Extending Dual Arc Consistency

S. Nagarajan*
QNX Software Systems,
175, Terence Matthews Crescent,
Kanata, Ontario, Canada
E-mail: shiv@nagarajan.net

S. Goodwin
Dept. of Computer Science,
University of Windsor,
Windsor, Ontario, Canada
Email: sgoodwin@cs.uwindsor.ca

A. Sattar
School of Comp. and Inf. Tech.,
Griffith Univ., Gold Coast Campus,
Queensland, Australia.
Email: sattar@cit.gu.edu.au

Abstract

Comparisons between primal and dual approaches have recently been extensively studied and evaluated from a theoretical standpoint based on the amount of pruning achieved by each of these when applied to non-binary constraint satisfaction problems. Enforcing arc consistency on the dual encoding has been shown to strictly dominate enforcing GAC on the primal encoding (Stergiou & Walsh 1999). More recently, extensions to dual arc consistency have extended these results to dual encodings that are based on the construction of compact constraint coverings, that retain the completeness of the encodings, while using a fraction of the space. In this paper we present a complete theoretical evaluation of these different consistency techniques and also demonstrate how arbitrarily high levels of consistency can be achieved efficiently using them.

Introduction

Recently, a lot of research has gone into the development of techniques that can directly handle non-binary constraints. On one hand, many extensions to existing binary constraint satisfaction algorithms have been proposed that directly deal with the non-binary constraints (Bessière *et al.* 1999). The other choice is to perform a structural transformation of the representation of the problem, so that the resulting problem is a binary CSP except that now the original constraints which were non-binary are replaced by binary compatibility constraints between relations. A lot of recent work has been concerned with comparing different levels of local consistency enforceable in the non-binary representation with the dual representation. The dual encoding can often enforce high levels of consistency when compared to the primal representations (Bacchus & van Beek 1998; Stergiou & Walsh 1999). In some cases the space complexity of the dual encodings is prohibitive and this is sometimes a drawback when trying to use these encodings. More recently (Nagarajan *et al.* 2000) modifications to the standard dual encoding have been proposed that can compactly represent the given CSP using an equivalent dual encoding that contains all the original solutions to the CSP, using constraint coverings. It has also been shown that enforcing arc consistency in these constraint covering based encodings, strictly dominates enforcement of GAC on the primal non-binary encoding. In this paper we present a complete analysis of these covering based encodings and the consistencies that can be enforced using them.

*This paper reports work that was conducted when the author was at the Dept. of Computer Science, at the University of Regina, Regina, Canada.

Background

In this section we present some preliminary definitions and background information.

Definition 1 *A constraint C_i on an ordered set of variables $V_i = (v_{i_1}, \ldots, v_{i_k}) \subseteq V$, is defined as a predicate on these variables, $S_i \subseteq D(v_{i_1}) \times \cdots \times D(v_{i_k})$. The set of subsets $\{V_1, \ldots, V_M\}$ on which constraints are specified is called the* **scheme** *of N. The number of variables in a constraint is called the* **arity** *of the constraint. A constraint network in which each constraint is of arity 2 is called a* **binary constraint network***, and the problem that it represents is called a* **binary CSP***.*

Definition 2 *Given a constraint C_i on variables V_i the set S_i is a subset of the Cartesian product $D(v_{i_1}) \times \cdots \times D(v_{i_k})$ that specifies the set of allowable combinations of values for the variables $v_{i_1} \times \cdots \times v_{i_k}$. An element $t_i \in S_i$ is known as a tuple on V_i.*

Definition 3 *Given a binary CSP, the* **primal constraint graph** *associated with it is a labeled constraint graph, where N=V, $(v_i, v_j) \in A$ **iff** $\exists C_{ij} \in C \mid V_{ij} = \{v_i, v_j\}$. Also the label on arc (v_i, v_j) is C_{ij}.*

Definition 4 *Given an arbitrary CSP, the* **dual constraint graph** *associated with it is a labeled graph, where N=C, $(C_i, C_j) \in A$ **iff** $V_i \cap V_j \neq \emptyset$. Also the label on arc (C_i, C_j) is $V_i \cap V_j$.*

Intuitively $|N|=|V|$, and $|A|=|C|$, and an arc $a \in A$ that connects two variables connected by constraint c, is labeled by the definition of c. This representation is good for binary CSPs but is not as useful for general CSPs. The primal graph for higher order CSPs is a hypergraph. The dual graph constraint network can be solved by techniques that are applicable to binary networks by considering the constraints as the variables and tuples that instantiate them as the domains.

Definition 5 *If V_i and V_j are sets of variables, let S_i be an instantiation of the variables in V_i. $S_i[V_j]$ is the tuple consisting of only the components of V_i that correspond to the variables in V_j. This is also called the* **projection** *of tuple S_i on the variables in V_j.*

We say that an instantiation t_x on the variables in V_x is **consistent** with respect to a constraint network N, iff for all V_i in the scheme of N such that $V_i \subseteq V_x$, $t_x[V_i] \in S_i$. If we enumerate all consistent instantiations of variables in V_x, we get a set of all solutions of the subnetwork defined by V_x.

Definition 6 *Two constraints $C_i, C_j \in C$ are* **consistent** *if either C_i and C_j are not connected, or the induced constraints of C_i and C_j on $V_i \cap V_j$ can be satisfied simultaneously by at least one instantiation of the variables in $V_i \cap V_j$.*

Clearly, if there exist $C_i, C_j \in C$ such that C_i and C_j are not consistent, then there is no solution to the given problem. Arc consistency has been used to enforce local consistency in binary CSPs with a lot of success. This definition of AC is not applicable directly to non-binary constraints. Arc consistency is extended for non-binary constraints as generalised arc consistency (GAC). A non-binary CSP is GAC **iff** for any variable in a constraint and a value that is assigned to it there exist compatible values for all other variables in the constraint (Mohr & Masini 1988).

Definition 7 *A tuple t on $(v_{i_1}, \ldots, v_{i_q})$ is valid* **iff** $t \in D(v_{i_1}) \times \ldots \times D(v_{i_q})$. *A CSP is said to be* **generalised arc consistent** *(GAC) if $\forall v_i \in V, \forall val_i \in D(v_i), \forall C_j \in C, \exists t \in S_j$ such that t is valid and $t[v_i] = val_i$.*

Given a dual encoding of a non-binary CSP, one can define arc consistency in terms of the variables in the dual variables and the tuples in the various constraints. This form of local consistency has been defined for non binary CSPs known as *pair-wise consistency*. Pair-wise consistency was originally introduced in databases, and is also called dual arc consistency.

Definition 8 ((Beeri *et al.* 1983)) *Given a CSP , iff $\forall C_i, C_j, S_i[V_i \cup V_j] = S_j[V_i \cup V_j]$ and $\forall S_i, S_i \neq \emptyset$, this CSP is said to be* **pair-wise consistent**.

Generalised dual arc consistency

If a binary CSP is arc consistent then there is always a consistent instantiation to any pair of variables. But in a general (non-binary) CSP pair-wise consistency does not guarantee a consistent instantiation to the variables involved in every pair of constraints. This is because a consistent instantiation to a pair of constraints must satisfy both the constraints in question and also all constraints that are posed on all the variables involved. Although pair-wise consistency guarantees that the common variables between constraints are assigned consistent values, the other constraints on the variables are not necessarily satisfied. In (Pang 1998) this insight regarding pair-wise consistency led to the definition of ω-consistency. Enforcing ω-consistency removes tuples from constraints that cannot participate in any solution.

Definition 9 *Given two constraints C_i and C_j, the tuple $t_{j,b} \in S_j$ is called a* **ω-support** *for tuple $t_{i,a} \in S_i$, if $t_{i,a}[V_i \cap V_j] = t_{j,b}[V_i \cap V_j]$ and $\forall C_k \in C | V_k \subseteq (V_i \cup V_j)$, $(t_{i,a} \bowtie t_{j,b})[V_k] \in S_k$. A tuple $t_{i,a}$ in a constraint C_i is* **ω-viable iff** *for every constraint C_j, tuple $t_{i,a}$ has ω-support in C_j. A constraint network is* **ω-consistent**, **iff** *for every constraint C_i, $S_i \neq \emptyset$ and all the tuples in S_i are ω-viable.*

Basically, enforcing ω-consistency on a set of constraints, solves the sub-problem induced by these constraints on the original CSP. Just like pair-wise consistency, ω-consistency is applicable to both binary and non-binary CSPs . By extending the algorithm for Dual arc consistency given earlier we can get an algorithm for enforcing ω-consistency.

In (Nagarajan *et al.* 2000) this was generalised to generalised dual arc consistency.[1] Generalised dual arc consistency (GDAC) is also defined on the dual encoding, and is an extension of pair wise consistency and ω-consistency that takes into account projections of constraint relations on subsets of variables while enumerating supports for the tuples.

Definition 10 *Given two constraints C_i and C_j, the tuple $t_{j,b} \in S_j$ is called a* **generalised dual arc support** *for tuple $t_{i,a} \in S_i$, if $t_{i,a}[V_i \cap V_j] = t_{j,b}[V_i \cap V_j]$ and $\forall C_k \in C | (V_k \cap (V_i \cup V_j)) \neq \emptyset$, $(t_{i,a} \bowtie t_{j,b})[V_k] \in S_k$. A tuple $t_{i,a}$ in a constraint C_i is* **generalised dual arc viable iff** *for every constraint C_j, tuple $t_{i,a}$ has generalised dual arc support in C_j. A constraint network is* **generalised dual arc consistent**, **iff** *for every constraint C_i, $S_i \neq \emptyset$ and all the tuples in S_i are generalised dual arc viable.*

In addition to verifying that all the pairs of tuples ($t_{i,a}$ and $t_{j,b}$) in the constraints are pair-wise compatible, generalised dual arc consistency also verifies that all constraints that share variables with $t_{i,a} \bowtie t_{j,b}$ are also compatible with them. In order to demonstrate the fact that both ω-consistency and GDAC enforce very high levels of consistency in the CSPs we present the following theorems.

Theorem 1 *If $\exists C_i \in C$ such that $|V_i| = n$, where n is the arity of the CSP, then enforcing ω-consistency or GDAC on the CSP is equivalent to solving the CSP.*

Proof Enforcing ω-consistency on the CSP, in addition to enforcing pair-wise consistency between all pairs of variables, also ensures that each pair of constraints satisfies all constraints contained in the sub-problem induced by them. In the case that there is an n-ary constraint included in the CSP, the induced sub-problem contains all the constraints and hence enforcing ω-consistency and GDAC is equivalent to solving the CSP. □

Theorem 2 *If $\exists C_i, C_j \in C$ such that $V_i \cup V_j = V$, where n is the arity of the CSP, then enforcing ω-consistency or GDAC on the CSP is equivalent to solving the CSP.*

Proof As in theorem 1, both ω-consistency and GDAC solve the sub-problem induced by every pair of constraints. Given two constraints C_i and C_j whose scopes are V_i and V_j respectively, such that $(V_i \cup V_j) = V$, the sub-problem induced by this pair includes all the constraints in the CSP. Hence enforcing ω-consistency or GDAC is equivalent to solving the CSP. □

[1] GDAC was called covering arc consistency.

As seen in theorems 1 and 2, both ω-consistency and GDAC enforce consistencies that are effectively as high as global consistency in some cases.

Analysis

An arc consistency algorithm removes all arc inconsistent values from the domains of the variables of the encoding. Constraint propagation (as performed by an arc consistency algorithms) infers *no-goods* in both the primal and the dual domains.

To theoretically compare the amount of pruning achieved by enforcing one form of arc consistency on a CSP with other forms of arc consistency, Stergiou and Walsh (Stergiou & Walsh 1999) define a scheme to compare the various *no-goods* derived in the different encodings. Constraint propagation in the dual might infer *no-goods* involving dual variables and these cannot be directly compared with the no-goods inferred in the original problem using generalised arc consistency. But, one can translate the *no-goods* derived in the dual into *no-goods* involving the original variables and values. i.e., If constraint propagation in the dual encoding removes all tuples from a dual variable that assign a value val_i, to a variable v_i, we can derive a single *no-good* that removes val_i from the domain of v_i in the original problem. Hence one can compare the *no-goods* in the original non-binary problem using arc consistency, with *no-goods* that can be derived from the dual arc inconsistent tuples.

In (Stergiou & Walsh 1999) enforcing arc consistency in the two binary encodings for non-binary CSPs, the dual encoding and the hidden variable encoding are compared to GAC. The following theorems are proven in (Stergiou & Walsh 1999).

Theorem 3 *Enforcing AC on the hidden variable encoding is equivalent to enforcing GAC on the variables in the original problem.*

Theorem 4 *Enforcing AC on the dual encoding is strictly stronger than enforcing GAC on the original problem.*

Theorem 5 *Enforcing AC on the dual encoding is strictly stronger than enforcing AC on the hidden variable encoding.*

The above results indicate that enforcing AC in the dual derives more *no-goods* than enforcing GAC or AC on the hidden encoding. These results were extended in (Nagarajan *et al.* 2000) to compare GDAC to GAC and PWC.

Theorem 6 *Enforcing GDAC on the dual encoding is strictly stronger than enforcing pair-wise consistency on the dual encoding (and therefore strictly stronger than enforcing GAC or AC on the hidden variable encoding).*

Covering based dual encodings

Intuitively, a tuple, t_i, is consistent if it satisfies all the constraints whose variables are completely instantiated by t_i. A complete solution is a consistent instantiation of all the variables. The goal of CSP solving algorithms is to find one (or all) consistent extensions on n variables. Given the set of all constraints in the CSP, a special subset of constraints called a constraint cover can be defined as follows.

Definition 11 *Let $C_{cover} = \{C_1, C_2, \ldots, C_m\}$. Also $C_{cover} \subseteq C$. Each $C_i \in C_{cover}$ is given as $\langle V_i, S_i \rangle$, where $V_i \subseteq V$. C_{cover}* **covers** *V* **iff** *$\bigcup_{i=1}^{m} V_i = V$. C_{cover} is a* **constraint cover** *of V. As well, C_{cover} is a* **minimal constraint cover** *of V if it is a constraint cover of V and no proper subset of C_{cover} is a constraint cover of V.*

Given a constraint cover, if one tuple is selected from each constraint in the cover, the relational join of these $|C_{cover}|$ tuples is a tuple on n variables. It can easily be shown that the covering based encoding, even though it includes only a subset of all the constraints, still contains all the solutions to the original CSP. Any method that enforces consistency on this covering based encoding (e.g. GDAC or ω consistency or forward checking that enforces these consistencies) is both sound and complete. Given a constraint cover, $C_{cover} = \{C_1, C_2, \ldots, C_m\}$ if $m > |V|$, $\exists C_i \in C_{cover}$ such that C_{cover}-C_i is still a constraint cover. Although the size of a minimal constraint cover is upper bounded by $|V|$, in practice in CSPs of higher arities, this number is even less.

We now re-define GDAC in terms of the covering based dual encoding. Instead of searching for support for values in the domains of the dual variables for every pair of values, the arc consistency algorithm w.r.t. a covering only searches for support for values in dual variables that are actually in the constraint covering.

Definition 12 *Consider a covering based dual encoding of a CSP with $C_{cover} = \{C_1, C_2, \ldots, C_m\}$. Given two constraints $C_i, C_j \in C_{cover}$, the tuple $t_j \in S_j$ is called a* **generalised dual arc support** *for tuple $t_i \in S_i$ w.r.t. C_{cover}, if $t_i[V_i \cap V_j] = t_j[V_i \cap V_j]$ and $\forall C_x \in \{C\text{-}\{C_i, C_j\}\}$, $(t_i \bowtie t_j)[V_{ij} \cap V_x] \in S_x[V_{ij} \cap V_x]$. A tuple $t_i \in C_i \in C_{cover}$ is* **viable iff** *for every constraint $C_j \in C_{cover}$, tuple t_i has generalised dual arc support in C_j w.r.t. C_{cover}. A constraint network is* **generalised dual arc consistent (GDAC)** *w.r.t. a covering C_{cover}, if $\forall C_i \in C_{cover}$, all the tuples in S_i are viable.*

Since there are a few ways to generate constraint covers, we now analyse the pruning achievable by special kinds of constraint covers.

Definition 13 *Given a CSP, a set of constraints $C_{cover} \subseteq C$ is called an ω-cover* **iff** *C_{cover} is a covering, and $\forall C_i \notin C_{cover}$, $\exists C_p, C_q \in C_{cover}$ such that $V_i \subseteq (V_p \cup V_q)$. A cover C_{cover} is called a minimal ω-cover if no subset of C_{cover} is an ω-cover.*

Definition 14 *Given a CSP, a set of constraints $C_{cover} \subseteq C$ is called an i-cover* **iff** *C_{cover} is a covering, and $\forall C_i, C_j \notin C_{cover}$, $\exists C_p, C_q \in C_{cover}$, such that $(V_i \cap V_j) \subseteq (V_p \cup V_q)$. Again, a cover C_{cover} is called a minimal i-cover if no subset of C_{cover} is an i-cover.*

Clearly every w-cover is an i-cover. Consider an ω-cover, C_{cover}. $\forall C_i \notin C_{cover}$, $\exists C_p, C_q \in C_{cover}$, such that $V_i \subseteq (V_p \cup V_q)$. For any V_i, $(V_i \cap V_j) \subseteq V_i$. Hence, for the same C_i and any $C_j \notin C_{cover}$, $(V_i \cap V_j) \subseteq (V_p \cup V_q)$. This is exactly the condition for an i-cover. ω-covers were briefly described in (Pang 1998)

Theorem 7 *If C_{cover} is a ω-cover, pair-wise consistency on the standard dual encoding prunes at most as much as GDAC on the covering based dual encoding w.r.t C_{cover}.*

Proof Pair-wise consistency find inconsistencies between pairs of constraints. GDAC performs pair-wise consistency on all the pairs of constraints in the covering. Given a constraint covering C_{cover}, if it is the case that $\forall C_i \notin C_{cover}$, $\exists C_p, C_q \in C_{cover}$ such that $V_i \subseteq (V_p \cup V_q)$, the algorithm enforcing GDAC will find all inconsistent tuples in all $C_i \notin C_{cover}$ by projection. Hence under this condition, pair-wise consistency prunes at most as much as GDAC on the covering based dual encoding w.r.t. a given C_{cover}. □

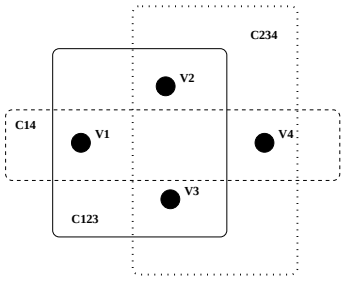
Theorem 8 *If C_{cover} is an i-cover, GDAC on the covering based dual encoding w.r.t. C_{cover} prunes at least as much as pair-wise consistency on the standard dual encoding.*

Proof If it is the case that $\forall C_i, C_j \notin C_{cover}$ if $\exists C_p, C_q \in C_{cover}$, $(V_i \cap V_j) \subseteq (V_p \cup V_q)$, then GDAC .w.r.t. C_{cover} derives all pair-wise inconsistent tuples that AC on the dual encoding would derive. (This is because all sets of common variables between pairs of constraints $\notin C_{cover}$ are subsumed by either the $\bowtie$ of two other constraints in C_{cover} or by the projection of that $\bowtie$ onto the constraints in C_{cover}). Hence this is precisely the condition when GDAC w.r.t. a cover is no worse than dual AC. □

Theorem 9 *Achieving GDAC on the constraint covering based dual encoding w.r.t. an arbitrary cover is incomparable to achieving AC on the standard dual encoding.*

Proof To show that enforcing GDAC on the covering based dual encoding and enforcing AC on the standard encoding are incomparable, all that is required is to show a) a problem where enforcing GDAC on the covering based dual encoding prunes more than AC on the standard dual encoding, and b) another problem where AC on the standard dual encoding prunes more than GDAC on the covering based dual encoding. To show a) we can consider the following example.

Consider the following example taken from (Bessière 1999). Here is a very simple CSP, with 4 variables, v_1, v_2, v_3 and v_4. The domain of each of the variables is a, b, c. The CSP is shown in figure 1. The CSP has three constraints, C_{123}, C_{234} and C_{14}. This problem is GAC, and all values in the domains of all the variables are *viable*. As a result an algorithm that enforces GAC, will not remove any values from the domains of the variables in the original problem.



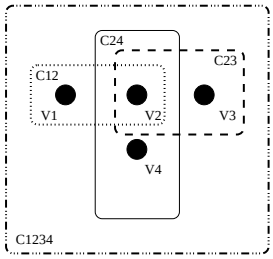
C123:		
V1	V2	V3
a	a	a
a	a	b
b	b	b
b	b	c
c	c	a
c	c	b

C234:		
V2	V3	V4
a	a	a
a	a	b
b	b	b
b	b	c
c	c	a
c	c	b

C14:	
V1	V4
a	c
b	a
b	b
c	a
c	b
c	c

Figure 1: Non-Binary CSP: An example

While enforcing GDAC, from the set of given constraints, C=$\{C_{123}, C_{234}, C_{14}\}$, we can construct a minimal constraint covering, by considering any two of the three constraints. From definition 12, the generalised dual arc consistency algorithm would enforce pair-wise consistency between pairs of constraints in a covering, while ensuring that the relational join of the pairs of constraints is consistent with rest of the constraints in C. The only dual domains that are pruned of values are the constraints in the covering. A constraint cover is constructed as C_{cover}=$\{C_{123}, C_{14}\}$. The pruning achieved by enforcing generalised dual arc consistency is given in the figure. The derived *no-goods* when translated back to the domains of the variables in the original problem, reduce the domains to singleton domains. For b) consider the following example. Given a CSP on 6 variables with binary domains, $\{a, b, c, d, e, f\}$. There are 6 constraints, which are given as follows.
$C_{a,b}$=$\{(0,0)\}$
$C_{b,c}$=$\{(0,0)\}$
$C_{d,e}$=$\{(0,1)\}$
$C_{e,f}$=$\{(1,0)\}$
$C_{a,b,c,d,e}$=$\{(0,0,0,0,1)\}$
$C_{a,b,c,d,f}$=$\{(0,0,1,0,0),(0,0,0,1,0),(1,0,0,0,0)\}$
Consider a constraint covering C_{cover}= $\{C_{a,b}, C_{b,c}, C_{d,e}, C_{e,f}\}$. This CSP is generalised dual arc consistent w.r.t. the covering C_{cover}. But enforcing AC on the standard dual encoding, will prove that the problem is insoluble since pair-wise consistency between $C_{a,b,c,d,e}$ and $C_{a,b,c,d,f}$ will fail. Hence there is a problem where AC on the dual encoding prunes more than GDAC on the covering based dual encoding. □



C12	C23	C24	C1234
0 0	0 0	0 0	1 0 0 0
			1 0 1 0
			1 1 0 1
			1 1 1 1
			1 0 0 1
			1 0 1 1

C_cover = {C12. C23. C24}

w_cover = {C1234}

Figure 2: CSP that is ω-consistent w.r.t. a cover

Now consider the example in figure 2. The CSP has 4 variables, and 4 constraints. The variables all have domains 0..1. The constraints are $\{C_{1,2}, C_{2,3}, C_{2,4}, C_{1,2,3,4}\}$. This problem is not generalised arc consistent, and enforcing GAC on it will show that the problem has no solution. Consider a cover C_{cover}=$\{C_{1,2}, C_{2,3}, C_{2,4}\}$. Now, this is not an ω-cover, but enforcing ω-consistency w.r.t. to C_{cover}, would see that the CSP is ω-consistent w.r.t. C_{cover}. On the other hand, enforcing GDAC w.r.t. the same cover C_{cover} would also show that the problem has no solution. It is interesting to see that this problem has only one minimal ω-cover, which includes the 4-ary constraint, $C_{1,2,3,4}$.

Theorem 10 *If C_{cover} is not an ω-cover, enforcing ω-consistency w.r.t. C_{cover} is incomparable to enforcing*

GAC on the original problem

Proof To show the ω-consistency w.r.t. an arbitrary C_{cover} that is not a ω-cover, is incomparable to GAC on the original problem, it suffices to show that a) a problem in which GAC $>$ ω-consistency w.r.t. an arbitrary C_{cover} and b) another problem where ω-consistency w.r.t. an arbitrary C_{cover} $>$ GAC where C_{cover} is not an ω-cover in either case. To show a) consider the example given in figure 2. Enforcing ω-consistency w.r.t. the C_{cover}= $\{C_{12}, C_{23}, C_{24}\}$, would recognise the CSP to be ω-consistent. But enforcing GAC would show that the problem has no solutions. (Since w.r.t. variable 1, constraint C_{12} does not assign the value 1 to any tuple in it, while constraint C_{1234} does not assign the value 0 to variable 1 in any of its satisfying tuples. To show b) consider the following CSP, with 4 variables and 5 constraints. The variables all have domains 0..1. The constraints are
$C_{1,2}$=$\{(0,0),(1,1)\}$
$C_{2,3}$=$\{(0,0),(1,1)\}$
$C_{2,4}$=$\{(0,0),(1,1)\}$
$C_{1,2,3,4}$=$\{(0,0,1,0),(1,1,0,1),(0,0,0,0),(1,1,1,1)\}$
$C_{1,3}$=$\{(0,1),(1,0)\}$

Consider the cover C_{cover}=$\{C_{1,2}, C_{2,3}, C_{2,4}\}$. This is not an ω-cover, while the only ω-covers for this CSP are $\{C_{1,2,3,4}\}$ and $\{C_{24}, C_{13}\}$. When we enforce ω-consistency w.r.t. this C_{cover}, one can see that the problem has no solutions, but enforcing GAC on the problem will not remove any inconsistent values. □

Theorem 11 *Enforcing GDAC w.r.t. an aribitary C_{cover} is strictly stronger than enforcing ω-consistency w.r.t. the same C_{cover}.*

Proof Given a constraint covering C_{cover}, GDAC and ω-consistency enforce pair-wise consistency between all pairs of constraints in the covering, and then go on to verify the pairs of constraints against other constraints in the CSP. The set of constraints verified against by GDAC is a superset of the constraints verified by ω-consistency To show strictness consider the example in figure 2. Using C_{cover}=$\{C_{1,2}, C_{2,3}, C_{2,4}\}$, GDAC determines that the problem has no solution without search, while the problem is ω-consistent w.r.t. C_{cover}. □

Theorem 12 *Enforcing ω-consistency on the standard dual encoding is incomparable with enforcing GDAC w.r.t. a constraint covering.*

Proof To prove this we must show that, a) a problem in which enforcing ω-consistency on the standard dual encoding is stronger than enforcing GDAC on a covering based dual encoding and b) another problem in which enforcing GDAC on a covering based dual encoding is stronger than enforcing ω-consistency on the standard dual encoding. To show a) consider the example used in part b) of theorem 9. Consider the constraint covering C_{cover}= $\{C_{a,b}, C_{b,c}, C_{d,e}, C_{e,f}\}$. This CSP is generalised dual arc consistent w.r.t. the covering C_{cover}. But enforcing ω-consistency on the standard encoding will show that this CSP is inconsistent and does not admit any solutions. To show b) consider the following example
$C_{1,2,3}$=$\{(0,1,0),(1,0,1)\}$
$C_{3,4,5}$=$\{(0,0,0),(1,1,1)\}$
$C_{1,4,6}$=$\{(1,0,0),(0,1,1)\}$
$C_{2,3}$=$\{(1,0),(0,1)\}$
$C_{3,5}$=$\{(0,0),(1,1)\}$

This problem is still ω-consistent. Consider a constraint covering C_{cover}=$\{C_{1,2,3}, C_{3,4,5}, C_{1,4,6}\}$. Enforcing GDAC on the problem w.r.t. C_{cover} would show that the problem has no solutions without search. □

Given all these relative orderings between the various forms of local consistency in the dual encodings, it is useful to quickly summarise the various consistencies and their relationship with other consistencies. The results can be summarised in a theorem as:

Theorem 13 *a. GDAC $>$ ωAC $>$ PWC.*
b. GDAC $>$ ωAC $>$ GAC.
c. GDAC $_{\omega-cover}$ $>$ ωAC$_{\omega-cover}$ $>$ PWC.
d. GDAC $_{\omega-cover}$ $>$ ωAC$_{\omega-cover}$ $>$ GAC.
e. GDAC $>$ GDAC $_{\omega-cover}$ $>$ GDAC $_{i-cover}$ $>$ PWC.
f. GDAC $>$ GDAC $_{\omega-cover}$ $>$ GDAC $_{i-cover}$ $>$ GAC.
g. GDAC $_{cover}$ $>$ GAC.
h. GDAC $_{cover}$ $\sim$ ωAC.
i. ωAC$_{cover}$ $\sim$ GAC.

where LC$_{cover}$ is enforcing local consistency property LC on the covering based dual encoding, w.r.t. a cover that is not an ω-cover (i.e., an arbitrary cover). LC$_{\omega-cover}$ is enforcing local consistency property LC on the covering based dual encoding w.r.t. an ω-cover. GDAC, ωAC and PWC are the three local consistencies when applied to the standard dual encoding. Since GAC is equivalent HVAC, the results regarding GAC apply to HVAC too.

As we saw in Theorems 1 and 2 enforcing GDAC and ωAC on some CSPs can be as high as enforcing global consistency on the CSP. This can be translated into a similar result for covering based encodings too.

Lemma 1 *If $\exists C_i \in C_{cover}$ such that $|V_i|$=n, where n is the arity of the CSP, then enforcing ω-consistency or GDAC on the CSP w.r.t. C_{cover} is equivalent to solving the CSP.*

Proof Similar to theorem 1 □

Lemma 2 *If $\exists C_i, C_j \in C_{cover}$ such that $V_i \cup V_j$=V, where n is the arity of the CSP, then enforcing ω-consistency or GDAC on the CSP w.r.t. C_{cover} is equivalent to solving the CSP.*

Proof Similar to theorem 2 □

Algorithms to enforce GDAC

By extending the algorithm to enforce AC for binary CSPs in the primal graph (AC-6), we can construct a similar algorithm to enforce ω-AC in the dual encoding. The space complexity of ω-AC6 is $O(e^2 \cdot d^k)$ where e=$|C|$. The time complexity is $O(e^3 \cdot k \cdot d^{2k})$. The extra e factor when compared to pair-wise consistency is because the every time support is sought, all constraints posed on the variable set are also checked to see if they are satisfied. GDAC has space complexity $O(e^2 \cdot d^k)$ where e=$|C|$ and the time complexity is

$O(e^3 \cdot k \cdot d^{2k})$ given that checking a k-ary constraint takes $O(k)$ time. This complexity is, in theory, the same as that for ω-consistency although, in practice, GDAC performs more constraint checks than ω-consistency since $\{C_k \in C | V_k \subseteq (V_i \cup V_j)\}$ (the set of constraints being checked in ω-consistency) is a subset of $\{C_k \in C | (V_k \cap (V_i \cup V_j)) \neq \emptyset\}$ (the set of constraints being checked in GDAC).

AC algorithms for the covering based encodings

In order to extend these algorithms to enforce GDAC or ω-AC in the covering based encodings, we can just apply the local consistency properties to the covers obtained. A constraint cover can be obtained in advance. The cardinality (size) of a minimal constraint cover is always $\leq |C|$ and $\leq |V|$ and often even lower.

As seen before, the space/time complexity of the AC6 algorithms enforcing ωAC or GDAC are ($O(e^2 \cdot d^k)$/ $O(e^3 \cdot k \cdot d^{2k})$), where e is the number of constraints. When enforced on the constraint covers, e is typically $\leq n$, where n is the number of variables in the problem. The space/time complexity of the AC algorithms on the covering based encodings then become ($O(n^2 \cdot d^k)$/ $O(n^2 \cdot e \cdot k \cdot d^{2k})$). As seen before while the time/space complexity of the AC algorithms has reduced considerably, the level of consistency enforced is still high.

Higher consistencies in the dual

In the same spirit of the extensions made in the primal graph for local consistencies to higher forms of consistencies like singleton consistencies, both GDAC and ω-AC can also be extended to similar singleton consistencies. A binary CSP is **singleton arc consistent** (SAC) **iff** it has non-empty domains and for any instantiation of a variable, the resulting sub-problem can be made arc-consistent. Singleton arc consistency can be achieved by any algorithm that achieves arc consistency. The definition of singleton arc consistency requires that upon assignment of a value to a variable, the resulting problem can be made arc consistent. In (Prosser, Stergiou, & Walsh 2000) singleton consistencies were studied and rank the singleton consistencies within the hierarchy of local consistencies. Among other results, they show that if, for some two local consistency properties LC1 and LC2, LC1 $>$ LC2, then singleton LC1 $>$ singleton LC2. Therefore, given the results described in the earlier sections in this paper we can conclude the following:

Singleton ωAC $>$ Singleton PWC $>$ Singleton GAC
Singleton GDAC $>$ Singleton PWC $>$ Singleton GAC
Singleton GDAC $>$ Singleton ωAC
Singleton GDAC $_{cover}$ $>$ Singleton GAC
Singleton GDAC $_{i-cover}$ $>$ Singleton PWC
Singleton ωAC$_{\omega-cover}$ $>$ Singleton PWC
Singleton GDAC $_{\omega-cover}$ $>$ Singleton ωAC$_{\omega-cover}$

where Singleton ωAC, Singleton GAC , Singleton PWC and Singleton GDAC are the singleton extensions to the different local consistencies. Singleton consistency for each of the local consistency properties can be achieved by using the same algorithm that achieves the relevant local property, by first making the problem LC (local consistent with the relevant property) and then going through each value val_j (tuple tup_j) in the domain of every variable v_i (dual variable V_i) in the CSP, and if the resulting sub-problem with this variable (dual variable) being assigned this value (tuple), cannot be made LC, this value is removed and the LC property restored. This process continues until all inconsistent values are removed and deleted (propagated).

Conclusions

In this paper we have presented extensive theoretical results relating to enforcing high levels of local consistency in the dual encoding. We show how it is possible to efficiently enforce extremely high levels of consistency by the use of constraint coverings. This paper compares and extends many different previous results in enforcing arc consistency in non-binary constraint satisfaction problems. The theoretical results presented here have also been empirically evaluated on non-binary CSPs, and will be available in an extended version of this paper.

References

Bacchus, F., and van Beek, P. 1998. On the conversion between Non-Binary and Binary constraint satisfaction problems. In *Proceedings of the National Conference on Articial Intelligence, AAAI-98*, 311–318. American Association for Artificial Intelligence.

Beeri, C.; Fagin, R.; Maier, D.; and Yannakakis, M. 1983. On the desirability of acyclic database schemes. *Journal of the ACM* 30(3):479–513.

Bessière, C.; Freuder, E. C.; Meseguer, P.; and Larrosa, J. 1999. On forward checking for non-binary constraint satisfaction. In *Principles and Practice of Constraint Programming, CP-99*, volume 1713, 88–102. Springer Verlag.

Bessière, C. 1999. Non-Binary constraints. In *Principles and Practice of Constraint Programming, CP-99, Invited Lecture*.

Mohr, R., and Masini, G. 1988. Good old discrete relaxation. In *Proceedings ECAI-88*, 651–656.

Nagarajan, S.; Goodwin, S.; Sattar, A.; and Thornton, J. 2000. On dual encodings for non-binary constraint satisfaction problems. In *Principles and Practice of Constraint Programming, CP-2000*, volume 1894, 531–536. Springer Verlag.

Pang, W. 1998. *Constraint structure in constraint satisfaction problems*. Ph.D. Dissertation, University of Regina.

Prosser, P.; Stergiou, K.; and Walsh, T. 2000. Singleton consistencies. In *Principles and Practice of Constraint Programming, CP-2000*, volume 1894, 353–368. Springer Verlag.

Stergiou, K., and Walsh, T. 1999. Encodings of non-binary constraint satisfaction problems. In *Proceedings of the National Conference on Articial Intelligence, AAAI-99*, 163–168. American Association for Artificial Intelligence.