort in order to find a solution. As an example, the confused n-queens problem, which has constraints with low tightness values, was shown to be easy to solve as it is backtrack-free after enforcing only low-order local consistency conditions. As another example, many instances of crossword puzzles are also relatively easy, as the constraints on the words that fit each slot in the puzzle have low tightness values (since not many words have the same length and differ only in the last letter of the word). On the other hand, graph coloring and scheduling problems involving resource constraints can be quite

hard, as the constraints are inequality constraints and thus have high looseness values.

Acknowledgements

The authors wish to thank Peter Ladkin and an anonymous referee for their careful reading of a previous version of the paper and their helpful comments.

References

[1] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 331–337, Sydney, Australia, 1991.

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

[3] M. C. Cooper, D. A. Cohen, and P. G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.

[4] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

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

[6] R. Dechter and I. Meiri. Experimental evaluation of preprocessing techniques in constraint satisfaction problems. *Artificial Intelligence*, 68:211–242, 1994.

[7] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.

[8] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[9] R. Dechter and P. van Beek. Local and global relational consistency. *Theoretical Computer Science*, 173:283–308, 1997.

[10] E. C. Freuder. Synthesizing constraint expressions. *Comm. ACM*, 21:958–966, 1978.

[11] E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29:24–32, 1982.

[12] E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32:755–761, 1985.

[13] J. Gaschnig. Experimental case studies of backtrack vs. waltz-type vs. new algorithms for satisficing assignment problems. In *Proceedings of the Second Canadian Conference on Artificial Intelligence*, pages 268–277, Toronto, Ont., 1978.

[14] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.

[15] P. Jeavons, D. Cohen, and M. Gyssens. A test for tractability. In *Proceedings of the Second International Conference on Principles and Practice of Constraint Programming*, pages 267–281, Cambridge, Mass., 1996. Available as: Springer Lecture Notes in Computer Science 1118.

[16] L. M. Kirousis. Fast parallel constraint satisfaction. *Artificial Intelligence*, 64:147–160, 1993.

[17] G. Kondrak, 1993. Personal Communication.

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

[19] A. K. Mackworth. The logic of constraint satisfaction. *Artificial Intelligence*, 58:3–20, 1992.

[20] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 17–24, Boston, Mass., 1990.

[21] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inform. Sci.*, 7:95–132, 1974.

[22] B. A. Nadel. Constraint satisfaction algorithms. *Computational Intelligence*, 5:188–224, 1989.

[23] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.

[24] R. Seidel. On the complexity of achieving k-consistency. Department of Computer Science Technical Report 83-4, University of British Columbia, 1983. Cited in: A. K. Mackworth 1987.

[25] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[26] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Vol. 1.* Computer Science Press, 1988.

[27] P. van Beek. On the inherent level of local consistency in constraint networks. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 368–373, Seattle, Wash., 1994.

[28] P. van Beek and R. Dechter. Constraint tightness versus global consistency. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 572–582, Bonn, Germany, 1994.

[29] P. van Beek and R. Dechter. On the minimality and global consistency of row-convex constraint networks. *J. ACM*, 42(3):543–561, 1995.

[30] C. P. Williams and T. Hogg. Using deep structure to locate hard problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 472–477, San Jose, Calif., 1992.