From: AAAI-97 Proceedings. Copyright © 1997, AAAI (www.aaai.org). All rights reserved.

Detecting Unsatisfiable CSPs by Coloring the Micro-Structure

Daya Ram Gaur and **W. Ken Jackson** and **William S. Havens**
Intelligent Systems Lab, School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6
{gaur,jackson,havens}@cs.sfu.ca

Abstract

Constraint satisfaction research has focussed on consistency checking using k-consistency and its variations such as arc-consistency, and path-consistency. We define a new form of consistency checking that is based on coloring the micro-structure graph of a constraint satisfaction problem (CSP). In our formulation, if the micro-structure graph of a CSP with n variables can be colored with $n-1$ colors then the problem is unsatisfiable. This new notion of consistency-by-coloring is compared to arc-consistency. We provide examples that show that neither arc-consistency nor consistency-by-coloring is more powerful than the other in a theoretical sense. We also describe the results of preliminary computational experiments that compare consistency-by-coloring and arc-consistency.

Introduction

Constraint satisfaction problems (CSPs) are often solved by backtracking algorithms that interleave search and consistency checking. Forward checking (Haralick & Elliot 1980) and maintained arc-consistency (Sabin & Freuder 1994) are examples of such algorithms. A key aspect to the success of these algorithms is that they can detect when a subproblem or sub-CSP is unsatisfiable and hence, prune a branch in the search tree earlier than when no consistency checking is done.

Work on consistency checking has focussed on variations of arc-consistency and k-consistency. These methods might detect that a CSP is unsatisfiable but they are not guaranteed to do so. We define a new form of consistency checking that is based on coloring the micro-structure graph of a CSP. The micro-structure graph for a CSP (Jegou 1993) has a vertex for every variable and value pair in the original CSP and connects vertices that correspond to compatible assignments in the CSP. In our new consistency checking scheme, if a CSP with n variables has a micro-structure graph whose vertices can be colored with $n-1$ colors then the CSP is unsatisfiable. A more precise description of the above ideas is given later in the paper.

Copyright ©1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Consistency-by-coloring is a new form of consistency checking in that it is unrelated to k-consistency. We describe an arc-consistent yet unsatisfiable CSP that is shown to be unsatisfiable by coloring the micro-structure, and vice versa.

We also describe the results of a preliminary computational experiment that compares arc-consistency with consistency-by-coloring. The results indicate that arc-consistency and consistency-by-coloring are similar in their ability to detect unsatisfiability.

Preliminaries

A *constraint satisfaction problem* (CSP) consists of a set of n variables, each associated with a finite domain of possible values, and a set of constraints involving subsets of the variables. The problem is to find a value from the domain of each variable such that all the constraints between the variables are satisfied. For this paper, we assume that all the constraints are binary.

The *micro-structure graph* for a CSP is an undirected graph G constructed as follows:

1. For every variable x and every domain value d for x, there is a vertex x_d in G.
2. Two nodes x_a and y_b in G are connected by an edge iff the pair of values (a, b) satisfies the constraint between the variables x and y.

For the purposes of the above definition, we assume that there is exactly one constraint between every pair of variables; a pair of variables with no explicit constraint is considered to be constrained by the universal relation (that is, all pairs of values are considered to be compatible).

Figure 1 shows an example of a CSP and its micro-structure graph. The micro-structure graph for a CSP with n variables is an n-partite graph with one partition for each variable.

We use a few basic definitions from graph theory. A *clique* in a graph is a subset of the vertices of the graph such that there is an edge connecting each pair of distinct vertices in the clique. A (vertex) *coloring* of a graph G is an assignment of colors to the vertices of G such that no two adjacent vertices get the same color.

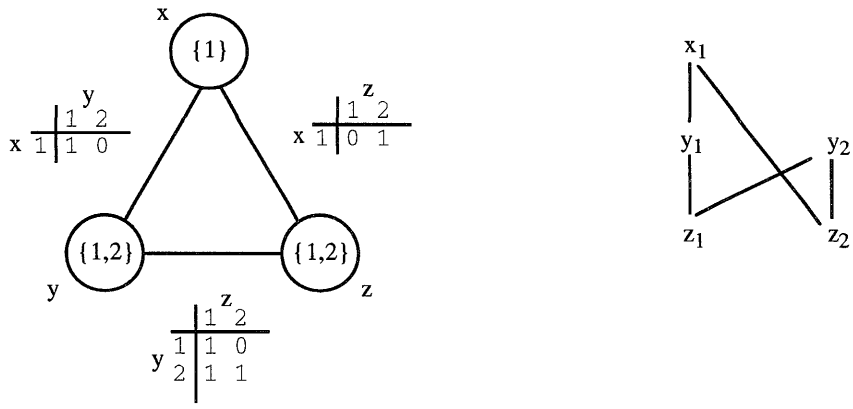

Figure 1: An Unsatisfiable CSP Detectable by Arc-consistency

Coloring the Micro-Structure

This section describes consistency-by-coloring more precisely. A solution to a CSP must assign a value to each variable that is compatible with the values assigned to all the other variables. The following lemma holds because in the micro-structure graph compatible values are connected by edges.

Lemma 1 A CSP with n variables is satisfiable iff its micro-structure graph contains a clique of size n.

Determining whether a graph contains a clique of size n is NP-complete (Garey & Johnson 1979). However, in some cases, it can be easily shown that a graph does *not* contain a clique of size n. For example, if a graph can be colored with 3 colors then the graph cannot contain a clique of size 4 because coloring the clique would require at least 4 colors. The following lemma generalizes the above idea.

Lemma 2 If the vertices of a graph G can be colored with $k-1$ colors then G cannot contain a clique of size k.

Checking for consistency by coloring the micro-structure follows directly from the above lemmas.

Corollary 1 If the micro-structure graph of a CSP with n variables can be colored with $n-1$ colors then the CSP is unsatisfiable.

We can attempt to determine if a CSP with n variables is unsatisfiable by constructing its micro-structure graph and then attempting to color the graph with $n-1$ colors. If the coloring attempt succeeds then we know the original CSP is unsatisfiable. If the coloring attempt fails, then we don't know whether the original CSP is satisfiable or not. This uncertainty also occurs when using arc-consistency in that arc-consistency can detect some but not all cases of unsatisfiability.

Trying to color the micro-structure graph may seem to be prohibitive because graph coloring is NP-complete. However, there are simple polynomial time algorithms for graph coloring that can color many but not all k-colorable graphs. The fact that an algorithm may fail to color some k-colorable graphs is not a real problem; we just remain undecided whether the CSP is satisfiable or not.

Relation to k-consistency

There does not seem to be an obvious relationship between consistency based on coloring the micro-structure and the more traditional notions of arc-consistency and k-consistency (Mackworth 1977; Freuder 1978). In this section, we describe two examples that show that neither consistency-by-coloring nor k-consistency is more powerful than the other. That is, there are unsatisfiable CSPs that are found to be unsatisfiable using one method but not the other method.

Two variables x and y in a CSP are considered *arc-consistent* iff for all values in the domain of x, there is a compatible value in the domain of y and vice versa. The compatible value in the domain of y is said "to support" a value in x's domain. A CSP is arc-consistent if every pair of variables is arc-consistent. A CSP can be made arc-consistent by removing values "with no support" from the domains of the variables. If any variable's domain becomes empty then the original CSP is known to be unsatisfiable. There has been considerable progress in developing efficient algorithms for enforcing arc-consistency (Bessière & Régin 1994; Bessière 1994; Freuder 1994; Schiex, Régin, & Verfaille 1996; Van Hentenryck, Deville, & Teng 1992).

The CSP shown in Figure 1 exemplifies the case where arc-consistency detects unsatisfiability not detected by coloring the micro-structure. The figure shows the original CSP on the left and its micro-structure graph on the right. The CSP has a pair of variables x and z that is not arc-consistent. Furthermore, enforcing arc-consistency causes the domain of a variable to become empty. For example, the following values can be eliminated to enforce arc-consistency.

1. The value 1 can be eliminated from z's domain because of the constraint with the variable x.
2. The value 2 can be eliminated from y's domain because of the constraint with x.
3. The value 2 can be eliminated from z's domain because of the constraint with y.

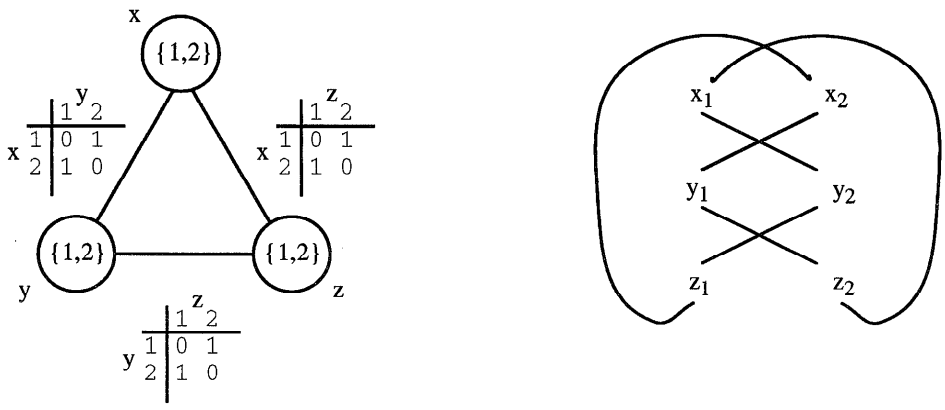

Figure 2: An Unsatisfiable CSP Detectable by Coloring the Micro-Structure

At this point, z's domain is empty and the CSP is known to be unsatisfiable.

The micro-structure graph of the CSP in Figure 1 contains the following cycle of length 5.

$$x_1, y_1, z_1, y_2, z_2, x_1$$

At least 3 colors are needed to color the micro-structure graph because of the presence of this odd-length cycle. Therefore, the CSP cannot be detected as unsatisfiable by coloring the micro-structure. Our experimental results, described later, suggest, however, that most random CSPs in which arc-consistency detects unsatisfiability are also found to be unsatisfiable by coloring the micro-structure.

The CSP shown in Figure 2 shows a case where coloring the micro-structure can detect unsatisfiability not detected by arc-consistency. Each of the constraints in the CSP is arc-consistent. The micro-structure graph can be easily colored with two colors by coloring the vertices x_1, y_1, z_1 with one color and the vertices x_2, y_2, z_2 with the other color. Therefore, coloring the micro-structure detects unsatisfiability in some cases where arc-consistency does not.

The above example generalizes to k-consistency by considering a CSP with n variables, each with the domain $\{1, 2, \ldots, n-1\}$, such that each variable is connected to every other variable by a dis-equality constraint. That is, the CSP is a clique of binary dis-equality constraints. A CSP is k-consistent iff for any consistent instantiation of any $k-1$ variables there is an instantiation of every kth variable such that the k values taken together satisfy all the constraints involving the k variables. Arc-consistency and path-consistency are equivalent to 2-consistency and 3-consistency, respectively. A CSP that is a clique of binary dis-equality constraints is $(n-1)$-consistent. However, its micro-structure can be colored with $n-1$ colors by using a different color for each "column" in the micro-structure graph. Therefore, there are cases for which coloring the micro-structure is more powerful than k-consistency, for $1 \leq k \leq n-1$.

Note that a CSP that is a clique of dis-equality constraints can be represented using a single n-ary constraint of difference. Régin (Régin 1994) has shown that n-ary arc-consistency can detect unsatisfiability for such constraints and there are efficient algorithms for CSPs of this form. We are not certain of the relation between n-ary arc-consistency and consistency by coloring the micro-structure. However, Régin's results are restricted to CSPs involving constraints of difference whereas coloring the micro-structure is applicable to any CSP.

Experimental Results

This section describes the results of some preliminary computational experiments that compare arc-consistency with consistency-by-coloring. We generated a set of unsatisfiable problems and determined which problems were detected as unsatisfiable by arc-consistency and which problems were detected as unsatisfiable by coloring the micro-structure graph.

We generated a set of random CSPs, following (Smith 1994), described by the following 4 parameters:

1. The number of variables, n.
2. The number of values in each variable's domain, m.
3. The probability p_1 that a constraint exists between between two variables. The instances are generated so that each instance has exactly $p_1 n(n-1)/2$ constraints.
4. The conditional probability p_2 that a pair of values is not compatible given that there is constraint between the variables. Constraints are generated so that the number of incompatible pairs is exactly $p_2 m^2$.

In our experiment we used the parameters $n = 10$, $m = 5$, p_1 was varied between 0.1 and 1.0 in steps of 0.1, and p_2 was varied between 0.10 and 0.89 in increments of 0.01. At each combination of values for $p1$ and $p2$ we generated 100 problems. In total, 80000 problems were generated and 15,773 of these problems were unsatisfiable.

We used the well-known Brélaz graph coloring algorithm (Brélaz 1979). The Brélaz algorithm is a heuristic algorithm that is not guaranteed to find a coloring if one exists. However, its time complexity is quite low, $O(|V|^2)$

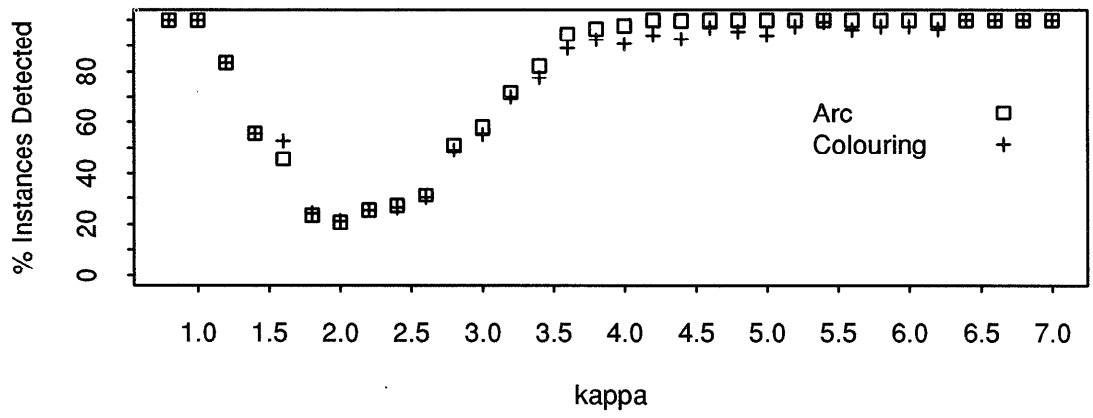

Figure 3: Percentage of Unsatisfiable CSPs Detected as Unsatisfiable

where $|V|$ is the number of vertices. The number of problems detected to be unsatisfiable by coloring the micro-structure would likely increase if a complete coloring algorithm were used.

Enforcing arc-consistency detected unsatisfiability in 9347 problems while coloring the micro-structure detected unsatisfiability in 9005 problems. We also measured, on a problem-by-problem basis, whether arc-consistency or consistency-by-coloring was able to detect an unsatisfiable problem when the other method did not: Arc-consistency bettered consistency-by-coloring on 532 problems while graph-coloring bettered arc-consistency on 190 problems. These results experimentally verify that consistency-by-coloring is different from arc-consistency.

The graph in Figure 3 shows the percentage of unsatisfiable problems that were detected unsatisfiable versus an estimate of the "constrainedness" of the problem (Ghent *et al.* 1996). The constrainedness, κ, of a random binary CSP is estimated as follows:

$$\kappa = -\sum_{c \in C} log(1 - p_2)) / \sum_{v \in V} log(m_v)$$

where C is the set of constraints, V is the set of variables, and m_v is the size of the domain of the variable v. A large κ indicates an over-constrained problem, while a small κ indicates an under-constraint problem. Since all the problems we examined are unsatisfiable, they are all over-constrained though their estimated constrainedness may be low.

The graph reveals two interesting phenomena: First, The shape of the curve is reminiscent of an inverse phase transition. The effectiveness of both arc-consistency and consistency-by-coloring diminishes quite sharply as the estimated constrainedness approaches a critical value and then the effectiveness rises again. Second, it is clear that arc-consistency is better over a wide range of problems. However, consistency-by-coloring matches and betters the effectiveness of arc-consistency when the estimated constrainedness is low. This behavior is especially interesting because unsatisfiable problems whose estimated constrainedness is low can be very difficult to solve (Smith 1994). The difference between the two techniques can be seen more sharply in the graph in Figure 4 that shows the percentage of problems where one technique detected an unsatisfiable problem that was not detected by the other technique. This graph further illustrates that consistency-by-coloring seems to be more effective on problems where the estimated constrainedness is low. For problems with $\kappa \leq 2.0$, consistency-by-coloring meets or out-performs arc-consistency.

Obviously, the computation time taken by both techniques is also of interest. We did not conducted experiments to measure the computation time because our code for enforcing arc-consistency and our code for Brélaz 's algorithm was not engineered to make measurements of the computation time meaningful. With this work, we have attempted to address some scientific questions before the addressing the engineering questions as suggested by (J.N.Hooker 1995). Investigating the computational performance of consistency-by-coloring is an important area of future work.

Conclusions and Future Work

Arc-consistency and k-consistency have received a great deal of attention in constraint satisfaction research. The main contribution of our work is the identification of an alternative method for detecting when a CSP is unsatisfiable. Our method is based on transforming a CSP into its micro-structure and then attempting to color the micro-structure graph. Finding a coloring implies that the original CSP is

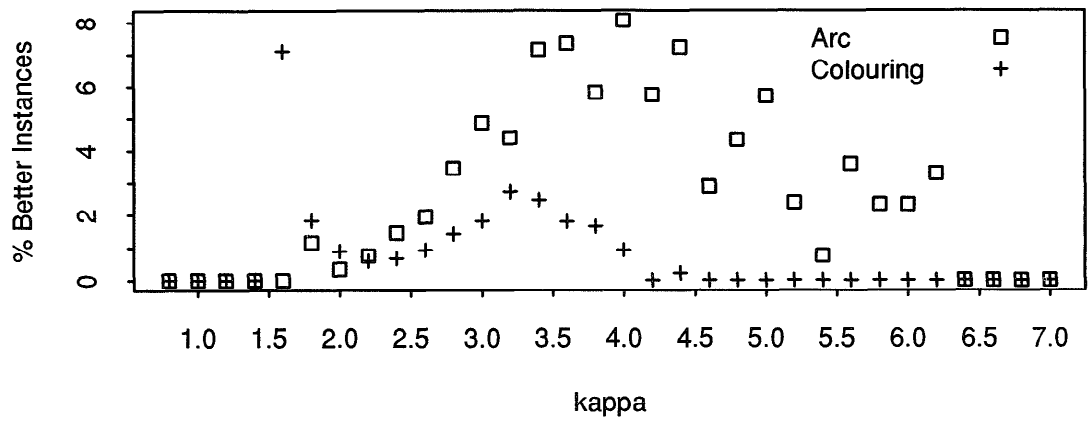

Figure 4: Percentage of CSPs Where One Technique Bettered the Other

unsatisfiable.

There does not seem to be an obvious connection between k-consistency and consistency by coloring the micro-structure. We have shown instances of unsatisfiable CSPs that can be detected as unsatisfiable by coloring the micro-structure but not detected as unsatisfiable by k-consistency, and vice versa. We have also found such instances in randomly generated binary CSPs. Therefore, we feel that consistency by coloring the micro-structure qualifies as a new form of consistency checking.

Initial experiments that compare arc-consistency and consistency by coloring the micro-structure indicate that both techniques have similar performance. Arc-consistency performs better on a wide range of problems but there is some evidence that consistency-by-coloring performs better when the estimated constrainedness is low.

One of our goals is to find new consistency checking techniques that can be effectively embedded within a backtracking search. Consistency-by-coloring is a new consistency checking technique but there are many questions that must be answered before it can be effectively merged with backtracking. As future work, we plan to address the following questions:

- How does consistency-by-coloring compare with n-ary arc-consistency?
- How do arc-consistency and consistency-by-coloring compare on structured problems and problems with non-uniform domain sizes?
- Would a complete graph coloring algorithm significantly affect the effectiveness of consistency-by-coloring?
- Can consistency-by-coloring be used to filter domains in addition to detecting unsatisfiability?
- How do arc-consistency and consistency-by-coloring compare when both are interleaved with a backtracking search as in the maintained arc-consistency algorithm?
- Are there graph coloring algorithms that work better than Brélaz's algorithm for coloring the micro-structure of a CSP?

Acknowledgments

This work was supported by the Canadian Institute for Robotics and Intelligent Systems project IC-6 on Intelligent Scheduling. We would like to thank the members of the Intelligent Systems Lab at SFU for their support and advice.

References

Bessière, C., and Régin, J.-C. 1994. An arc-consistency algorithm optimal in the number of constraint checks. In Meyer, M., ed., *Proceedings ECAI'94 Workshop on Constraint Processing.*

Bessière, C. 1994. Arc-consistency and arc-consistency again. *Artificial Intelligence* 65(1):179–190.

Brélaz, D. 1979. New methods to color the vertices of a graph. *Communications of the ACM* 22(4):251–256.

Freuder, E. C. 1978. Synthesizing constraint expressions. *Communications of the ACM* 21:958–966.

Freuder, E. 1994. Using metalevel constraint knowledge to reduce constraint checking. In Meyer, M., ed., *Proceedings ECAI'94 Workshop on Constraint Processing.*

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman & Co.

Ghent, I. P.; MacIntyre, E.; Prosser, P.; and Walsh, T. 1996. The constrainedness of search. In *AAAI-96: Pro-*

ceedings of the 13th National Conference on Artificial Intelligence, 246–252.

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

Jegou, P. 1993. Decomposition of domains based on the micro-structure of finite constraint satisfaction problems. In *AAAI93: Proceedings of the 11th National Conference on Artificial Intelligence*. Washington, DC: American Association for Artificial Intelligence.

J.N.Hooker. 1995. Testing heuristics: We have it all wrong. *Journal of Heuristics* 1(1).

Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8(1):99–118.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *AAAI-94: Proceedings of the 12th National Conference on Artificial Intelligence*. Seattle, WA: AAAI Press / The MIT Press.

Sabin, D., and Freuder, E. 1994. Contradicting conventional wisdom in constraint satisfaction. In Borning, A., ed., *PPCP'94: Second Workshop on Principles and Practice of Constraint Programming*.

Schiex, T.; Régin, J.-C.; and Verfaille, G. 1996. Lazy arc consistency. In *AAAI-96: Proceedings of the 13th National Conference on Artificial Intelligence*, 216–221.

Smith, B. 1994. The phase transition and the mushy region in constraint satisfaction problems. In *Proceedings ECAI'94*. (Also Leeds University Research Report 93.41).

Van Hentenryck, P.; Deville, Y.; and Teng, C. 1992. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence* 57(2–3):291–321.