On Tightness of Constraints[⋆]

Yuanlin Zhang

Cork Constraint Computation Center
University College Cork
Cork, Republic of Ireland
y.zhang@4c.ucc.ie

Abstract. The *tightness* of a constraint refers to how restricted the constraint is. The existing work shows that there exists a relationship between tightness and global consistency of a constraint network. In this paper, we conduct a comprehensive study on this relationship. Under the concept of k-consistency (k is number), we strengthen the existing results by establishing that only some of the tightest, *not all*, binary constraints are used to predict a number k such that strong k-consistency ensures global consistency of an arbitrary constraint network which may include non-binary constraints. More importantly, we have shown a lower bound of the number of the tightest constraints we *have to* consider in predicting the number k. To make better use of the tightness of constraints, we propose a new type of consistency: *dually adaptive consistency.* Under this concept, only the tightest *directionally relevant* constraint on each variable (and thus in total $n - 1$ such constraints where n is the number of variables) will be used to predict the level of "consistency" ensuring global consistency of a network.

1 Introduction

Informally, the *tightness* of a binary constraint is the *maximum number of compatible values* allowed for each value of the constrained variables. For example, let $x, y \in 1..10$ be two variables and consider a constraint $x = y$. For any value of x, the constraint allows at most 1 compatible value for y . The constraint is also said 1-tight. An interesting discovery is that there is a close relationship between the tightness and the global consistency of a constraint network. (When we say a network is *globally consistent*, we mean it is satisfiable.) For example, if all the constraints in a binary network is 1-tight, path consistency (i.e., strongly 3-consistency) is sufficient to determine the global consistency of the network. If *not all* constraints are 1-tight, the existing method will use the least tight constraint to determine the level of consistency sufficient for global consistency. *This level is higher (and thus more expensive) if the constraints are less tight.* The main objective of this paper is to determine the level of consistency by using less and tighter constraints. For example, by our results, if a constraint network

⋆ This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075.

with n variables has n 1-tight constraints on "correct" variables, a variation of path consistency is still able to ensure its global consistency.

One of the first interesting pieces of work related to the tightness of constraints is by Dechter [2]. She observes that the size of the largest domain in a constraint network can be used to predict a number k such that strong k-consistency [6] is sufficient to guarantee the global consistency of the network. Later, van Beek and Dechter [10] propose the the concept of tightness. They observe that the least tight constraint in a network are also able to be used to predict a k. The k derived by tightness is obviously lower than that derived by the domain size because the tightness of a constraint is at most the size of the domains involved in the constraint.

Recently, we proposed a concept of *weakly m-tight constraint networks* in [11]. Using this concept, one can use the tightness of all binary constraints (while ignoring the tightness of all other non-binary constraints) to predict the k-consistency sufficient for global consistency.

In this paper, we first study the potential of weak m-tightness in reducing the number of constraints needed to predict a number k such that strong k-consistency on the network ensures global consistency. Some property of weakly m-tight networks is presented and we show that we can use a certain number (*but not all*) of the tightest binary constraints to make an predication of k. Unexpectedly and importantly, we also find that there exists a lower bound on the number of constraints we have to use in a prediction under the concept of weak m-tightness of a network.

The weak m-tightness of a network grows from the concept of k-consistency which requires the consistent extensibility of a valid instantiation of any $k-1$ variables to any new variable. A weakly m-tight network assures that there is an m-tight constraint involved when extending any instantiation to a new variable. k-consistency is so strong a property that it might restrict the role of tightness in determining the global consistency of a network.

We then study the impact of tightness on global consistency under the concept of directional k-consistency. In directional k-consistency, it is only necessary to assure the consistent extensibility of a valid instantiation to a new variable which comes *after* the instantiated variables in terms of some variable ordering. This approach further reduces the number of constraints required for a prediction of a local level of consistency ensuring global consistency. However, the reduction is still not very substantial.

We continue the exploration by considering *adaptive consistency* – an elegant and natural extension of directional consistency – which is devised when considering the topological structure of a constraint network. It needs an ordering of variables and guarantees that for any variable, all constraints involving it and its predecessors are "consistent" on it. It is observed that to make the "relevant" constraints consistent on a variable, the computation effort is dependent on the tightest constraint (which may or may not be binary) on the variable. This observation leads to the concept of *dually adaptive consistency* which assures global consistency but requires "less" consistency inside a network by making full use of its topological structure and the tightness of the constraints.

With dually adaptive consistency, we are finally able to say that we only need the $n-1$, where n is the number of variables, tightest constraints (in the sense of taking the tightest "relevant" constraint on each variable) to determine the computational cost for achieving global consistency.

2 Preliminaries

In this section, we review the basic concepts and notations used in this paper.

Constraint Networks A *constraint network* consists of a set of variables $V = \{x_1, x_2, \cdots, x_n\}$ with a domain D_i for each variable $x_i \in V$, and a set of constraints $C = \{c_{S_1}, c_{S_2}, \cdots, c_{S_e}\}$ where S_i is a subset of V for all $i : 1..n$, and each constraint $c_S \in C$ is a relation defined on the domains of variables in S. Given a constraint c_S, S is also called the *scope* of c_S. The *arity* of c_S is the number of variables in S. If $|S|$ is two, c_S is *binary* and denoted by c_{ij} on variables x_i and x_j. Throughout the paper, n denotes the number of variables in a network.

See [8, 9] for more information on constraint networks.

Consistent Instantiation and Image An instantiation of a set of variables $Y = \{x_1, \cdots, x_j\}$ is denoted by $\bar{a} = (a_1, \cdots, a_j)$ where $a_i \in D_i$ for $i \in 1..j$; and it is *consistent* if it satisfies all constraints involving only variables in Y. For $b \in D_i$, $(\bar{a}, b)$ denotes an instantiation of $Y \cup \{x_i\}$. Given a constraint c_S and an instantiation $\bar{a}$ of $X - \{x_i\}$ $(S \subseteq X \subseteq V)$ for any $x_i \in S$, $u \in D_i$ is a support (with respect to c_S) of $\bar{a}$ if $(\bar{a}, u)$ satisfies c_S; and the *image* of $\bar{a}$ on D_i under c_S is the set of all its supports in D_i.

***m*-tightness and Proper *m*-tightness** [10, 11] Given a number m, a constraint c_S is *m-tight on x_i* if and only if the image of any instantiation of $S-\{x_i\}$ is of size at most m or the size of D_i. If a constraint is m-tight on x, its tightness on x is m. A constraint is m-tight if it is m-tight on each of its variables.

A constraint c_S is *properly m-tight on x* if and only if the image of any instantiation of $S - \{x\}$ is of size up to m. A constraint is *properly m-tight* if and only if it is properly m-tight on every variable in its scope.

Example The constraint in Fig. 1 is properly 3-tight because b_1 has image $\{a_2, b_2, c_2\}$ which is maximum among all images of all values here. However in deciding the m-tightness of the constraint, we ignore the image of b_1 because it is equal to the domain of y. The image $\{b_1, c_1\}$ (or $\{a_1, b_1\}$) of a_2 (or c_2) is maximum and thus the constraint is 2-tight. $\Box$

3 Tightness under *k*-consistency

We first discuss k-consistency, weakly m-tight constraint networks and the existing results, and then present a detailed analysis of weakly m-tight networks.

Relevant Constraints A *relevant constraint* on a variable x with respect to a set of variables Y is one whose scope consists of only x and variables from

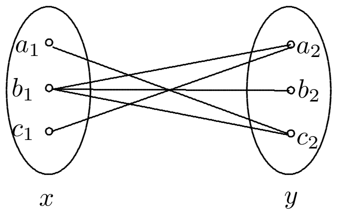

Fig. 1. A constraint between x and y. The domain of x is $\{a_1, b_1, c_1\}$ and the domain of y $\{a_2, b_2, c_2\}$. An allowed tuple by the constraint is drawn as an edge.

Y. In other words, it involves x, but does not involve any variable outside Y. $R_Y(x)$ is used to denote the set of relevant constraints on x wrt Y. When Y is clear from the context, $R(x)$, rather than $R_Y(x)$, will be used.

Example Consider the network in Fig. 4. At this moment, we do not assume any ordering on the variables and just take it as a normal network. The relevant constraints on x_2 with respect to $\{x_1, x_3\}$ are c_{12} and c_{32}. Without a reference set of variables, the relevant constraints of x_2 refer to those on it with respect to all variables: c_{12}, c_{32}, and c_{42}. $\Box$

(Strong) k-consistency [6] A constraint network is *k-consistent* if and only if for any consistent instantiation $\bar{a}$ of any distinct $k-1$ variables, and for any new variable x_i, there exists $u \in D_i$ such that $(\bar{a}, u)$ is a consistent instantiation of the k variables. A network is *strongly* k-consistent if and only if it is j-consistent for all $j \leq k$. *A strongly n-consistent network where n is the number of constraints is globally consistent.* We use local consistency in this section to denote (strongly) k-consistency with some $k < n$.

Weakly m-tight Constraint Networks A constraint network is *weakly m-tight* at level k iff for every set of variables $Y = \{x_1, \cdots, x_k\}$ and a new variable x, there exists an m-tight relevant constraint on x wrt Y.

This definition is simpler than the one given in [11] where every set Y of size k or greater than k is considered. The Proposition 1 below shows that the two definitions are equivalent.

Remark The definition needs the assumption that given a network, there is a universal constraint among any set of variables on which there is no explicit constraint. A universal constraint on a set of variables allows any instantiation of the variables. So, in this section, we need to keep in mind that there is a constraint among any set of variables.

For a weakly tight network, there exists the following relationship between local and global consistency.

Theorem 1. [11] *If for some m, a constraint network with constraints of arity at most r is strongly $((m+1)(r-1)+1)$-consistent and weakly m-tight at level $((m+1)(r-1)+1)$, it is strongly n-consistent.*

As we can see from the definition, the weak m-tightness of a network does not require us to consider the tightness of all constraints. What interests us here is how many constraints are needed to make a network weakly m-tight. In our earlier work [11], we were not able to answer this question except to show a sufficient condition that if all binary constraints (possibly including universal constraints) of a network are m-tight, then it is weakly m-tight. In this section we give some characterization of the weakly m-tight constraint networks.

We first find that there is a strong relationship among different levels of weak tightness in a network.

Proposition 1. *If a constraint network is weakly m-tight at level k for some m, it is weakly m-tight at any level $j > k$.*

Proof. For any $j > k$, we prove that the network is weakly tight at level j. That is, for any set of variables $Y = \{x_1, \cdots, x_j\}(k \leq j < n)$ and a new variable x, we show that there exists an m-tight relevant constraint on x with respect to Y. Since the network is weakly tight for $k < j$, there exists an m-tight relevant constraint on x with respect to a subset of Y. This constraint is still relevant on x with respect to Y, and thus the one we look for. □

In the following two results, we show more sufficient conditions for a constraint network to be weakly m-tight.

Theorem 2. *Given a constraint network (V, D, C), if for every $x \in V$, there are at least $n-2$ binary m-tight constraints on it for some m, then the network is weakly m-tight at level* 3.

Proof. For any two variables $\{x, y\}$, and a third variable z, the relevant constraints on z with respect to $\{x, y\}$ are c_{xz} and c_{yz}. We know that the number of relevant binary constraints on z with respect to V is $n-1$. That $n-2$ of them are m-tight means either c_{xz} or c_{yz} must be m-tight. □

In fact, for the weakness at a higher level, we need fewer constraints to be m-tight as shown by the following result.

Theorem 3. *A constraint network (V, D, C) is weakly m-tight at level k if for every $x \in V$, there are at least $n - k + 1$ m-tight binary constraints on it for some m and k.*

Proof. For any set Y of $k-1$ variables, and a new variable z, we show that there is an m-tight relevant constraint on z with respect to Y. Otherwise, all the $k-1$ binary constraints on z are not m-tight. Since the total number of relevant constraints on z is $n-1$, the number of m-tight binary constraints on z is at most $(n-1)-(k-1)$, which contradicts that z is involved in $n-k+1$ m-tight binary constraints. □

This result shows that for a network of arbitrary constraints to be weakly tight at level k, it could need as few as $n(n-k+1)/2$ m-tight constraints, in contrast to the result in [11] that all binary constraints are required to be m-tight.

An immediate question is what is the minimum number of m-tight constraints required for a network to be weakly tight? The following result answers this question on weak tightness at level 3.

Theorem 4. *For a constraint network to be weakly m-tight at level 3, it needs at least*

$$n(n-1)/2 - 2\lfloor n/3 \rfloor \quad \textit{if} \quad n = 0, 1 \pmod 3$$

or otherwise

$$(n-2)(3n-1)/6$$

m-tight binary or ternary constraints for some m.

Proof. Given a network, its weak m-tightness at level 3 depends on the tightness of only binary and ternary constraints. Among all weakly m-tight (at level 3) constraint networks with n variables, let the network (V, D, C) have a minimum set M of m-tight binary or ternary constraints. Let B be the set of binary constraints and T the set of ternary constraints in M.

In the following exposition, a constraint is denoted by its scope. For example, we use $\{u, v, w\}$ and $\{u, v\}$ to denote ternary constraint $c_{\{u,v,w\}}$ and binary constraint c_{uv} respectively.

We first prove that keeping the total number of constraints in M unchanged and the underlying network weakly m-tight, we can change T and B, if necessary, such that for any constraint $\{u, v, w\}$ in T, none of the binary constraints $\{u, v\}$, $\{v, w\}$, and $\{u, w\}$ is m-tight.

1) At most one of $\{u, v\}$, $\{v, w\}$, and $\{u, w\}$ is m-tight. Otherwise, at least two of them are m-tight, which means $\{u, v, w\}$ can be non-m-tight, contradicting the minimum assumption of M.

2) Assume $\{u, v\}$ is m-tight. Since $\{u, v, w\}$ is m-tight, there should be a reason for $\{u, v\}$ to be m-tight. The only reason is that there exists another variable z such that one of $\{u, z\}$ and $\{v, z\}$ is not m-tight and $\{u, v, z\}$ is not m-tight, too. See Fig. 2. Let us say $\{u, z\}$ is m-tight, then $\{v, z\}$ has to be non-m-tight. There must be an m-tight constraint $\{z, v, w\}$ because $\{v, z\}$ and $\{v, w\}$ are not m-tight. Now we make the following transformation of T and B. Remove from T constraints $\{u, v, w\}$ and $\{z, v, w\}$, and add to B constraints $\{z, v\}$ and $\{v, w\}$, which means we replace the m-tight ternary constraints by m-tight binary constraints. This transformation preserves the number of m-tight constraints.

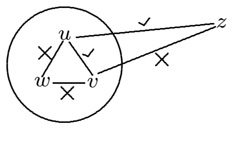

Fig. 2. The circle represents the m-tight ternary constraint $\{u, v, w\}$. An edge between two variables indicates a binary constraint. A tick besides an edge means it is m-tight while a cross means it is not *m-tight.*

Next, we show that keeping the number of constraints in M unchanged and the underlying network weakly m-tight, we can change T and B, if necessary, such that any two ternary constraints do not share any variables.

Case 1: two constraints $\{u, v, w\}$ and $\{u, v, z\}$ in T share two variables $\{u, v\}$. See Fig. 3(a). For $\{w, u\}$ and $\{u, z\}$ are not m-tight, $\{w, u, z\}$ has to be m-tight. For $\{w, v\}$ and $\{v, z\}$ are not m-tight, $\{w, v, z\}$ has to be m-tight. Again we remove the four ternary constraints from T and add to B the four binary constraints $\{w, u\}$, $\{u, z\}$, $\{z, v\}$ and $\{v, w\}$. This transformation preserves the weak m-tightness of the network.

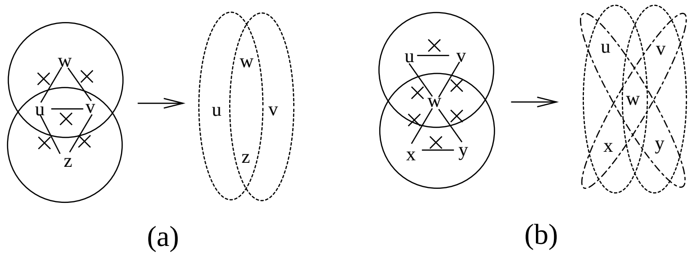

Fig. 3. A dotted ellipse together with the three variables inside it represents a ternary constraint. (a) Left: Two ternary constraints share two variables $\{u, v\}$. Right: The ternary constraints have to be m-tight. (b) Left: Two ternary constraints share one variable w. Right: The ternary constraints have to be m-tight.

Case 2: two constraints $\{u, v, w\}$, and $\{w, x, y\}$ share one variable w. Since $\{u, w\}$ and $\{w, x\}$ are not m-tight, $\{u, w, x\}$ has to be m-tight. Since $\{v, w\}$ and $\{w, y\}$ are not m-tight, $\{v, w, y\}$ has to be m-tight. Similarly $\{u, w, y\}$ and $\{v, w, x\}$ have to be m-tight. Now obviously if we make the four binary constraints $\{u, w\}$, $\{w, x\}$, $\{v, w\}$, and $\{w, y\}$ m-tight while relaxing the six ternary constraints to be non-m-tight, the new network is still weakly m-tight, but has fewer m-tight constraints. This contradicts the minimality of M. Hence, case 2 is not possible.

Therefore, the scopes of the ternary constraints in T are disjoint, and the binary constraint between any two variables of a ternary constraint in T is not m-tight.

Assume there are k constraints in T. Since it is difficult to count B, we count the maximum number of non-m-tight binary constraints. We have $3k$ non-m-tight binary constraints because of T. We should not have any non-m-tight binary constraints between a variable in T and a variable outside T. Let V' be the variables outside T. We have $n - 3k$ variables outside T. The other non-m-tight constraints fall only between variables in V'. Since there is no two non-m-tight constraints on a single variable in V', there are at most $(n - 3k)/2$ non-m-tight constraints if $n - 3k$ is even, and at most $(n - 3k - 1)/2$ otherwise. So the number of m-tight constraints in B and T would be

$$\begin{aligned} &n(n-1)/2 + k - 3k - \lfloor (n-3k)/2 \rfloor \\ &= n(n-1)/2 - 2k - \lfloor (n-3k)/2 \rfloor. \end{aligned}$$

We know that this should be minimized, and thus k should be maximized. If n is a multiple of 3, the number of m-tight constraints is $n(n-1)/2 - 2n/3$; if n is 1 more than a multiple of 3, the number is $n(n-1)/2 - 2(n-1)/3$; otherwise the number is $(n-1)(3n-1)/6$. □

All these results will apply to weakly *properly* m-tight constraint networks.

Weakly Properly m-tight Constraint Networks [11] A constraint network is *weakly properly m-tight* at level k iff for every set of variables $Y = \{x_1, \cdots, x_l\}$ and a new variable x, there exists a properly m-tight relevant constraint on x wrt Y.

The idea behind Proposition 1 is also applicable to this definition. For completeness we list the results on weak proper m-tightness below.

Corollary 1. *If a constraint network is weakly properly tight at level k, it is weakly properly tight at any level $j > k$.*

Corollary 2. *Given a constraint network (V, D, C), if for every $x \in V$, there are at least $n-2$ binary properly m-tight constraints on it, then the network is weakly properly m-tight at level 3.*

Corollary 3. *A constraint network (V, D, C) is weakly properly m-tight at level k if for every $x \in V$, there are at least $n-k+1$ properly m-tight binary constraints on it.*

Corollary 4. *For a constraint network to be weakly properly m-tight at level 3, it needs at least*

$$n(n-1)/2 - 2\lfloor n/3 \rfloor \quad \textit{if} \quad n = 0, 1 \pmod 3$$

or otherwise

$$(n-2)(3n-1)/6$$

properly m-tight binary or ternary constraints.

From the discussion above, under the concept of k-consistency we can not reduce the number of m-tight constraints required in a network by much to predict the k-consistency ensuring global consistency.

4 Tightness under directional consistency

We know that strong n-consistency is stronger than we need to obtain the global consistency of a network in that it requires any partially consistent instantiation extensible to a solution. In fact, in many problems, even for those which need all solutions, we can instantiate the variables one by one along a special ordering. In order to find a solution without backtracking (i.e., efficiently), it is sufficient to ensure that a valid instantiation of a set of variables is extensible to a variable after them. This is the idea behind directional consistency proposed by Dechter and Pearl [5]. In this section, we study tightness under directional consistency.

Directional k-consistency and Directionally Relevant Constraints A constraint network is *directionally k-consistent* with respect to a variable ordering if and only if every consistent instantiation of every $k-1$ variables can be extended to any new variable after them. A network is *strongly* directionally k-consistent if it is directionally j-consistent for all $j \leq k$. It is easy to see *a strongly directionally n-consistent network is globally consistent.* A relevant constraint on x with respect to a set of variables Y, is *directionally relevant* if it involves x and only variables before x. For example, for the network shown in Fig. 4, c_{12} is the only directionally relevant constraint on variable x_2. The other two relevant constraints, c_{32} and c_{42}, involve variables after x_2.

Accordingly, we have this relaxed version of weak tightness. which does not require a constraint to be tight on *each* of its variables.

Definition 1. *A constraint network is* directionally *weakly m-tight at level k with respect to an order of variables iff for every set of variables $Y = \{x_1, \cdots, x_l\}$(l:k..n-1) and a new variable x , there exists an m-tight directionally relevant constraint on x.*

Theorem 5. *Given a network, let r be the maximum arity of its constraints. If it is directionally weakly m-tight at level $(m+1)(r-1)+1$ and is strongly directionally $(m+1)(r-1)+1$-consistent, then it is strongly directionally n-consistent.*

The proof is similar to that of Theorem 1.

Next we present a sufficient condition for a network to be directionally weakly m-tight.

Theorem 6. *A network of arbitrary constraints is directionally weakly m-tight at level k with respect to a variable ordering if for all $i > k$, there are at least $i-k$ directionally relevant binary constraints which are m-tight on the i_{th} variable.*

The proof is similar to that of Theorem 2. The total number of (partially) m-tight constraints needed is about $1+2+\cdots+(n-(k-1)) = (n-k+2)(n-k+1)/2$. In other words, for a network to be directionally weakly m-tight, we still need a significant number of constraints each of which is m-tight (on certain variables).

Again the results here apply to *proper m-tightness.*

If a network is not directionally k-consistent, we can enforce directional consistency on it. An algorithm to enforce directional consistency is significantly different from that for k-consistency. There exists an ordering of variables on which to enforce directional consistency – the reverse of the given variable ordering – such that no iterative propagation is necessary. This observation leads to the concept of adaptive consistency under which a solution can be found without backtracking.

5 Dually adaptive consistency

One main purpose of our characterization of weak tightness of a network is to help identify a consistency condition under which a solution of a network can be found without backtracking (i.e., efficiently).

The idea of adaptive consistency [5] is to enforce only the necessary "amount" of directional consistency on a network to ensure global consistency. Specifically, for any variable x, we only need to be sure that a consistent instantiation of the variables in the directionally relevant constraints on x is consistently extensible to x. The variables outside its directionally relevant constraints do not play any direct role on x and thus can be ignored.

The *width* of a variable with respect to a variable ordering is the number of the directionally relevant constraints on it. See Fig. 4 for an example.

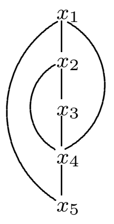

Fig. 4. The variables $\{x_1, x_2, \cdots, x_5\}$ are ordered according to their subscripts. For example, x_1 is before x_2. The width of x_2 is 1.

Example Using the network and variable ordering in Fig. 4, let us try to figure out ideal conditions under which a solution can be found by a backtracking free search. Assume the domain of x_1 is not empty. Pick a value and assign it to x_1. *If* c_{12} *is directionally (arc)* 2*-consistent*, the existence of a value in x_2 is ensured. *If* c_{23} *is directionally arc consistent*, there is a value for x_3 compatible to the previous assignment. As for x_4, *if its three directionally relevant constraints* $\{c_{14}, c_{24}, c_{34}\}$ *"agree" on* x_4 *(e.g., directionally* 4*-consistent on* x_4*)*, there exists a value for x_4 to satisfy the constraints so far. Finally, *if* c_{45} *and* c_{15} *"agree" on* x_5 *(e.g., directionally* 3*-consistent on* x_5*)*, we get a solution for the whole network.

From this example, we can see that different levels of consistency on different variables are sufficient to enable backtracking free search. It is not necessary to have a uniform (e.g., directionally k-) consistency everywhere on a network to enable a backtracking free search.

Here it is also convenient (for presentation and understanding) to use the *agreement* of the directionally relevant constraints on a variable, which will be formalized below.

Definition 2. *Given a network, a variable ordering, and a variable x, let $R(x)$ be the set of directionally relevant constraints on x and S be the union of the scopes of the constraints of $R(x)$. The constraints of $R(x)$ are* consistent *on x, if and only if for any consistent instantiation $\bar{a}$ of $S - \{x\}$, there exists $u \in D_x$ such that $(\bar{a}, u)$ satisfies all the constraints in $R(x)$.*

Now we are able to define the adaptive consistency of a network.

Definition 3. *Given a constraint network and an ordering on its variables. It is* adaptively consistent *if and only if for any variable x, its directionally relevant constraints are consistent on x.*

The adaptive consistency is presented as an algorithm in [5, 4]. However, for the purpose of this paper, we prefer a declarative characterization. It leads to the following consistency result.

Theorem 7. *Given a constraint network and an ordering on its variables. It is strongly directionally n-consistent if it is adaptively consistent.*

It can be proved in the way as in the motivating example above.

When a network is not adaptively consistent, we can make the directionally relevant constraints on each variable – *in the reverse of the variable ordering* – consistent. This is exactly the algorithm in [4, page 105].

Adaptive consistency is not only more accurate in estimating the local consistency sufficient for global consistency, but also makes intuitive the algorithms to enforce consistency and to find a solution.

With the knowledge of tightness of constraints presented in the previous sections, we know that for a network to be adaptively consistent, it is sufficient to make sure that only some, not all, directionally relevant constraints on a variable are consistent.

We define a network with this new tightness property.

Definition 4. *Given a constraint network and an ordering of its variables. The network is* adaptively *m-tight if and only if for any variable x, there exists an m-tight directionally relevant constraint on it.*

We have the following sufficient condition for such a network to be adaptively consistent and thus globally consistent.

Theorem 8. *Given a constraint network and an ordering of its variables, assume the network is adaptively m-tight. The network is adaptively consistent if*

1) for any variable x whose width is not greater than m, the directionally relevant constraints on it are consistent, and

2) for any variable x whose width is greater than m, every $m + 1$ of the directionally relevant constraints on it are consistent.

We omit the proof which is similar to that of Theorem 9.

From this theorem and Theorem 7, we only need the tightest (either binary or non-binary) directionally relevant constraint on each variable (totally $n-1$ such constraints) to predict the global consistency of a network. This could be considered a significant improvement over the results in the previous two sections.

Compared with the result in [5], this theorem also provides a lower level (the smaller of tightness or width) of consistency ensuring global consistency.

Before the introduction of a natural extension of adaptive consistency (Definition 3) – dually adaptive consistency – we present a new result on set intersection.

Lemma 1. *Given a number m and a collection of sets $\{E_1, \cdots, E_l\}$, assume there is a set E among them such that $|E| \leq m$. $\bigcap_{i \in 1..l} E_i \neq \emptyset$ iff the intersection of E and every other m sets is not empty.*

Proof. The necessary condition is clear.

The sufficient condition is proved by induction on l. It is obviously true when $l \leq m+1$. Assuming the intersection of every k ($> m$) sets is not empty, we show that any $k+1$ sets intersect. Without loss of generality, the subscripts of the $k+1$ sets are numbered from 1 to $k+1$ and let $|E_1| \leq m$. Let A_i be the intersection of all the $k+1$ sets except E_i:

$$A_i = E_1 \cap \cdots \cap E_{i-1} \cap E_{i+1} \cap \cdots \cap E_{k+1}, \ for \ 1 < i \leq k+1.$$

If $A_i \cap A_j \neq \emptyset$ for some $i, j \in 2..k+1, i \neq j$,

$$\bigcap_{i \in 1..k+1} E_i = A_i \cap A_j \neq \emptyset.$$

Let $A_i \cap A_j = \emptyset$ for all distinct i and j. In terms of the construction of A_i's, $E_1 \supseteq \bigcup_{i \in 2..k+1} A_i$. $|A_i| \geq 1$ by the induction assumption. Hence,

$$|E_1| \geq \sum_{i \in 2..k+1} |A_i| \geq k > m$$

which contradicts $|E_1| \leq m$. □

This result differs from the *small set intersection* lemma [11] in that the latter demands the intersection of *every* $m+1$ of the l sets not be empty.

Definition 5. *Given a constraint network and an ordering of its variables, let c_x be one of the tightest directionally relevant constraints on x and m_x be its tightness. It is* dually adaptively consistent *if and only if*

1) for any variable x whose width is not greater than m_x, the directionally relevant constraints on it are consistent, and

2) for any variable x whose width is greater than m_x, c_x is consistent with every other m_x directionally relevant constraints on x.

Now, we have the main result of this section.

Theorem 9. *Given a constraint network and an ordering of its variables, it is strongly directionally n-consistent if it is dually adaptively consistent.*

Proof. We only need to prove that the network is adaptively consistent: for any variable x, its directionally relevant constraints $DR(x)$ are consistent on x. Let S be the variables involved in $DR(x)$. Consider any consistent instantiation $\bar{a}$ of $S - \{x\}$. We show that there exists $u \in D_x$ such that $(\bar{a}, u)$ satisfies constraints in $DR(x)$. Let l be the number of constraints in $DR(x)$, and let c_x be one of the tightest constraint in $DR(x)$ with tightness m_x. For any constraint $c_i \in DR(x)(i : 1..l)$, let $\bar{a}$'s image on x under c_i be E_i. It is sufficient to show

$$\cap_{i \in 1..l} E_i \neq \emptyset.$$

We know c_x is consistent with every other m_x constraints. Hence, E_x, $\bar{a}$'s image under c_x, intersects with every other m_x images of $\bar{a}$. The set intersection lemma implies that

$$\cap_{i \in 1..l} E_i \neq \emptyset.$$

□

Compared with Theorem 8, this result requires fewer constraints to be consistent by making use of the new property of tightness. For example, assuming there is a 1-tight constraint in $DR(x)$, Theorem 8 demands *every pair* of constraints in $DR(x)$ be consistent while dually adaptive consistency requires the 1-tight constraint is consistent with every other constraint in $DR(x)$ on x.

Remark on Proper Tightness Dually adaptive consistency can also be defined by *proper m-tightness*. However, when we check whether a constraint network is adaptively consistent, we use m-tightness which is weaker than proper m-tightness. If a network is not adaptively consistent, we use *proper* m-tightness to predict the work needed to enforce adaptive consistency on it because proper m-tightness is preserved in the enforcing process while m-tightness is not.

When a constraint network is not dually adaptively consistent with respect to a variable ordering, it can be made so by enforcing the required consistency on each variable, in the reverse order of the given ordering. To make the procedure more efficient, we should chose a better variable ordering, depending on both the topological structure and tightness of the constraints.

Improving Bucket Elimination on Constraint Networks Bucket elimination [3] is an algorithmic framework which unifies the algorithms from traditional Operations Research, Constraint Networks, Probabilistic Reasoning and other related fields. In Constraint Networks, it is exactly the adaptive consistency. For a variable x, the fact that we enforce all its directionally relevant constraints consistent on x, is described as joining (a Database operation of *natural join*) all the constraints (taken as relations) and projecting away x (and thus eliminating x) in bucket elimination. We know that both time and space complexity of the join operation is exponential to the number of constraints involved. In terms of dually adaptive consistency, if one of the constraints c_x is

m-tight and m is smaller than the number of constraints of concern, we only need to join c_x and every other m constraints and then project away x. An extreme case is that if a constraint c_x is 1-tight, it is sufficient to join c_x and every other constraint of concern, and then project away x. This is not only more efficient, but also introduces a novel way of variable elimination: *We do not eliminate a variable with respect to a whole bucket of constraints, but eliminate it with respect to many smaller buckets while achieving the same consistency on all constraints in the bucket.* We also notice that in the process of making a constraint network dually adaptively consistent, more constraints (with possibly smaller arity) will be generated than in bucket elimination. In the future, we are interested in working out where the balance of cost is between bucket elimination and enforcing a dually adaptive consistency on a network.

Possible Applications Due to the high time and space complexity of bucket elimination and enforcing dually adaptive consistency on a constraint network, they may not be directly used to solve any problem, which is also one reason why we introduce dually adaptive consistency in a declarative, rather than an algorithmic, way. However, the idea behind them may be useful in speeding up the search procedure for certain problems. For example, bucket elimination has inspired some heuristics to solve constraint networks (e.g., [7]). We plan to investigate whether any improvement can be made by applying dually adaptive consistency (or even using more semantics than tightness of constraints) to those problems for which bucket elimination provides promising heuristics.

6 Conclusion

The theme of this paper is to study the impact of the tightness of constraints on the global consistency of a network. Specifically, the tightness of the constraints determines the level of local consistency sufficient to guarantee global consistency. Under the concept of k-consistency, to determine the local consistency ensuring global consistency, we show that it is sufficient to consider only some of the binary constraints. We also show that a weakly tight constraint network *does* need a significant number of constraints to be tight. After studying directional consistency, we propose a new type of consistency – dually adaptive consistency – which considers not only the topological structure, but also the tightness of the constraints in a network. Based on this concept, only the tightest (in a local sense) constraints or the widths of variables, depending on which are smaller, determine the local consistency ensuring global consistency.

The tightness of constraints can also be naturally employed to improve the efficiency of *bucket elimination* algorithms for constraint networks [3]. We also note that *Pivot consistency* proposed by David [1] is a special case of the dually adaptive consistency where 1-tight constraints (functional constraints) are considered.

The dually adaptive consistency may be helpful where the heuristics from bucket elimination have shown some promise. Having shown that theoretically there is a close relationship between the tightness of constraints and global consistency, in the future, we will explore whether the tightness of constraints can play greater role in solving practical constraint networks.

7 Acknowledgement

I thank Angela Glover, Georg Ringwelski, and other members at 4C for their helpful comments on an earlier draft of this paper.

References

1. P. David. Using pivot consistency to decompose and solve functional CSPs. *Journal of Arificial Intelligence Research*, 2:447–474, 1995.
2. R. Dechter. From local to global consistency. *Artificial Intelligence*, 55:87–107, 1992.
3. R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.
4. R. Dechter. *Constraint Processing*. Morgan Kaufmann, San Francisco, 2003.
5. R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1987.
6. E.C. Freuder. Synthesizing constraint expressions. *Communications of ACM*, 21(11):958–966, 1978.
7. J. Larrosa and R. Dechter. Boosting search with variable elimination in constraint optimization and constraint satisfaction problems. *Constraints*, 8(3):303–326, 2003.
8. A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):118–126, 1977.
9. U. Montanari. Networks of constraints: Fundamental properties and applications. *Information Science*, 7(2):95–132, 1974.
10. P. van Beek and R. Dechter. Constraint tightness and looseness versus local and global consistency. *Journal of The ACM*, 44(4):549–566, 1997.
11. Yuanlin Zhang and Roland H. C. Yap. Consistency and set intersection. In *Proceedings of IJCAI-03*, pages 263–268, Acapulco, Mexico, 2003. IJCAI Inc.