Pruning Search Space for Weighted First Order Horn Clause Satisfiability

Naveen Nair[123], Anandraj Govindan[2], Chander Jayaraman[2], Kiran TVS[2], and Ganesh Ramakrishnan[21]

[1] IITB-Monash Research Academy, Old CSE Building, IIT Bombay
[2] Department of Computer Science and Engineering, IIT Bombay
[3] Faculty of Information Technology, Monash University
{naveennair,ganesh}@cse.iitb.ac.in
{anand.itsme,chanderjayaraman,t.kiran05}@gmail.com

Abstract. Many SRL models pose logical inference as weighted satisfiability solving. Performing logical inference after completely grounding clauses with all possible constants is computationally expensive and approaches such as LazySAT [8] utilize the sparseness of the domain to deal with this. Here, we investigate the efficiency of restricting the Knowledge Base (Σ) to the set of first order horn clauses. We propose an algorithm that prunes the search space for satisfiability in horn clauses and prove that the optimal solution is guaranteed to exist in the pruned space. The approach finds a model, if it exists, in polynomial time; otherwise it finds an interpretation that is most likely given the weights. We provide experimental evidence that our approach reduces the size of search space substantially.

Key words: First Order Logic, Horn Clauses, MaxSAT, Satisfiability

1 Introduction

Representing sets of objects and their relationships in a compact form has been the focus of researchers for more than a decade. Weighted first order formulas proved to be one such representation which also allows inference and learning in a structured way. Inference in these sets of formulas are mostly done by Satisfiability testing. We summarize some of the works that addressed the problem of satisfiability in the next paragraph.

Traditional SAT solvers in propositional logic try to find an assignment for all the literals that makes all the clauses true. They return a model if it exists or return unsatisfiable. SAT solvers such as DPLL [6] give exact solutions but employ backtracking and take exponential time in the worst case. Local search methods for satisfying the maximum number of clauses (Max-SAT) has been implemented in GSAT [1], WalkSAT [2] etc. Weighted Max-SAT problems assign weights to the clauses and aim to minimize the sum of the weights of unsatisfied clauses. Teresa et. al. proposed a Lazy approach [9] which uses some bound computation and variable selection heuristic for satisfiability. MiniMaxSAT [4] uses a

depth-first branch-and-bound search approach for satisfiability. Satisfiability of first order logic (univerally quantified Conjunctive Normal Form (CNF)) can be done by grounding all the clauses (exponential memory cost) and then running satisfiability as in propositional case. Since, many learning techniques require the repeated use of inference and satisfiability, complete grounding of the clauses becomes a bottle neck. Lifted inference [5] techniques used first order variable elimination for probabilistic inference but have not proved their applicability in large domains. As there could be many contradictions in real world, it is better to perform weighted satisfiability. Weighted satisfiability solvers are used for MPE/MAP inference in relational domains [7]. But the complete grounding issue remained unsolved. A LazySAT approach [8] that doesn't ground all clauses was proposed for satisfiability in domains where majority of ground atoms are false. Their approach, a variation of WalkSAT, keeps track of all the clauses that can be affected when a literal in an unsatisfied clause is flipped. Recently in [10], the ground clauses that are satisfied by the evidence are excluded. The approach, which is depended only on the evidence set, processes each clause independently and does not find the dependent clauses transitively. Mihalkova et. al. cluster query literals and perform inference for cluster representatives [12]. Queries are clustered by computing signatures using a recursive procedure based on adjacent nodes. Inference is performed for each cluster representative by running MaxWalkSAT on corresponding Markov Network constructed recursively. Alen Fern mentions about the applicability of Forward Chaining in horn clauses [11] but has not given any alogrithm or proof for doing so. In the case of contradicting clauses, it is not straight forward to do forward chaining. The objective of our work is stated in the next paragraphs.

We address the issue of complete grounding by restricting our domain to first order horn clauses and pruning the search space for satisfiability. Our approach caters to several real world applications that use the horn clausal language.

If a set of horn clauses are fully satisfiable, then a minimal model can be found using T_Σ operator (referred to as the immediate consequence operator T_P in [3]) in polynomial time. However weighted unsatisfiable problems require to find the most likely state based on the weights. We propose an extension to the minimal model approach wherein we find (i) the relevant set of ground horn clauses which has a potential to be part of a contradiction and (ii) an interpretation near to the result. MaxSAT algorithm can be used on this subset of clauses, (optionally) starting from the interpretation returned, to get the most likely state. We also prove that local search for optimality in the pruned space cannot affect the satisfiability of the rest of the clauses. Our experiments show that the approach reduces search space substantially and helps maxSAT to converge in short time.

The paper is organized as follows. Section 2 explains the conventional T_Σ operator and the proposed $Modified\_T_\Sigma$ operator. In section 3, we give an overall procedure for satisfiability and also state and prove our claims. Results are discussed in section 4. We conclude our work in section 5.

2 Satisfiability in Horn Clauses

If any of the atoms in the body part of a horn clause is false, then the clause is satisfied because of its inherent structure of having atmost one positive atom and all others negative. The groundings of a set of first order horn clauses (Σ) with all the constants give a large set in which majority of the atoms are false in real world. This makes a large subset of these clauses satisfied by default. We can neglect these clauses and restrict our attention to the clauses that have a potential to be part of a contradiction. We call this set, the relevant set (RS).

We propose an algorithm, $Modified\_T_\Sigma$, to identify the relevant set along with the truth assignments that are almost near to the result. Local search for optimality can be done on this set, starting with the interpretation returned, rather than considering the huge set of clauses and arbitrary truth assignments. Next we explain T_Σ before going to the Modified version.

2.1 T_Σ Operator

T_Σ Operator provides a procedure to generate an interpretation from another. It builds on the concept that for satisfiability in horn clauses, all the unit clauses should be True and if the body of a clause is True, then the head should also be True. Let I_k be the interpretation at the k^{th} step of the operation. Then,

$$I_{k+1} = I_k \cup T_\Sigma(I_k) \tag{1}$$

where,
$$T_\Sigma(I) = \{a : a \leftarrow body \in \Sigma \quad and \quad body \subseteq I\} \tag{2}$$

If we start with $I = \emptyset$, and iteratively apply the above function assignment (with respect to the set of clauses), we will eventually converge at an interpretation that is the minimal model of the formulae if one exists. If there is no model for this set, the operation will reach a contradiction and will return *Unsatisfiable*.

In weighted satisfiability problems, if the given set is unsatisfiable, we need to get a most likely state based on the weights. MaxSAT algorithms can do this optimization. Since applying MaxSAT to the complete groundings is expensive, we improve the above method to prune the search space for MaxSAT. $Modified\_T_\Sigma$ Step described in the next section helps us to prune the search space.

2.2 $Modified\_T_\Sigma$ Step

$Modified\_T_\Sigma$ operation returns a model if one exists; Otherwise returns the set of clauses to be used by a local search algorithm and an initial interpretation for the local search. The method is given in Algorithm 1 and is explained below.

Start with applying T_Σ to the set of ground clauses until it converges in a model or some contradiction is attained. In the former case, we can stop and return the current interpretation as the solution. If we land up in a contradiction, we get an atom whose truth value determines the set of clauses satisfied. We assign true to the atom and proceed further till no more clauses can be visited. All the clauses discovered by $Modified\_T_\Sigma$ irrespective of whether satisfied or

Algorithm 1 $Modified\_T_{\Sigma}(\Sigma, DB)$
Input: Σ, the set of first order clauses with weights; DB, evidence set given.
Output: RS, the set of clauses to be considered for optimization; TS, truth assignments of all atoms in RS except those in DB.

```
1. TS := ∅
2. RS := ∅
3. for each unit clause c in Σ do
4.   for each grounding c' of c do
5.     if c' ∉ RS then
6.       Add c' to RS
7.     end if
8.     if c'.head ∉ {TS ∪ DB} then
9.       Add c'.head to TS
10.    end if
11.  end for
12. end for
13. repeat
14.   for each non unit clause c in Σ do
15.     for each grounding c' of c where c'.body ⊆ {TS ∪ DB} do
16.       if c' ∉ RS then
17.         Add c' to RS
18.       end if
19.       if c'.head ∉ {TS ∪ DB} then
20.         Add c'.head to TS
21.       end if
22.     end for
23.   end for
24. Until no new clauses are added to the set RS
25. Return {RS, TS}
```

not form the relevant set. The interpretation got at the end of the algorithm can optionally be used as initial truth assignment for the optimization step. Note that the truth values for evidences given are always true and cannot be changed.

Any Weighted satisfiability algorithm can be applied on the Relevant Set of clauses and the (optional) initial truth values to get a minimum cost interpretation. We now discuss weighted satisfiability approach using $Modified\_T_{\Sigma}$.

3 Modified_Weighted_SAT

In the new approach, $Modified\_T_{\Sigma}$ operation is used to find relevant subset as well as initial truth assignment. Then weighted MaxSAT version given in Algorithm 2 is used. Algorithm 3 gives the overall algorithm.

Claim 1. *All the unsatisfied clauses will be in RS.*

Proof. A horn clause c' is unsatisfied if $c'.body \subseteq \{TS \cup DB\}$ and $c'.head \notin \{TS \cup DB\}$. Step 6 in $Modified\_T_{\Sigma}$ adds all clauses c' of the form $(c'.head \vee \neg True)$

to RS irrespective of whether it is satisfied or not. Step 17 in $Modified\_T_\Sigma$ adds all clauses c' where $c'.body \subseteq \{TS \cup DB\}$. This covers both the cases of $c'.head$ is $True$ and $c'.head$ is $False$. All other clauses c'' where $c''.body \nsubseteq \{TS \cup DB\}$ are satisfied by default. So set of unsatisfied clauses is a subset of RS. □

Claim 2. *Any flip done in any maxSAT step to make an unsatisfied clause satisfiable only affects the satisfiability of clauses in RS.*

Proof. Let us prove this by contradiction.
Suppose a clause, $c' = (l_1 \vee \neg l_2 \vee \neg l_3 \vee \ldots \vee \neg l_n)$ is not satisfied by the current assignments in $\{TS \cup DB\}$. This happens only when $l_1 \notin \{TS \cup DB\}$ and $\forall i = 2 \ldots n \quad l_i \in \{TS \cup DB\}$. To make c' satisfied, there are two cases. case 1: flip l_1, case 2: flip any of $l_2, l_3, \ldots, l_n$.
case 1: Flip l_1 ($False$ to $True$). Assume that flipping l_1 will affect the state of a clause $c'' \notin RS$. Since $c'' \notin RS$, $c''.body \nsubseteq \{TS \cup DB\}$. Otherwise step 17 in $Modified\_T_\Sigma$ would have covered c'' and it would have been in RS. Also all the unit clauses are covered by step 6 in $Modified\_T_\Sigma$.
Now let $c''.head = l_1$. Since flipping $c''.head$ to $True$ changes the state of c'', $c''.body \subseteq \{TS \cup DB\}$. If this is the case, c'' should have been covered by step 17 in $Modified\_T_\Sigma$ and would have been in RS. Hence the assumption that $c'' \notin RS$ is wrong.
Now let $l_1 \in c''.body$ and flipping it to $True$ changes the state of c''. Then $c''.body \backslash l_1 \subseteq \{TS \cup DB\}$. But applying our approach to c' would have made $l_1 \in TS$ and transitively $c''.body \subseteq \{TS \cup DB\}$ and $c'' \in RS$. Hence the assumption that $c'' \notin RS$ is wrong.
case 2: Flip any $l_i \in \{l_2, l_3, \ldots, l_n\}$ ($True$ to $False$). Assume that flipping l_i will affect the state of a clause $c'' \notin RS$. Since $c'' \notin RS$, $c''.body \nsubseteq \{TS \cup DB\}$. Otherwise step 17 in $Modified\_T_\Sigma$ would have covered c'' and it would have been in RS. Also all the unit clauses are covered by step 6 in $Modified\_T_\Sigma$.
Now let $c''.head = l_i$. Since flipping $c''.head$ to $False$ changes the state of c'', $c''.body \subseteq \{TS \cup DB\}$. If this is the case, c'' should have been covered by step 17 in $Modified\_T_\Sigma$ and would have been in RS. Hence the assumption that $c'' \notin RS$ is wrong.
Now let $l_i \in c''.body$ and flipping it to $False$ changes the sate of c''. Then before flipping, $c''.body \subseteq \{TS \cup DB\}$ which must have been covered by step 17 in $Modified\_T_\Sigma$ and $c'' \in RS$. Hence the assumption that $c'' \notin RS$ is wrong. □

Claim 3. *If α is the cost of an optimal solution to RS, then α is the cost of an optimal solution to Σ*

Proof. let β and γ be the cost of optimal solutions to Σ and RS respectively. That is β should be the sum of costs of RS and $\Sigma \setminus RS$. Increase in cost occurs only because of contradictions and this is in the set RS (proved in claim 1). The best possible solution to the non contradicting part is zero. We get a minimum cost solution for RS part using MaxSAT and any modification to that can result (proved in claim 2 that this doesn't affect $\Sigma \setminus RS$) an increase in cost only in RS. Therefore $\beta = 0 + \gamma$ and thus $\beta = \gamma$ □

Algorithm 2 Modified_Weighted_MaxSAT(Σ_g, TS, DB, $target$)

Input: Σ_g, all grounded clauses with weights; TS, initial truth assignment; DB, the evidence given; target, the upper bound of cost.
Output: TS, An interpretation that is the best solution found.

```
1. atms := atoms in Σ_g
2. repeat
3.    cost := sum of weights of unsatisfied clauses
4.    if cost ≤ target
5.       Return Success, TS
6.    end if
7.    c := a randomly chosen unsatisfied clause
8.    for each atom a ∈ c and a ∉ DB do
9.       compute DeltaCost(a), the cost incurred if a is flipped
10.   end for
11.   a_f := a with lowest DeltaCost(a)
12.   TS := TS with a_f flipped
13.   cost := cost + DeltaCost(a_f)
14.until the cost is no more decreasing
15.Return Failure, TS
```

Algorithm 3 Weighted_HornSAT(Σ, DB, $target$)

Input: Σ, the set of first order clauses with weights; DB, evidence set given; target, maximum cost expected for the optimization step if required.
Output: TS, An interpretation when combined with DB gives the (local) optimum solution.

```
1. {RS, TS} := Modified_T_Σ(Σ, DB)
2. if {TS ∪ DB} is a model for Σ then
3.    Return TS
4. else
5.    TS := Modified_Weighted_MaxSAT(RS, TS, DB, target )
6. end if
7. Return TS
```

4 Results

We implemented new HornSAT algorithm with $Modified\_T_\Sigma$ in java. We have done our experiments in AMD Athlon 64 bit dual core machine (2.90 GHz) with 2.8 GB RAM and running Ubuntu 8.04.

Our results show that, for satisfiablility, the $Modified\_T_\Sigma$ method gives a fewer number of groundings to optimize and that the optimization step converges in a short time when the search space is pruned.

We used the uwcse knowledge base and dataset provided by alchemy [13] for our experiments after making small modifications to make the clause set horn. The constants given as the evidence set is considered as the complete domain

for each variables. We have run three experiments on each dataset. First experiment does the complete groundings and runs MaxWalkSAT. Second grounds the clauses with pruning and runs traditional MaxWalkSAT with random truth assignments. The third experiment runs MaxWalkSAT on the pruned clauses set with the initial truth assignment returned by $Modified\_T_{\Sigma}$. Evidence set of different sizes are used and the comparison is given in Table 1. Figures 1.a, 1.b, 1.c portrays the results when 181 atoms of uwcse language dataset, 87 atoms of uwcse language dataset and 766 atoms of uwcse AI dataset are used respectively as evidence set. Experimental results show that the proposed method outperforms the traditional approach in terms of memory and speed. Implementation and other details are available at www.cse.iitb.ac.in/~naveennair/HornSatisfiability/

Table 1. Results with uwcse KB and different evidence sets

	No. of groundings		Converged Cost			Time taken (ms)		
Evidence set	Complete	Pruned	Complete grounding + MaxWalkSAT	Pruned + MaxWalkSAT	HornSAT	Complete grounding + MaxWalkSAT	Pruned + MaxWalkSAT	HornSAT
language 181 atoms	508788	6908	90.452	70.265	70.265	2475736	1823778	6896
language 87 atoms	177738	3205	81.463	37.892	37.892	2329459	1098285	2351
AI_766 atoms	Memory Error	182690	NA	344.584	344.584	NA	7507578	7462967

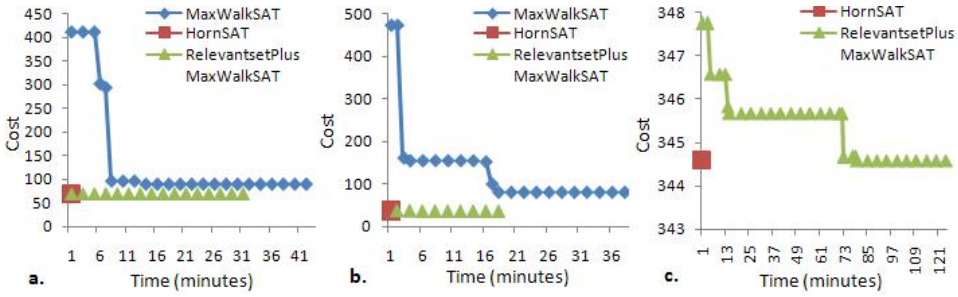

Fig. 1. Comparison of the approaches when applied to uwcse KB. **a.** All 181 atoms from language dataset are given. **b.** 87 atoms from language dataset are given. **c.** All 766 atoms from AI dataset are given. In this experiment, complete grounding approach failed and didnot give any result.

5 Conclusion and Future work

Several ground clauses formed as a result of propositionalization of first order horn formulae are satisfied by default and it is a wastage of resources to consider them for optimization. We presented an algorithm that prunes the search space and proved that the optimal solution must lie in the pruned space. Experiments indicate the scope for efficient inference using MaxSAT for the set of horn clauses.

The algorithm can be extended for general clauses by assigning true value artificially to all non-negated atoms if all the negated atoms are true and proceeding like in Algorithm 1.

Acknowledgments. We would like to thank Dr. Ashwin Srinivasan, IBM India Research Laboratory for his helpful comments.

Supplementary materials: appendix.pdf

References

1. Bart Selman, Hector Levesque, David Mitchell: A New Method for Solving Hard Satisfiability Problems. In: AAAI-92, San Jose, CA, 440-446 (1992)
2. Bart Selman, Henry Kautz, Bram Cohen: Local Search Strategies for Satisfiability Testing. In: Second DIMACS Implementation Challenge on Cliques, Coloring and Satisfiability (1993)
3. Christopher John Hogger: Essentials of logic programming. Oxford University Press, New York, USA (1990)
4. Federico Heras, Javier Larrosa, Albert Oliveras: MINIMAXSAT: an efficient weighted max-SAT solver. In: Journal of Artificial Intelligence Research Volume 31, Issue 1 1-32 (2008)
5. Jacek Kisynski, David Poole: Lifted aggregation in directed first-order probabilistic models. In: Proceedings of the 21st international jont conference on Artifical intelligence, California, USA 1922-1929 (2009)
6. Martin Davis, Hilary Putnam, George Logemann and Donald W. Loveland: A Machine Program for Theorem Proving. In: Communications of the ACM 5 (7): 394-397 (1962)
7. Parag Singla, Pedro Domingos: Discriminative training of Markov Logic Networks. In: AAAI-05, 868-873 (2005)
8. Parag Singla, Pedro Domingos: Memory-Efficient Inference in Relational Domains. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 488-493), 2006. Boston, MA, AAAI Press(2006)
9. Teresa Alsinet, Felip Many, Jordi Planes: An efficient solver for weighted Max-SAT. In: Journal of Global Optimization, Springer Netherlands, 61-73 (2008)
10. Jude Shavlik, Sriraam Natarajan: Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In:Proceedings of the 21st international JCAI (2009)
11. Alan Fern: A Penalty-Logic Simple-Transition Model for Structured Sequences. In:Computational Intelligence, 302?334 (2009)
12. Lilyana Mihalkova, Matthew Richardson: Speeding up inference in statistical relational learning by clustering similar query literals. In:ILP 2009 (2009)
13. http://alchemy.cs.washington.edu/