Unrestricted Nogood Recording in CSP search

George Katsirelos and Fahiem Bacchus

Department of Computer Science, University Of Toronto,[*]
Toronto, Ontario, Canada
[gkatsi,fbacchus]@cs.toronto.edu

Abstract. Recently spectacular improvements in the performance of SAT solvers have been achieved through nogood recording (clause learning). In the CSP literature, on the other hand, nogood recording remains a fairly minor technique for improving backtracking algorithms. In this paper we demonstrate how recent nogood recording techniques from SAT can be generalized to CSPs. The result is a significant enhancement over current nogood recording techniques used in CSPs. We also report on some preliminary empirical results which indicate that generalized nogood recording can have a signficant performance benefit.

1 Introduction

A number of works have investigated nogood recording as a technique for improving backtracking search, e.g., [FD94] (a longer version of this paper, [KB03], contains many more details, as well as a more detailed list of citations). Abstractly, a nogood is an easily checkable condition that can be used to test the nodes of a backtracking search tree. If the condition is true, then there can be no solution to the CSP below that node, otherwise the node remains plausible. Nogoods are discovered as we explore nodes in the backtracking tree. These nogoods capture the reason various nodes of the tree failed to yield a solution. By recording these reasons we can test nodes subsequently visited by the search, backtracking immediately if the node satisfies any previously recorded nogood.

Although it is well known that nogood recording can offer benefits in CSPs, it remains an under utilized algorithmic technique in the field. For example, the main commercial solvers offer no support for nogood recording. Furthermore, the nogood recording techniques in the CSP literature are significantly less general than the more modern techniques utilized in SAT solvers. First, the CSP literature (even recent work like [JDB00]) has only explored the recording of restricted types of nogoods that contain only literals with the same sign. This means that no extra reasoning, e.g., unit propagation, is possible over the nogood database. Second, due to this restriction, alternate ways of learning nogoods from conflicts, e.g., the 1-UIP technique used in Zchaff [MMZ$^+$01], cannot be supported. Instead, the recorded nogoods are always subsets of the decisions made on the way to the conflict. Third, heuristics based on the most recently learned nogoods have not been examined. And fourth, the CSP literature has concentrated on techniques that restrict the number and or size of the nogoods that can be recorded, e.g., using relevance or length bounded nogood recording [BM96].

All of these nogood recording techniques are prominent components of modern SAT solvers, and they have yielded spectacular improvements in SAT solver performance. In

[*] This research was supported by the Canadian Government through their NSERC program.

this paper we demonstrate how all of these improved techniques can be generalized to CSPs, and report on some preliminary empirical results based on an implementation of these techniques. Our empirical results indicate that these more sophisticated techniques for utilizing nogood recording can sometimes have a significant benefit—e.g., allowing us to solve some previously unsolved problems. However, the results also indicate that further tuning of these techniques might be needed for them to attain their full potential with CSPs.

2 Nogoods in Backtracking Search

Definitions: A CSP consists of a set of variables $\{V_1, \ldots, V_k\}$, a domain of values for each variable, and a set of constraints. Each constraint being a boolean valued function over a subset of the variables that maps each assignment of values to these variables to TRUE/FALSE. If its value is TRUE, we say that this particular assignment of values *satisfies* the constraint. A solution to the CSP is an assignment of a value to each variable such that every constraint is satisfied by this set of assignments.

The standard notion of nogood, as explored in the CSP literature, is *a set of assignments that cannot be extended to a solution of the problem*. Each node in the backtracking tree is defined by the set of assignments made so far, with all descendant nodes extending this set of assignments. Hence, if a node covers a nogood, i.e., includes all of the assignments in the nogood, all nodes in the subtree below it must also cover the nogood, and none of them can be a solution.

Nogoods are the CSP equivalent of clauses learned by SAT solvers. To see this consider the simplest SAT encoding of a CSP [Wal00] in which each possible assignment of a variable, $V \leftarrow a$ becomes a proposition asserting that V has been assigned that value. The constraints of the CSP are then encoded as clauses over this set of propositional symbols. In addition, the constraint that each variable must be assigned one and only value, implicit in the CSP encoding, is also encoded as a set of clauses. Under this direct encoding, a nogood $V_1 \leftarrow a_1, \ldots, V_i \leftarrow a_i$ is equivalent to the clause $(V_1 \not\leftarrow a_1, \ldots V_i \not\leftarrow a_i)$. That is, at least one of the assignments (literals) in the nogood must be false. Viewing the nogood as a conjunction (rather than a disjunctive clause), when one of its assignments becomes false the nogood is falsified the nogood becomes inactive. Similarly, when one of its assignments becomes true, we can *reduce* the nogood (implicitly) removing that assignment. Finally, when the nogood has been reduced to a single assignment (become unit), we can *force* this last one assignment to be false (an assignment $V \leftarrow a$ is forced to be false by pruning a from the domain of V).

Using Nogood: Nogood recording can be accomplished in any algorithm that maintains the reasons for each pruned value [Bac00]. These reasons can be stored in a global array: if the reason for pruning $V \leftarrow a$ is NG, then $NoGood[V, a] = NG$. Whenever all the values of a variable V are eliminated through search or propagation, a new nogood is discover and recorded: $NG = \bigcup\{NoGood[V, d] - (V, d) : d \in Domain[V]\}$.

To use the recorded nogoods, the nogood store is checked for unit nogoods after making an assignment, before constraint propagation. Standard nogoods can only be reduced by assignments, so checking at this stage is sufficient. For every nogood NG

that has been made unit by the assignment, we force its remaining untrue assignment to false, by pruning the corresponding value and setting NG as the reason for that pruning. Prunings due to a set of assignments violating a constraint use the assignments to the constraint's scope as the reason for the pruning.[1]

3 Utilizing SAT Techniques for Nogoods

SAT solvers gain much mileage from the fact that their store of recorded clauses can generate long chains of unit propagations, quickly simplifying the problem. Standard nogoods as described above do not support such chains of unit propagation—all the literals in standard nogoods have the same sign so no chaining is possible.

The solution is to record nogoods containing literals of both sign: assignments and non-assignments. With this we obtain propagation in the nogood store. For example, let $NG_1 = \{V_1 \not\leftarrow a, V_2 \not\leftarrow c, V_3 \leftarrow d\}$, and $NG_2 = \{V_1 \not\leftarrow b, V_3 \not\leftarrow d, V_4 \leftarrow c\}$; say the search makes the assignment $V_1 \leftarrow c$; and the value c is pruned from the domain of V_2. $V_1 \leftarrow c$ implies that $V_1 \not\leftarrow a$ and $V_1 \not\leftarrow b$ both become true. The pruning of c means that $V_2 \not\leftarrow c$ becomes true. Thus NG_1 is reduced to the unit nogood $\{V_3 \leftarrow d\}$, which prunes d from the domain of V_3. This then causes NG_2 to become the unit $\{V_4 \leftarrow c\}$, resulting in c being pruned from the domain of V_4.

Recording Generalized Nogoods: To understand how general nogoods can be recorded during search, it is useful to consider why it is that the unioning of nogoods is a valid way of producing a new nogood. Implicit in the CSP representation is a "must have a value" nogood, $M = \{V \not\leftarrow x_1, \ldots, V \not\leftarrow x_d\}$. When we have a nogood $NG_i = \{V \leftarrow x_i, R_i\}$ for each of V's values, we can resolve each NG_i against M. The final result will be $\{R_1, \ldots, R_d\}$.

To discover generalized nogoods, we replace the above procedure with one that incrementally unwinds the "must have a value" nogood, always replacing the chronologically most recent assignment by the reason (nogood) associated with it. Eventually, we have a nogood whose most recent assignment is the choice assignment in its level. We can then backtrack to that level, and store the nogood. The key is that during this incremental unwinding process some of the non-assignments in the "must have a value" nogood persist.

Unique Implication Point (UIP): An alternative to completely unwinding the "must-have-a-value" nogood is to stop when exactly one assignment remains at the current level. That assignment is called a unique implication point [ZMMM01]. Nogoods computed via UIPs can be quite different from those computed via the standard technique. We have experimented with both techniques for nogood recording.

[1] In the case of prunings due to GAC propagation the reason (nogood) for the pruning can be composed from the nogoods of the values pruned from the other variables in the constraint's scope.

Non-Binary Domain Processing The SAT encoding of a CSP contains clauses to enforce the constraint that each variable must one and only one value. These clauses (nogoods) could be added to the nogood store prior to search, but it is more efficient (and effective) to account for the implicit presence of these clauses by special processing of the nogoods. There are three cases that can be processed in this way. First, whenever a variable V is assigned a value a all assignments $V \leftarrow x$ for $x \neq a$ become false. Second, if a nogood contains an assignment to a variable we can remove all non-assignment to the same variable prior to storing the nogood (the assignment subsumes the non-assignments). Third, if a nogood is reduced to a collection of non-assignments to the same variable, we can immediately prune all other values from the variable's domain. It can be noted that the last two cases cannot be captured by simply unit propagating the implicit "must have a value" and "only-one-value" nogoods.

Heuristics based on recently recorded nogoods: The recorded nogoods can be used to rank the unassigned variables by the frequency with which they appear in recently recorded nogoods. The Variable Decay heuristic utilized in SAT solvers uses this technique to encourage the tree search to produce short clauses. We have experimented with such a heuristic, but to date have not found an effective version of it in the context of CSPs. Nevertheless, there is increasing evidence that heuristic guidance can be the most effective of the nogood recording techniques utilized in SAT solvers. Hence, we are continuing our investigations in this area.

Recording large numbers of nogoods: SAT solvers utilize lazy data structures (watch literals) to optimize the management and propagation of large numbers of large nogoods. With these techniques previous restrictions on nogood recording utilized in CSP algorithms can be removed, and instead large databases of nogoods can be efficiently managed.

4 Empirical Results

We have implemented unrestricted nogood recording, with standard nogoods as well as generalized nogoods. We have also implemented UIP processing which can be used with generalized nogoods. All experiments were performed on a Pentium-4 2.2 GHz machine, with 4GB of RAM, and times are reported in CPU seconds.

We report on one set experiments containing 100 instances of hard crossword puzzles, presented in [BCSvB01], Beacham et al.. Beacham et al. identified EAC (extentional arc consistency) as the best algorithm. In our experiments we compared their implementation of EAC against various nogood recording algorithms based on FCCBJ: *FCCBJ+RB* (3rd order relevance bounded recording of standard nogoods), *FCCBJ+S* (unrestricted recording of standard nogoods), *FCCBJ+G* (unrestricted recording of generalized nogoods) and *FCUIP* (unrestricted recording of generalized nogoods that result from UIP reasoning). In Table 1 we report on the instances that were unsolvable by at least one of the algorithms. The rest were easily solvable by all, with no major time differences.

We see that relevance bounded nogood recording is not very effective, always being slower than the rest of the algorithms, and solving fewer problems. FCCBJ+S (and

	EAC		FCCBJ+RB		FCCBJ+S		FCCBJ+G		FCCBJ+UIP	
Problem	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
UK-21.04	20.36	447	817.36	5924008	1491.22	1170356	-	-	5312.95	824067
UK-23.06	84.72	1306	-	-	5203.07	2994762	-	-	1028.48	835313
UK-23.10	210.54	1834	177.06	1236956	134.59	415718	2199.28	710566	-	-
words-15.01	-	-	-	-	5395.93	2423058	-	-	1122.5	655479
words-15.10	2.29	265	-	-	5752.77	3702947	-	-	-	-
words-19.03	502.12	21096	-	-	156.56	526797	-	-	490.26	482251
words-19.04	10.58	580	-	-	15.46	118325	43.47	124524	30.53	97721
words-21.01	-	-	-	-	3921.84	5056685	-	-	1266.06	1207743
words-21.06	4.35	484	-	-	45.11	168548	350.27	287223	31.94	75443
words-23.03	-	-	-	-	144.57	672178	5006	1902648	1959.23	1400760
words-23.04	207.16	3783	-	-	14715.87	6194196	-	-	-	-
words-23.08	-	-	-	-	-	-	-	-	4529.65	1817350
words-23.09	269.79	9933	-	-	-	-	-	-	-	-

Table 1. Crossword puzzles with FC-based algorithms

GACCBJ+S) present either a clear improvement. Recording generalized nogoods with UIP reasoning in some cases can payoff, but more tuning is need to obtain maximal benefit from this technique.

5 Conclusion

We have developed methods for importing current clause learning techniques from SAT into CSPs. These techniques do allow us to solve some previously unsolved problems, but more work remains to obtain to maximal potential from these techniques in the CSP context.

References

[Bac00] Fahiem Bacchus. Extending forward checking. In *International Conference on Principles and Practice of Constraint Programming*, number 1894 in Lecture Notes in Computer Science, pages 35–51. Springer-Verlag, New York, 2000.

[BCSvB01] Adam Beacham, Xinguang Chen, Jonathan Sillito, and Peter van Beek. Constraint programming lessons learned from crossword puzzles. In *Proceedings of the 14th Canadian Conference on Artificial Intelligence*, pages 78–87, 2001.

[BM96] R. J. Bayardo Jr and D. P. Miranker. A complexity analysis of space-bounded learning algorithms for the constraint satisfaction problem. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 298–304, Portland, Oregon, 1996.

[FD94] Daniel Frost and Rina Dechter. Dead-end driven learning. In *Proceedings of the AAAI National Conference*, pages 294–300, 1994.

[JDB00] Narendra Jussien, Romuald Debruyne, and Patrice Boizumault. Maintaining arc-consistency within dynamic backtracking. In *International Conference on Principles and Practice of Constraint Programming*, number 1894 in Lecture Notes in Computer Science, pages 249–261. Springer-Verlag, New York, 2000.

[KB03] George Katsirelos and Fahiem Bacchus. Unrestricted nogood recording in csp search. availble at www.cs.toronto.edu/˜gkatsi/publications.html, 2003.

[MMZ$^+$01] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *Proc. of the Design Automation Conference (DAC)*, 2001.

[Wal00] Toby Walsh. Sat v csp. In *International Conference on Principles and Practice of Constraint Programming*, number 1894 in Lecture Notes in Computer Science, pages 441–456. Springer-Verlag, New York, 2000.

[ZMMM01] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of IEEE/ACM International Conference on Computer Design (ICCAD)*, pages 279–285, 2001.