On dual encodings for non-binary constraint satisfaction problems

S. Nagarajan[1], S. Goodwin[1], A. Sattar[2], and J. Thornton[2]

[1] Department of Computer Science, University of Regina,
Regina, Saskatchewan, Canada
[shiv, goodwin]@cs.uregina.ca

[2] School of Information Technology,
Griffith University, Gold Coast, Queensland, Australia
j.thornton@gu.edu.au, sattar@cit.gu.edu.au

Abstract. In [Walsh and Stergiou, 1999] enforcing arc consistency (AC) in the dual encoding was shown to strictly dominate enforcing AC on the hidden or GAC on the original problem. We introduce a dual encoding that requires only a small subset of the original constraints to be stored in extension, while the remaining constraints can be stored intensionally. In this paper we present a theoretical comparison between the pruning achieved by enforcing AC on this dual encoding, versus enforcing GAC and dual arc consistency on the standard encoding. We show how the covering based encoding retains the dominance over enforcing GAC on the original problem, while using less space than the existing dual encoding.

1 Introduction

In this paper we present a new dual encoding that is based on the construction of constraint coverings from the original CSP. We show how this covering based dual encoding can be used to address the space complexity issue of the dual encodings, while still retaining the soundness and completeness of the solution procedures. A new form of local consistency based on this dual encoding called *covering arc consistency* (CAC), is defined. The amount of pruning achieved by enforcing CAC on the new encoding is compared theoretically to GAC on the original problem, AC on the hidden encoding, and AC on the standard dual encoding.

2 Preliminary definitions

Given below are a few definitions. Let $\langle V, D, C\rangle$ be a *CSP* where V is the set of variables, D are their domains, and C is the set of constraints. Furthermore, we can assume that each constraint $C_i = \langle V_i, S_i\rangle \in C$ consists of a list of variables $V_i = (v_{i1}, \ldots, v_{ik}) \subseteq V$ and a predicate on these variables, $S_i \subseteq D_{v_{i1}} \times \cdots \times D_{v_{ik}}$. A **binary** *CSP* is one in which all the constraints are defined over pairs of variables. Associated with every binary *CSP* is a constraint graph with a node for every variable and an edge between two nodes if their variables share a constraint [2, 3].

Definition 1. *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}. *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* $V_i \cap V_j$.

Definition 2. *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* S_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. *Let* C_i, C_j *be two constraints* $\in C$. *The* **join** *of* C_i, C_j*, denoted by* $C_i \bowtie C_j = C_{ij}$*, is the set* $\{t \mid t \in S_{ij} \wedge (t[V_i] \in S_i) \wedge (t[V_j] \in S_j)\}$.

Definition 3. *Consider a tuple* t_i *as a consistent instantiation of variables in* V_{t_i}. *An* **extension** *of* t_i *to variables in* $V_{t_i} \cup V_{t_j}$ *is a tuple* t_{ij} *where* t_{ij} *is an instantiation to variables in* $V_{t_i} \cup V_{t_j}$. *The two tuples* t_i *and* t_j *are* **compatible** *if* $t_i[V_{t_i} \cap V_{t_j}]{=}t_j[V_{t_i} \cap V_{t_j}]$*, i.e., the two tuples agree on values for all common variables.*[1] *The tuple* $t_{ij} = t_i \bowtie t_j$ *is a* **consistent extension** *of* t_i **iff** t_i *and* t_j *are* **compatible** *and* $\forall C_i$ *such that* $V_i \subseteq V_{t_{ij}}, t_{ij}[V_i] \in S_i$.

Definition 4. *Given a constraint* C_{ij}*, the value* b *in* D_j*, is called a* **support** *for value* a *in* D_j*, if the pair* $(a, b) \in S_{ij}$. *A value* a *for a variable* i *is* **viable** *iff for every variable* j *such that a constraint* C_{ij} *exists,* a *has a support in* D_j. *The domain* D *of a constraint network, is* **arc consistent** *if for every variable* i *in the network, all the values in* d_i *are viable.*

Definition 5. *[5] A tuple* t *on* $(v_{i_1}, \ldots, v_{i_q})$ *is valid iff* $t \in D(v_{i_1})X \ldots XD(v_{i_q})$. *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* $t[v_i] = val_i$. *A CSP is said to be* **pair-wise consistent***, iff* $\forall C_i, C_j, S_i[C_i \cup C_j]{=}S_j[C_i \cup C_j]$ *and* $\forall S_i, S_i \neq \emptyset$ *[4].*

The notion of arc consistency can be defined for tuples and dual variables in the dual encoding as follows.

Definition 6. *Given two constraints* C_i *and* C_j*, the tuple* $t_j \in S_j$ *is called a* **support** *for tuple* $t_i \in S_i$*, if* $t_i[V_i \cap V_j]{=}\ t_j[V_i \cap V_j]$. *A tuple* t_i *in a constraint* C_i *is* **viable** *iff for every constraint* C_j*, tuple* t_i *has support in* C_j. *A constraint network is* **dual arc consistent***, if for every constraint* C_i*, all the tuples in* S_i *are viable.*

Definition 7. *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. *If* C_{cover} *is a minimal constraint cover,* $|C_{cover}| \leq |V|$.

[1] If $V_{t_i} \cap V_{t_j} = \emptyset$, t_i and t_j are automatically compatible.

3 Covering Arc Consistency

Consider a dual encoding of a CSP, where the nodes of the dual encoding are the constraints in a constraint cover of the given CSP. We now define a new form of arc consistency based on this dual encoding known as *Covering Arc Consistency*, which is defined on constraints in a cover.

Definition 8. *Let* $C_{cover} = \{C_1, C_2, \ldots, C_m\}$*. Given two constraints* $C_i \in C_{cover}$ *and* $C_j \in C_{cover}$*, the tuple* $t_j \in S_j$ *is called a* **covering arc support** *for tuple* $t_i \in S_i$*, 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 covering arc support in* C_j*. A constraint network is* **covering arc consistent (CAC)** *w.r.t a covering* C_{cover}*, if* $\forall C_i \in C_{cover}$*, all the tuples in* S_i *are viable.*

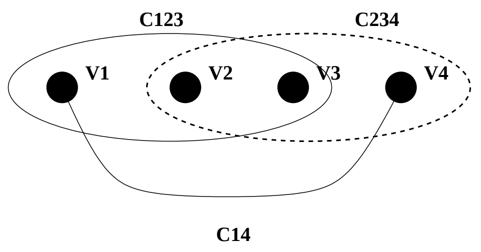
An arc consistency algorithm removes all arc inconsistent values from the domains of the variables of the encoding. The following theorems are proven in [6].

Theorem 1. *Achieving AC on the hidden variable encoding is equivalent to achieving GAC on the variables in the original problem.*

Theorem 2. *Achieving AC on the dual encoding is strictly stronger than achieving GAC on the original problem.*

Theorem 3. *Achieving AC on the dual encoding is strictly stronger than achieving AC on the hidden variable encoding.*

In the following we perform a similar theoretical comparison between enforcing covering arc consistency (CAC), GAC and dual arc consistency. Consider the following example taken from [1]. This CSP is already GAC, while enforcing Dual AC removes some values from the domains. Enforcing CAC, reduces the domains to singleton domains.



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

Fig. 1. Non-Binary CSP: An example

Theorem 4. *Achieving CAC on the constraint covering based dual encoding is strictly stronger than achieving GAC on the original problem.*

Proof. If GAC on the original encoding removes a value val_i from the domain of variable v_i then there exists some constraint C_i that mentions variable v_i, and the assignment of val_i to v_i cannot be extended to a consistent assignment to the rest of the variables in C_i. Consider a covering based dual encoding. Either the cover contains the previously mentioned constraint C_i, or $C_i \in \{C\text{-}C_{cover}\}$. If $C_i \in C_{cover}$, then we can derive the *nogood* that removes val_i from v_i (since no tuple in C_i assigns val_i to v_i). Otherwise, if $C_i \notin C_{cover}$, then there is some other constraint $C_j \in C_{cover}$ that mentions v_i (since C_{cover} must cover all variables). If C_j contains no tuple that assigns val_i to v_i, then we can derive the same *nogood*. If C_j contains some tuples that assign val_i to v_i, then these tuples will all be discarded when a consistent extension is verified against the constraint projection of C_i (since $C_i \in C\text{-}C_{cover}$). Hence we can derive the *nogood* that val_i cannot be assigned to v_i. To show strictness we can consider the example previously given in Figure 1. □

Theorem 5. *Achieving CAC on the constraint covering based dual encoding is strictly stronger than achieving AC on the hidden variable encoding.*

Proof. From Theorem 1, enforcing GAC on the original problem and enforcing AC on the hidden encoding are equivalent. Using this and Theorem 4, if follows that enforcing CAC on the covering based dual encoding is strictly stronger than enforcing AC on the hidden variable encoding. □

Theorem 6. *Achieving CAC on the constraint covering based dual encoding is is incomparable to achieving AC on the standard dual encoding.*

Proof. To show that enforcing CAC 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 CAC 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 CAC on the covering based dual encoding. To show a) we can consider the example in Figure 1. For b) consider a CSP with 6 variables with binary domains. The 6 constraints are
$C_{a,b}=\{(0,0)\}$, $C_{b,c}=\{(0,0)\}$, $C_{d,e}=\{(0,1)\}$, $C_{e,f}=\{(1,0)\}$
$C_{a,b,c,d,e}=\{(0,0,0,0,1)\}$, $C_{a,b,c,d,f}=\{(0,0,1,0,0),(0,0,0,1,0),(1,0,0,0,0)\}$
Consider a constraint covering $C_{cover}=\{C_{a,b}, C_{b,c}, C_{d,e}, C_{e,f}\}$. This CSP is covering 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. □

Theorem 7. *When $|C_{cover}|=|C|$, CAC on the covering based encoding prunes at least as much as AC on the dual encoding.*

Proof. When $C_{cover}=C$, CAC on the covering based encoding enforces pair-wise consistency between all the constraints in the cover, and then verifies that the relational join of all pairs of consistent constraints, satisfy all the other constraints, on all common variables, by projection. Hence the pruning achieved is at least as much as pair-wise consistency or dual AC. □

Theorem 8. *If $\forall C_i \notin C_{cover}$, $\exists C_p, C_q \in C_{cover}$ such that $V_i \subseteq (V_p \cup V_q)$, pair-wise consistency prunes at most as much as CAC on the covering based dual encoding w.r.t C_{cover}.*

Proof. Pair-wise consistency find inconsistencies between pairs of constraints. CAC performs pair-wise consistency on all the pairs of constraints in the covering. Given a constraint covering C_{cover}, if $\forall C_i \notin C_{cover}$, $\exists C_p, C_q \in C_{cover}$ such that $V_i \subseteq (V_p \cup V_q)$, the algorithm enforcing CAC 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 CAC on the covering based dual encoding w.r.t a given C_{cover}. □

Theorem 9. *If $\forall C_i, C_j \notin C_{cover}$, such that $\exists C_p, C_q \in C_{cover}$, $(V_i \cap V_j) \subseteq (V_p \cup V_q)$, CAC on the covering based dual encoding w.r.t C_{cover} prunes at least as much as pair-wise consistency.*

Proof. From the example in part b) of Theorem 6 it is clear that there exists a problem that satisfies this condition where pruning using AC on the dual encoding prunes more than CAC on the covering based dual encoding w.r.t a given C_{cover}. To show that is *precisely* the condition when dual AC prunes more than CAC w.r.t a C_{cover}, can be done as follows. 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 CAC .w.r.t C_{cover} derives all pair-wise inconsistent tuples that AC on the dual encoding would derive. Hence this is precisely the condition when CAC w.r.t a cover is no worse than dual AC. □

4 Conclusions

This paper presents a new dual encodings for CSPs based on constraint coverings. We introduce a new form local consistency that is defined on this dual encoding called covering arc consistency (CAC). It is shown that enforcing CAC dominates GAC and hidden variable AC, and is incomparable to standard Dual AC. We also present a precise characterisation of conditions under which the pruning achieved by CAC is comparable to the pruning achieved by dual AC.

References

1. Bessiere C. Non-binary constraints. In *Principles and Practice of Constraint Programming, CP-99, Invited Lecture*, 1999.
2. R. Dechter. Constraint networks. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 276–285. Wiley, 1992. Volume 1, second edition.
3. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.
4. Janssen P, Jegou P, Nouguier B., and Vilarem M.C. A filtering process for general constraint satisfaction problems: achieving pair-wise consistency using an associated binary representation. In *Proceedings of the IEEE Workshop on Tools for Artificial Intelligence*, pages 420–427, Fairfax, USA, 1989.
5. Mohr R. and Masini G. Good old discrete relaxation. In *Proceedings ECAI'88*, pages 651–656, 1988.
6. Kostas Stergiou and Toby Walsh. Encodings of non-binary constraint satisfaction problems. In *Proceedings of the 16th National Conference on Artificial Intelligence*, pages 163–168, 1999.