An Optimal SAT Encoding of Arc Consistency in Binary CSP's *

Ian P. Gent

School of Computer Science, University of St. Andrews, Fife KY16 9SS, United Kingdom. ipg@dcs.st-and.ac.uk

Abstract. In this very short paper I present a new encoding of binary Constraint Satisfaction Problems (CSP's) into SAT. This encoding has the property that the standard SAT technique of unit propagation is strong enough to establish arc consistency (AC) in the original's in the best possible worst case complexity. Moreover, the time to translate the CSP into SAT and to carry out unit propagation is of optimal time complexity. I prove both these claims, and discuss the implications of these results.

There is continuing in interest in arc consistency algorithms for CSP's [BR01,ZY01] and in translations between CSP's and SAT [Wal00]. In this paper I show some unity between these strands of work by showing a new encoding from CSP's into SAT. In this encoding, unit propagation in the SAT instance is enough to establish arc consistency (AC) in the CSP. This in itself is new, as none of the encodings studied in Walsh's survey achieve this [Wal00]. More remarkably, the translation into SAT and unit propagation is an *optimal* algorithm for establishing AC, as I prove here. However, excitement has to be tempered with the realisation that the encoding may not be important in practice: studying this remains the main piece of work I leave to the future.

For simplicity I assume that all domains are the same size d, and that we have n variables. This is not essential in any way; it merely saves me subscripting each occurrence of d below. The encoding can easily be presented and similar theoretical results given for arbtirary domain sizes. I assume that values run from 1 to d.

Variables have the same interpretation as in the usual encoding of CSP's into SAT. Walsh calls the 'direct' encoding [Wal00].

- $x_{i,v} = T$ means
  'variable i takes value v' or in short '$i = v$'.

We have the following constraints, with v varying from 1 to d. The critical ones are the last:

* I thank members of the APES research group, of which I am a member, for helpful discussions: www.dcs.st-and.ac.uk/~apes. I am especially grateful for helpful discussions with Christian Bessière.

1. At most one value: for $1 \leq v < u \leq d$

$$x_{i,v} = T \rightarrow x_{i,u} = F$$

   There are $nd(d-1)/2$ such constraints.
2. At least one value:

$$x_{i,1} = T \vee x_{i,2} = T \ldots \vee x_{i,d} = T$$

   There are n such constraints of arity d.
3. Support: if $v_1, v_2, \ldots v_k$ are the support for $j = w$ in variable i

$$x_{i,v_1} = F \& x_{i,v_2} = F \& \ldots \& x_{i,v_k} = F \rightarrow x_{j,w} = F$$

   There is one such constraint for each ordered pair (i,j) involved in a constraint, and for each value in the domain of j, for a total of $O(ed)$ constraints of size d. Note that the constraint is unnecessary if every value in i supports $j = w$, so it can be omitted then.

The two key things to prove are that unit propagation establishes Arc Consistency, and that unit propagation can be done in time $O(ed^2)$. For the first, the proof is almost trivial:

Theorem 1 *If no unit propagation is possible, and falsity has not been established, an arc consistent set of domains is given by those values $i = v$ such that the domain of $x_{i,v}$ contains T.*

Proof. Suppose we have a set of current domains of the x variables in which no unit propagation is possible, and no domain is empty. Consider any i, j, w such that there is no support in i for $j = w$. That is, for each possible supporting value v of $j = w$, the domain of $x_{i,v} = \{F\}$. But then the corresponding support clause is unit and forces $x_{j,w}$. That is, the value w is not in the domain of variable j, and the domains are arc consistent as required. Since at least one value must be in each domain, we have a set of non empty domains which are arc-consistent.

Notice as an aside that the 'at most one value' constraints are not used in the proof, and are therefore not necessary for correctness. However in practice I think they would probably be important: specifically, they guarantee maximum possible propagation from a search decision to set a variable to a value. Frisch & Peugniez have looked at the varying need for at-most-one and at-least-one clauses in CSP encodings into SAT [FP01].

Theorem 2 *Translation of a binary CSP into a SAT problem and the exhaustive application of unit propagation can be achieved in time $O(ed^2)$ provided that $e \geq n$.*

Proof. First we show that the size of the SAT problem is no bigger than $O(ed^2)$. There are $O(nd^2)$ clauses of type 1 of constant size, so this is $O(ed^2)$ because of the proviso that $e \geq n$. There are only n clauses of type 2, but each is of size d

for a total size of $O(nd)$, which is again $O(ed^2)$. Finally, there are $O(ed)$ clause of type 3, each of size d, for a total of $O(ed^2)$. The SAT instance is of size $O(ed^2)$ and it can easily be constructed in time $O(ed^2)$.

We now need to show that unit propagation can be exhaustively applied in the required time. For this I appeal to the results of Stickel and Zhang [ZS00]. They show that each literal can be propagated in time proportional to the number of occurrences of the opposite polarity literal. To apply unit propagation exhaustively, we need unit propagate each literal at most once, so the whole instance can be exhaustively unit propagation in time proportional to the size of the instance, in particular in time $O(ed^2)$.

This encoding is optimal with respect to AC in the same way that AC-4 is optimal [MH86], and AC-4 is not used in practice because it is too expensive to build its data structures and then maintain them during search [Wal93]. Indeed, Christian Bessière suggested to me that this encoding can be seen as an encoding of AC-4, so it is likely to suffer from similar problems. Christian points out that maintenance of clause sets within search would lead to similar inefficiences as in AC-4, and I can only agree. However, the encoding is very simple and does have nice theoretical properties. Further, the practical outlook is not entirely negative. Efficient implementations of unit propagations (as are in all state of the art solvers), might make it practicable to build the set of SAT clauses once at the root of the search tree, then maintain it efficiently during search. For example, It would be interesting to look at the effects of using Zhang and Stickel's efficient unit propagation algorithm. This maintains only pointers to the first and last remaining literals in a clause – when these are the same the clause is unit. In the context of the support clauses in this encoding, this corresponds to the first or last element of a domain being removed, upon which the rest of the domain is searched for another supporting value. This corresponds closely to the behaviour of the new AC algorithm AC3.1 [ZY01].

To conclude, I have presented a new encoding of SAT into CSP. This is the first such encoding I am aware of for which unit propagation in the SAT instance establishes arc consistency in the original. It not only does this, but does this in the optimal time for establishing arc consistency using any possible technique. I have proved these results theoretically, but it remains to investigate their practical significance.

References

[BR01] C. Bessière and J.-C. Régin. Refining the basic constraint propagation algorithm. In *Proceedings of IJCAI-01*, pages 309–315, 2001.

[FP01] A. Frisch and T.J. Peugniez. Solving non-boolean satisfiability problems with stochastic local search. In *Proceedings of IJCAI-01*, pages 282–288, 2001.

[MH86] R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

[Wal93] R.J. Wallace. Why AC-3 is almost always better than AC-4 for establishing arc consistency in CSPs. In *Proceedings of IJCAI-93*, pages 239–245, 1993.

[Wal00] T. Walsh. SAT v CSP. In *Proceedings of CP-2000*, number 1894 in LNCS, pages 441–456. Springer, 2000.

[ZS00] H. Zhang and M. Stickel. Implementing the Davis-Putnam method. *Journal of Automated Reasoning*, 24:277–296, 2000.

[ZY01] Y. Zhang and R.H.C. Yap. Making AC-3 an optimal algorithm. In *Proceedings of IJCAI-01*, pages 316–321, 2001.