Modelling in Propositional Logic: A Case Study

No Author Given

No Institute Given

Abstract. Car sequencing has been a traditional benchmark in the Operation Research and Constraint Programming community. In this paper we will demonstrate that state-of-the-art propositional satisfiability solvers can compete with these established paradigms. We will show how to express the traditional car sequencing decision problem in propositional logic and give an empirical evaluation of this non-trivial encoding. By this we underline the applicability of Logic as a representation language for Computation.

1 Introduction

3. Do the introduction fully . What we will show in the this paper. A defintion of this problem is to translate and solve it with state-of-the art SAT solvers. More insight is required that this and we will show with the example of car sequencing the methodoly to improve a modelling.

Contributions of the paper:

1) to the (best of our knowledge) first non-trivial SAT model for car sequencing and 2) demonstration of its usefulness on experiments with the CSP lib 3) a comprehensive future work secion with future work and promising research directives.

Introduce the NP complete problems [Garey and Johnson, 1979] and its canoncal problem of finding a satisfying assignment to a formula given in conjucntive normal form. Reductions among the problems in this classse, to express one in terms of the other in polynomial time and space. Thus solving one of the problems in P would solve all. This problem is still unsolved and considered one of the greatest open problems in theoretical computer science. The majority of researchers in the field believe that these two complexity classes are different and all empirical evaluations underline this believe.

1.1 SAT Solving

First we will state some preliminary definitions and then give a brief background on satisfiability solving.

Given a set of boolean variables $X = \{x_1, x_2 \dots x_n\}$, a literal is of the form x_i or $\neg x_i$ and a clause is a disjunction of literals. The boolean satisfiability problem (SAT) requests a satisfying assignment to a formula that consists of a conjunction of disjunction of literals over X, or a proof if no assignment exists. A formula of this form is in conjunctive normal form (CNF). A formula. in

propositional logic (over connectives $\neg, \wedge$ and $\vee$) can easily be transformed to CNF [Tseitin, 1983]. A clause with only one literal is called unit clause.

The SAT problem is considered the cannonical NP-complete problem, and in addition to its theoretical relavance it is emerging as a new paradigm to solve arbitrary problems its complexity class. The general idea here is to reduce a given problem to a (potentially large) formula in CNF and apply a general purpose domain independent SAT solver to find an assignment or prove unsatisfiability.

In the last two decades there has been tremendous research in both theoretical and practical SAT solving techniques, for a comprehensive overview we refer to [Biere et al., 2009]). For the scope of this paper we briefly mention the basic underlying algorithm to most successful SAT solvers and its mayor extensions. The fundamental DPLL algorithm ([Davis and Putnam, 1960]) is based on a partial assignment backtrack search and a light-weight reasoning method called unit propagation (UP). The concept is to identify literals in clauses that have become unit under the current assignment and forcing the assignment of this literal such that the clause is satisfied. Further improvements to the DPLL algorithms go under the name of Conflict-Driven Clause Learning (CDCL) solvers. These solvers record an approriate reason for each conflict in the search and potentially prune unseen parts of the search tree. Furthermore, SAT solvers contain robust domain-independent branching and decision heuristics and in many cases solve formulas with millons of variables and clauses. Now application of SAT solving reach from formal verifciation to the domain of logistical problems.

The open source SAT solver Minisat [Eén and Sörensson, 2003] represents a cannonical, properly engineered implementation of state-of-the boolean solving techniques and has been a starting point for many improved implementations.

1.2 Modelling in SAT

4 . Discribe the challenges in boolean modelling. Propagators of CP are rather procedural descriptions of the reasoning task. Pro: Declarativity of the problem. Low level view and give advantages when tweaking the model, which can make the difference for hard instances. Links to the handbook of CP.

Give list of techniques on what to focus on with encodings. Modelling in SAT is not just translating to CNF and a solver will find the solution. Common pitfalls.

1) Design of variables, representing integer domains references 2) identifying global constraints 3) efficiently decompose these constraints to CNF by introducing auxiliary variables and 4) Introduce redundant constraints and break symmetries.

Quality measures of CNF formulation for DPLL based solvers. Size in terms of clauses and total number of literals. A more involed measure is the. Assumption that consistency measures from CP translates to SAT, show some papers.

1.3 Car Sequencing

First we will define the car sequencing problem (CS) as given in the library of CSP problems [Gent and Walsh, 1999] and then reference the relevant literature.

Car sequencing deals with the problem of scheduling cars along an assembly line with capacity constraints for different stations (e.g. radio, sun roof, air-conditioning, etc). Formally there are n cars divided into k classes, that require a subset of possible options . Each option l has a limit of u_l occurences on each subsequence of length q_l (denoted as u_l/q_l), i.e. no more than u_l cars with option l can be sequenced among q_l consecutive cars. A solution to this problem is a complete, valid sequence of cars.

Example 1. Given classes $\{1, 2, 3\}$ and options $\{1, 2\}$. The demands (number of cars) for the classes are $3, 2, 2$ and constraints on options are $1/2$ and $1/5$, respectively. Class 1 requires no options, class 2 needs option 1 and class 3 needs both option 1 and 2. The only legal sequence for this problem is $3, 1, 2, 1, 2, 1, 3$, since class 2 and 3 cannot be adjacent and cars from class 3 need to be at least 5 positions apart.

The problem was proven to be NP-complete by Gent in [Gent, 1998]. A early approach to solve the problem with CP came by the introduction of a special propagator in [Régin and Puget, 1997]. Subsequently local search techniques and integer programming were used to improve solutions to the benchmark set [Gottlieb et al., 2003], [Estellon et al., 2006],[Gravel et al., 2005]. More recently, it has been shown that CP can keep up with the other paradigms in solving the current set of benchmark problems, [Siala et al., 2012].

2 Modelling the Car Sequencing Problem

5 . This section introduces the CNF encoding of the car sequencing problem. First give background and references to the underlying ideas of the encoding. Then we show how to extend a so called counter encoding to represent a global constraint in this problem and then show how to link Cars and options.

For convenience we introduce notation that is used in the rest of the paper. The demand constraints requires a global cardinality constriants. There has been numerous proposals for translations of these constraints. For our purpose we will use a counter encoding for the cardinality constraint. Set the link to CP of the decomposition to the sequence constraints in ninas papers. Then we reuse the auxiliary variabels of the counter to propagate. Also we enforce bounds consistency on the cumulative sums, by encoding the domains of the cumulative sums by an order encoding.

2.1 Counter Encoding

We will show how to encode a cardinality constraint of the form $\sum_{i \in \{1 \dots n\}} x_i = d$ where x_i are boolean variables and d is a fixed demand. The key idea is to

introduce auxiliary variables to represent cumulative sums. The integer sums are translated to boolean variables using the order encoding ([Tamura et al., 2009]).

In this section we will use two types of variables, for each position i

- x_i is true iff an object (car or option) is at position i
- $y_{i,j}$ is true iff in the positions $0, 1 \dots i$ the property holds at least j times.

The following equation formalizes the relationship between x and y:

$$y_{i,j} \iff (j \leq \sum_{l=1}^{i} x_l)$$

We now state the clauses that defines the counter: Note that in order to correctly set unit clauses for the edge cases, we need we generate clauses also for the positions 0, demand 0 and demand $d + 1$.

The following binary clauses relate the variables y among each other:

$$\bigwedge_{i \in \{0 \dots n-1\}} \bigwedge_{j \in \{0..d+1\}} \neg y_{i,j} \vee y_{i+1,j} \tag{1}$$

$$\bigwedge_{i \in \{1..n\}} \bigwedge_{j \in \{1..d+1\}} \neg y_{i,j} \vee y_{i-1,j-1} \tag{2}$$

These clauses restrict assignments of the auxiliary variables to consistenty represent a counter that maximally increases by 1 in each position.

Now we need to create channeling clauses to push and pull information between x and y. First we restrict the counter not to increase if x_i is false:

$$\bigwedge_{i \in \{1 \dots n\}} \bigwedge_{j \in \{0..d+1\}} x_i \vee \neg y_{i,j} \vee y_{i-1,j} \tag{3}$$

Second we define clauses that push the counter up if x_i is true.

$$\bigwedge_{i \in \{0 \dots n-1\}} \bigwedge_{j \in \{0..d\}} \neg x_{i+1} \vee \neg y_{i,j} \vee y_{i+1,j+1} \tag{4}$$

As a final step we "initialize" the counter by setting the following unit clauses.

$$y_{n,d} \wedge \left(\bigwedge_{i \in \{0 \dots n\}} y_{i,0} \right) \wedge \neg y_{0,1} \wedge \left(\bigwedge_{i \in \{0 \dots n\}} \neg y_{i,d+1} \right) \tag{5}$$

The clauses set the lower part of the counter to true - it always has to hold - and the upper part of variables ot false.

Note that the given encoding is not the most compact one for a single cardinality constraint. However, the auxiliary variables are reused in the following section to state the capacity retriction.

2.2 The Capacity Constraint

We now encode a more global view on the counting constraint by integrating the capacity of each subsequnce. Given a sequence of boolean variables of which exactly d have to be true and each window of size q cannot contain more than u true variables. More formally:

$$\text{AtMostSeqCard } (u, q, d, [x_1, \ldots, x_n]) \iff (\sum_{i=1}^{n} x_i = d) \wedge \bigwedge_{i=1}^{n-q} (\sum_{l=1}^{q} x_{i+l} \leq u)$$

To detect unsatisfiabiliy of this constraint by the CNF decomposition we propose to decompose the cardinality constraints by reusing the counter encoding of the previous section and add the following binary clauses:

$$\bigwedge_{i \in \{q \ldots n\}} \bigwedge_{j \in \{u \ldots d+1\}} \neg y_{i,j} \vee y_{i-q,j-u} \tag{6}$$

The clauses restrict the internal counting not to exceed the window capacities.

2.3 Car Sequencing

Let C be the set of classes and O be the set for options. An instance for a car sequencing problem is given by a mapping $f : C \to 2^O$, relating to each class a set of options and the individual demands for the classes. We can use this information to generate for each class and option a AtMostSeqCard constraint:

For each class we are given cardinality constraint by the demands and for each option a capacity rule. From this we can easily precompute for each option its explicit,exact demand by summing up the demands of all classes that require this options. Furthermore we can determine the strictest capacity constraint for each class from all its options. This gives us a an AtMostSeqCard constraint for each option and class. All these cosntraints are decomposed to CNF by using equations (1) to (6).

Now we are left with relating options and cars. This is done by the following clauses. Let c_i^k represent a car of class k in position i and o_i^l option l be in position i. Let $g : O \to 2^C$ be the reverse mapping relating an option to the set of classes that contain this option.

$$\bigwedge_{i \in \{1 \ldots n\}} \bigwedge_{\substack{k \in C \\ l \in f(k)}} \neg c_i^k \vee o_i^l \tag{7}$$

and the support

$$\bigwedge_{i \in \{1 \ldots n\}} \bigwedge_{l \in O} \left(\neg o_i^l \vee \bigvee_{k \in g(l)} c_i^k \right) \tag{8}$$

3 Evaluation

1 Describe evaluation

Introduce the tool written, give link to internet address. Explain the different combinations of clauses. Show two statistice on solved instances and give cactusplot of the different models.

create table for size and solving time. experiment with cactus plots

Give overview of size of the different models.

* Size does not always matter! * which is the best approach? * comparison to the solutions in the literature.

4 Conclusion and Future Work

6

We have shown a SAT encoding to the decision verison of the car sequencing problem. Our approach is still work in progress and we are improving our models and extending the emprirical and theoretical evaluation. In the following we identify our next steps and further future work.

Size of each cardinality constraint with the window restriction is $n * d_i$ for the ith option or car. Identify the precise level of consistency of the decomposition of the constraints, as in previous literture.

Alternative encodings, the rich space of formulations. Link to sorting networks, differen cardinliyt constraint formulations. Put eqmphasis on the analysis of auxilliary variables, Reusing cross-constraint usage is important. Tightening their definitions is the key to a good SAT encoding. Further redundant constraints that can be assesed in preprocessing.

Comparisong to different approaches from literture, IP and CP in an controlled environment. Focus on the decision version of the problem as state in benchmark. However, the greater part of the literature focues on the opt. Though Lower bounds are of theoretical interest, the practical need in this problem lies rather in finding good enough solutions rather the proving optimality of the solution. Due to the discretness of the problem for we can translate optimization functions to a sequence of decision problems. This will ease a fair comparison

The translations of gen-seq constraints to SAT through cumulative some is the best in our analysis. Try other benchmarks that are naturally defined through this constraint.

Analyse the encodings with Kullmanns tricks, that defines quality measures of CNF formulas. A theory of good SAT encodings and more concrete . Maybe results that lead to automatic preprocessing, that prevents one from common pitfull of bad SAT models.

Encodings in CNF can be seen as a form of Knowledge Compilation, and there is rich literture behind this paradigm. Into propositional theories. This field takes a more general appraoch to representation of knowledge and its different queries.

References

[Biere et al., 2009] Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors (2009). *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

[Davis and Putnam, 1960] Davis, M. and Putnam, H. (1960). A Computing Procedure for Quantification Theory. *J. ACM*, 7(3):201–215.

[Eén and Sörensson, 2003] Eén, N. and Sörensson, N. (2003). An Extensible SAT-solver. In *SAT*, pages 502–518.

[Estellon et al., 2006] Estellon, B., Gardi, F., and Nouioua, K. (2006). Large neighborhood improvements for solving car sequencing problems. *RAIRO - Operations Research*, 40(4):355–379.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[Gent, 1998] Gent, I. P. (1998). Two Results on Car-sequencing Problems. In *Report APES-02-1998*.

[Gent and Walsh, 1999] Gent, I. P. and Walsh, T. (1999). CSP_{LIB}: A Benchmark Library for Constraints. In *CP*, pages 480–481.

[Gottlieb et al., 2003] Gottlieb, J., Puchta, M., and Solnon, C. (2003). A Study of Greedy, Local Search, and Ant Colony Optimization Approaches for Car Sequencing Problems. In *EvoWorkshops*, pages 246–257.

[Gravel et al., 2005] Gravel, M., Gagné, C., and Price, W. L. (2005). Review and Comparison of Three Methods for the Solution of the Car Sequencing Problem. *The Journal of the Operational Research Society*, 56(11):1287–1295.

[Régin and Puget, 1997] Régin, J.-C. and Puget, J.-F. (1997). A Filtering Algorithm for Global Sequencing Constraints. In *CP*, pages 32–46.

[Siala et al., 2012] Siala, M., Hebrard, E., and Huguet, M.-J. (2012). An Optimal Arc Consistency Algorithm for a Chain of Atmost Constraints with Cardinality. In *CP*, pages 55–69.

[Tamura et al., 2009] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M. (2009). Compiling finite linear CSP into SAT. *Constraints*, 14(2):254–272.

[Tseitin, 1983] Tseitin, G. (1983). On the Complexity of Derivation in Propositional Calculus. *Automation of Reasoning: Classical Papers in Computational Logic*, 2:466–483.