Non binary constraint satisfaction: From the dual to the primal

S. Nagarajan*
QNX Software Systems,
Kanata, Ontario,
Canada, K2M 1W8
shiv@nagarajan.net

S. D. Goodwin
Dept. of Computer Science,
University of Regina, Regina,
Saskatchewan, Canada, S4S 0A2
goodwin@cs.uregina.ca

A. Sattar
School of Information Technology
Griffith University, Gold Coast Campus
Queensland, Australia, 4111
sattar@cit.gu.edu.au

Abstract

Non binary constraints have recently been studied quite extensively since they represent real life problems very naturally. Specifically, extensions to binary arc consistency into generalised arc consistency (GAC), and forward checking that incorporates a limited amount of GAC have been proposed, to handle non-binary constraints directly. Enforcing arc consistency on the dual encoding has been shown to strictly dominate enforcing GAC on the primal encoding. More recently, modifications 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 results that combine the enforcement of arc consistency in these covering based dual encodings, with performing forward checking based search in the primal encoding. We demonstrate how this new scheme can be shown to strictly dominate standard non-binary forward checking, while being able to efficiently enforce extremely high levels of consistency.

Introduction

Many industrial applications can be easily represented as CSPs and solved using algorithms that have been designed for general constraint satisfaction. Constraint satisfaction problems have been successfully deployed in design, diagnosis, scheduling, planning, timetabling, temporal reasoning and other optimisation problems.

While being quite prevalent in terms of real life problems, constraint satisfaction has also been shown to be very hard in general. Recently, it has become clear that non-binary resolution techniques are necessary, and 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. This paper presents a method of combining the two approaches.

*Work done while at the University of Regina.

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. It has been seen that 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. On the other hand it is well known that methods that enforce high levels of consistency during search can often solve large classes of problems efficiently and scale well (Forward checking (FC)/Maintaining arc consistency (MAC)). More recently (Nagarajan *et al.* 2000), modifications to the standard dual encoding have been proposed, using constraint coverings, that can compactly represent the given CSP using an equivalent dual encoding that contains all the original solutions to the CSP. 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 combine the covering encoding based techniques with the non-binary forward checking algorithms based on the primal encodings, to provide an efficient way to solve general CSPs. This technique allows us to obtain the benefit of the high level of consistency enforced in the covering based dual encodings, while retaining the implementational and computational efficiency of search in the primal encoding. Many of the successful search based techniques using the primal encoding for arbitrary constraint satisfaction problems, owe their success to the various branching heuristics that are available (e.g. unit propagation, minimum remaining values). Many algorithms using the dual encoding have limited branching capabilities because of their structure (Walsh 2000). This paper attempts to provide a platform to overcome this limitation of dual encodings, while still inheriting the advantages of the primal algorithms.

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 $t_{i,a} \in S_i$ is 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}. An encoding of a CSP using a primal constraint graph is called a* **primal encoding.**

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 $V_i \cap V_j$. An encoding of a CSP using the dual constraint graph is called a* **dual encoding.**

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 $t_{x,a}$ 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 $V_i \subseteq V_x$, $t_{x,a}[V_i] \in S_i$. 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.

Non binary constraint satisfaction

Arc consistency is a form of consistency defined for binary constraints defined using the notion of support and viability (Bessière & Cordier 1993).

Definition 7 *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.*

Intuitively, arc consistency for binary CSPs checks the consistency of labels for each pair of nodes linked by a binary constraint and removes the labels that cannot satisfy this local condition. Arc consistency is based on the notion of support. If we are considering value $a \in D_i$ for variable i, as long as value a has a minimum of support from the labels at each of the other nodes (variables) j (j not equal to i), a is considered a viable label for i. But once there exists a node at which no remaining label satisfies the required relation with a, the a can be eliminated as a possible label for i.

The definition of AC as described earlier is not directly applicable 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 8 *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 $t[v_i] = val_i$.*

This definition is valid for both binary and non-binary constraints. A value $a \in D_i$, is consistent with a constraint C_m, **iff** either $v_i \notin V_m$ or $\exists$ a tuple $t \in S_m$, such that $t[v_i] = a$. A constraint C_m is generalised arc consistent **iff** $\forall v_i \in V_m$, $D_i \neq \emptyset$, and $\forall a \in D_i$, a is consistent with C_m.

Given the success of FC and MAC at solving a large number of varied CSPs, the generalisations of FC for non-binary constraints were studied in (Bessière *et al.* 1999). While generalising the binary FC algorithm, several generalisations are possible while the MAC algorithm allows only one generalisation. Binary forward checking guarantees at each node that there is no future node with values in its domain that is inconsistent with some past node. The set of constraints involving past and future variables is used to enforce AC at each level in the algorithm. When all constraints are binary, there is only one option for that set of constraints, i.e., constraints involving one past variable with one future variable. But when constraints are non-binary there are many alternatives, since one has to deal with partially instantiated constraints.

All these generalisations collapse to the standard version in the case where all constraints are binary. The results in that study can be summarized as follows. NFC5 > NFC3 > NFC2, NFC5 > NFC4 > NFC2, NFC2 > NFC1 > NFC0, NFC3 $\sim$ NFC4. The NFC5 algorithm, after assigning the current variable, makes the set of constraints involving at least one past variable and at least one future variable GAC. An algorithm FC+ that was presented in (Bacchus & van Beek 1998) to operate on the hidden variable encoding, has also been shown to be equivalent to NFC1.

Given a dual encoding of a non-binary CSP, one can define arc consistency in terms of the constraints in the

CSP representing dual variables and the tuples in the various constraints, representing values to each of these dual variables. 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 9 *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. 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 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.

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

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

Theorem 3 *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 4 *Enforcing GDAC on the dual encoding is strictly stronger than enforcing GAC on the original problem.*

Theorem 5 *Enforcing GDAC on the dual encoding is strictly stronger than enforcing pair-wise consistency on the dual encoding.*

Theorem 6 *Enforcing GDAC on the dual encoding is strictly stronger than enforcing AC on the hidden variable encoding.*

Covering based dual encodings

Intuitively, a tuple, $t_{i,a}$, is consistent if it satisfies all the constraints whose variables are completely instantiated by $t_{i,a}$. 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.

[1]GDAC was called covering arc consistency.

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 $t_{j,b} \in S_j$ is called a* **generalised dual arc support** *for tuple $t_{i,a} \in S_i$ w.r.t. C_{cover}, if $t_{i,a}[V_i \cap V_j] = t_{j,b}[V_i \cap V_j]$ and $\forall C_x \in \{C\text{-}\{C_i, C_j\}\}$, $(t_{i,a} \bowtie t_{j,b})[V_{ij} \cap V_x] \in S_x[V_{ij} \cap V_x]$. A tuple $t_{i,a} \in C_i \in C_{cover}$ is* **viable iff** *for every constraint $C_j \in C_{cover}$, tuple $t_{i,a}$ 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.*

Theorem 7 *Achieving GDAC on the constraint covering based dual encoding is strictly stronger than achieving GAC on the original problem.*

Proof Assume that enforcing GAC on the original encoding removes a value val_i from the domain of variable v_i. This implies that 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. In the covering based dual encoding we construct a constraint cover. 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 *no-good* 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 *no-good*. 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 *no-good* that val_i cannot be assigned to v_i. To show strictness we can consider the example given in Figure 1. Enforcing GDAC on the covering based dual encoding, can derive more *no-goods* than enforcing GAC on the original problem. □

In fact it is possible to show that enforcing pair wise consistency on the standard dual encoding is incomparable with enforcing GDAC w.r.t an arbitrary constraint cover. But for special covers it is easy to show how GDAC can still enforce high levels of consistency.

Definition 13 *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.*

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. □

Example

Consider a simple example with 3 constraints, and 4 variables given in figure 1. Enforcing GAC on this non-binary CSP will remove no values since it is already GAC.

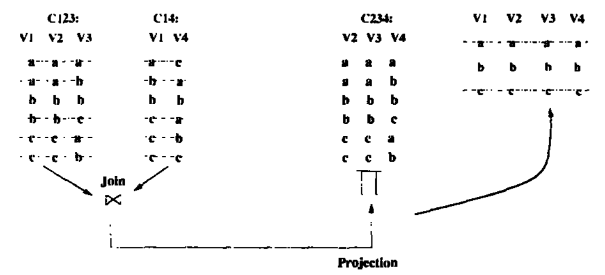

Figure 1: GDAC w.r.t. a cover

Consider the same example 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 GDAC 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 the rest of the constraints in C. The only dual domains that are pruned of values are the constraints in the covering. In figure 1, a constraint cover is constructed as $C_{cover} = \{C_{123}, C_{14}\}$. The pruning achieved by enforcing GDAC 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, and we can now solve the problem in a backtrack free manner. Although we enforce GDAC only on the constraints in the covering, and prune only the domains of the constraints in the cover, we are able to enforce a level consistency that is higher than enforcing dual arc consistency.

From the dual to the primal

As seen in the example above and from the theoretical results given earlier, effective pruning can be achieved by enforcing high levels of consistency in the dual encoding. In fact by using the covering based encoding we can enforce a level of consistency that is strictly higher than enforcing GAC, and that is in many cases higher than PWC too while doing much less work. Given this insight it is interesting to consider using this pruning to effectively remove values that cannot participate in any solution, before search, and then proceed to search for a solution in the modified domains of the variables using standard non-binary forward checking.

Results

In this section we present some empirical results to support the ideas presented in this paper. We generated 500 random CSPs using a random non-binary CSP generator that we built. The CSPs were generated with 25 variables, 60 constraints, arity 4 and domain size 3. We generated problems at 5 different tightness values, between 0.3 and 0.7 in steps of 0.1. For each tightness value, we generated 100 problems and reported the results averaged over these 100 problems. We report 3 different tables that contain the number of nodes visited, number of constraint checks performed and the percentage of values removed by arc consistency pre-processing. We implemented NFC1 and also weighted constraint covering algorithms, and enforced GDAC on them. We also report the results by translating the no-goods from the dual domains after pre-processing to the original domains, and then executing NFC1 using the resulting variable domains. As seen in the tables, the covering based algorithms perform very well when compared to standard NFC1. Also when we consider the algorithms using the translated no-goods, the pruning of the dual algorithms allows us to reduce the search space visited by the NFC1 algorithm even further.

P_t	*FC*	*GDACw*	*GDAC-FC*	*GDACw-FC*
0.30	5127.6	78.3	5127.6	5127.6
0.40	35791.3	1346	1698.3	18651.1
0.50	3150.1	77.1	0	65.5
0.60	0	0	0	0
0.70	11	0	0	0

Table 1: Nodes Visited with PP:$\langle 25, 3, 4, 60 \rangle$

P_t	FC	GDACw	GDAC-FC	GDACw-FC
0.30	424888	156626	3.52039e+06	517087
0.40	4.40499e+06	1.54035e+06	2.24581e+06	2.25971e+06
0.50	489456	74249.5	145103	46137.8
0.60	415.4	636.8	2321.4	636.8
0.70	2165	194	687	194

Table 2: CCks with PP:$\langle 25, 3, 4, 60 \rangle$

P_t	FC	GDACw	GDAC-FC	GDACw-FC
0.30	0	1.75505	0	0
0.40	5.8	21.9237	26.2074	10.1
0.50	10.6	61.0904	16.7215	49.5
0.60	20.5	6.25	2.08791	6.25
0.70	32	6.69643	1.67984	6.69643

Table 3: % Vals Removed with PP:$\langle 25, 3, 4, 60 \rangle$

Conclusions

In this paper we have presented some ideas to combine local consistency enforcement in the dual encoding for non-binary CSPs with search using non-binary FC in the primal encoding. We enforce high levels of consistency in the pre-processing phase, and prune many inconsistent values. It still remains to be seen whether the algorithms can be combined in a tighter fashion so that the high level dual encoding based consistencies can be maintained while searching in the primal graph.

Acknowledgements

This research was supported by National Science and Engineering Research Council (NSERC), Canada and Australian Research Council (ARC) Large grant scheme.

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.

Bessière, C., and Cordier, M. O. 1993. Arc-consistency and Arc-consistency again. In *Proceedings of the National Conference on Articial Intelligence, AAAI-93*, 108–113. American Association for Artificial Intelligence.

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.

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.

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.

Walsh, T. 2000. Sat v csp. In *Principles and Practice of Constraint Programming, CP-2000*, volume 1894, 441–456. Springer Verlag.