Telecommunications Feature Subscription as a Partial Order Constraint Problem

Michael Codish, Vitaly Lagoon[2], and Peter J. Stuckey[2,3]

[1] Department of Computer Science, Ben-Gurion University, Israel
[2] Department of Computer Science and Software Engineering
The University of Melbourne, Australia
[3] National ICT Australia, Victoria Laboratory
mcodish@cs.bgu.ac.il, {lagoon,pjs}@cs.mu.oz.au

Abstract. This paper describes the application of a partial order constraint solver to a telecommunications feature subscription configuration problem. Partial order constraints are encoded to propositional logic and solved using a state-of-the-art Boolean satisfaction solver. The encoding is based on a symbol-based approach: symbols are viewed as variables which take integer values and are interpreted as indices in the order. Experimental evaluation indicates that partial order constraints are a viable alternative to previous solutions which apply constraint programming techniques and integer linear programming.

1 Introduction

Modern telecommunications providers enable a customer to subscribe to services selected from a catalog of features. The configuration of a feature subscription is often personalized based on preferences provided by the customer and constraints imposed by the provider to prevent undesirable feature interactions at run-time. When the subscription requested by a user is inconsistent, one problem is to find an optimal relaxation which is consistent.

In recent research, described in [4], the authors formalize the telecommunications feature subscription configuration problem and prove that the complexity of finding an optimal relaxation is NP-hard. That paper compares three techniques to address the problem using: constraint programming, SAT encoding, and integer linear programming. The authors conclude that the constraint programming approach is able to scale well compared to the other approaches.

This paper reexamines the encoding of telecommunications feature subscription configuration problem to SAT. Our approach to the encoding leads to a scalable solution which is considerably faster than the three techniques reported in [4]. Our approach is based on an encoding of partial order constraints into propositional logic and using the same implementation described in [2]. Partial order constraints are just like usual propositional formulae except that propositions involve also statements about a partial order on a finite set of symbols.

The encoding considered in [4] is atom-based. It models an atom of the form $(f > g)$ (which may be interpreted as f is after g) as a propositional variable. Then, propositional statements are added to encode the axioms of partial orders which the atoms are subject to. For a partial order constraint on n symbols, such

encodings typically introduce $O(n^2)$ propositional variables and involve $O(n^3)$ propositional connectives to express the axioms. In contrast, the proposal in [2] takes a symbol-based approach modeling the symbols in a partial order constraint as integer values (in binary representation). For n symbols this requires $k = \lceil \log_2 n \rceil$ propositional variables for each symbol. The integer value of a symbol reflects its index in a total order extending the partial order. Constraints of the form $(f > g)$ are then straightforward to encode in k-bit arithmetic and involve $O(\log n)$ connectives each.

2 Problem Statement

This section presents the formal statement of the telecommunications feature subscription configuration problem and is taken (with slight modification) from [4].

Let F denote a finite set of features. For $f_i, f_j \in F$ a *precedence constraint* $(f_i{>}f_j)$ indicates that f_i is after f_j. An *exclusion constraint* $(f_i{<>}f_j)$ between f_i and f_j indicates that f_i and f_j cannot appear together in a sequence of features, and is equivalent to the pair $(f_i{>}f_j)$, $(f_j{>}f_i)$. A *catalog* is a pair $\langle F, P\rangle$ with F a set of features and P a set of precedence constraints on F. A *feature subscription* S of a catalog $\langle F_c, P_c\rangle$ is a tuple $\langle F, C, U, W_F, W_U\rangle$ where $F \subseteq F_c$ is the set of features selected from F_c, C is the projection of P_c on F, U is a set of user defined precedence constraints on F, and $W_F{:}F \rightarrow \mathcal{N}$ and $W_U{:}U \rightarrow \mathcal{N}$ are maps which assign weights to features and user precedence constraints. The value of S is defined by $Value(S) = \Sigma_{f\in F} W_F(f) + \Sigma_{p\in U} W_U(p)$. The weight associated with a feature or a precedence constraint signifies its importance for the user.

A feature subscription $\langle F, C, U, W_F, W_U\rangle$ is *consistent* iff the directed graph $\langle F, C{\cup}U\rangle$ is acyclic. Checking for consistency is straightforward using topological sort as described in [4]. If a feature subscription is inconsistent then the task is to relax it and to generate a consistent one with maximum value. A *relaxation* of a feature subscription $S = \langle F, C, U, W_F, W_U\rangle$ is a consistent subscription $S' = \langle F', C', U', W_{F'}, W_{U'}\rangle$ such that $F' \subseteq F$, C' is the projection of C on F', U' is a subset of the projection of U on F', $W_{F'}$ is the restriction of W_F to F', and $W_{U'}$ is the restriction of W_U to U'. We say that S' is an optimal relaxation of S if there does not exist another relaxation S'' of S such that $Value(S'') > Value(S')$. In [4], the authors prove that finding an optimal relaxation of a feature subscription is NP-hard. This is the problem addressed in this paper.

3 Partial Order Constraints

Partial order constraints were introduced in [2]. Informally, a partial order constraint is just like a formula in propositional logic except that statements may involve propositional variables as well as atoms of the form $(f > g)$ where f and g are symbols.[4]

The semantics of a partial order constraint is a set of solutions. A solution is an assignment of truth values to propositional variables and atoms which is

[4] For brevity of presentation, we omit here atoms of the form $(f{=}g)$ considered in [2].

required to satisfy both parts of the formula: the "propositional part" and the "partial order part". Namely, if φ is a partial order constraint and μ a truth assignment, then μ is a solution for φ if: it satisfies φ when viewing the atoms as propositional variables, μ does not map an atom of the form $(f > f)$ to true, and if μ maps both $(f > g)$ and $(g > h)$ to true then also μ maps $(f > h)$ to true.

We are concerned with the question of satisfiability of partial order constraints: given a partial order constraint φ does it have a solution? Similarly to the general SAT problem, the satisfiability of partial order constraints is NP-complete, and the reduction from SAT is straightforward.

The following definition from [2] introduces the integer-based interpretation of partial order constraints. Let φ be a partial order constraint on propositional variables $\mathcal{B}$ and symbols $\mathcal{F}$ and let $|\mathcal{F}| = n$. An integer *assignment* for φ is a mapping μ which maps propositional variables from $\mathcal{B}$ to truth values $\{0, 1\}$ and symbols from $\mathcal{F}$ to values from $\{0, \ldots, n-1\}$. We say that μ is an integer *solution* of φ if it makes φ true under the standard interpretation of $>$ on the non-negative integers. In [2], the authors prove that a partial order constraint is satisfiable if and only if it has an integer solution.

To check the satisfiability of partial order constraints we apply an encoding to propositional logic. A partial order constraint φ on a set of propositional variables $\mathcal{B}$ and symbols $\mathcal{F}$ is encoded by a propositional formula φ' such that each solution of φ corresponds to a model of φ' and in particular such that φ is satisfiable if and only if φ' is. The idea is to construct the encoding in terms of the integer-based interpretation of partial order constraints. We view the n symbols in $\mathcal{F}$ as integer variables taking finite domain values from the set $\{0, \ldots, n-1\}$. Each symbol is thus modeled using $k = \lceil \log_2 n \rceil$ propositional variables which encode the binary representation of its value. Constraints of the form $(f > g)$ on $\mathcal{F}$ are interpreted as constraints on integers and it is straightforward to encode them in k-bit arithmetic.

The experiments described in [2] apply a partial order constraint solver written in SWI-Prolog [5] which interfaces the MiniSat solver [3] for solving SAT instances as described in [1]. This paper makes use of the same constraint solver.

4 The Encoding

Let $S = \langle F, C, U, W_F, W_U \rangle$ be a subscription of a catalog $\langle F_c, P_c \rangle$. We seek an optimal relaxation $S' = \langle F', C', U', W_{F'}, W_{U'} \rangle$. With each feature $f \in F$ we associate a propositional variable b_f indicating if f is included in F'. With each constraint $p \in C$ (and $p \in U$) we associate a propositional variable b_p to indicate if p is in C' (or in U').

For each constraint $p = (f > g)$ in C, f and g occur in the relaxation F' iff p occurs in the relaxation C'. Hence we introduce the propositional constraint

$$b_f \wedge b_g \leftrightarrow b_p \tag{1}$$

For each constraint $p = (f > g)$ in U, p may occur in the relaxation U' if f and g occur in F'. Hence we introduce the propositional constraint

$$b_f \wedge b_g \leftarrow b_p \tag{2}$$

For each constraint $p = (f > g)$ in $C \cup U$, if p occurs in the relaxation (C' or U') then the corresponding partial order constraint must hold

$$b_p \rightarrow (f > g) \tag{3}$$

Solving the partial order constraint obtained as the conjunction of the above equations (1), (2), and (3) assigns truth values to the propositional variables b_f and b_p indicating a consistent relaxation. To obtain an optimal relaxation we need an additional step. Let φ be the encoding of the above constraints (1) – (3) to a propositional formula.

With each of the propositional variables b_f and b_p, which indicate if feature f and constraint p are selected to appear in the relaxation, we associate a (integer value) weight w_f and w_p. To simplify presentation, consider the multiset $Bits$ of propositional variables which contains w_f occurrences of b_f for each $f \in F$ and w_p occurrences of b_p for each $p \in U$. Let ψ be the propositional formula which specifies that the sum of these $Bits$ is the binary number with digits $Sum = \{s_1, \ldots, s_k\}$.

The Prolog interface to MiniSat described in [1] offers the functionality `maximize(Vector,Cnf)` which seeks a satisfying assignment for the conjunctive normal form `Cnf` which maximizes the binary number `Vector`. If `Cnf` is unsatisfiable this call fails. This functionality is implemented by successively determining the bits in the `Vector`. This involves a call to the underlying SAT solver for each bit. If the `Vector` represents the sum of n bits then this involves $\log n$ calls to the SAT solver.

For the telecommunications feature subscription configuration problem, this functionality is applied taking the conjunctive normal form of $\varphi \wedge \psi$ (`Cnf`) and the sum bits $Sum = \{s_1, \ldots, s_k\}$ (`Vector`). In the actual encoding, the formula representing the Sum is constructed by summing the pseudo Boolean formula $b_f * w_f$ and $b_p * w_p$ and not as described above.

5 Experimental results

The experimentation is based on a collection of random catalogs and feature subscriptions obtained following the guidelines described in [4].

Table 1 describes the experiments for random catalogs with 50 features and 250 (750) precedence constraints (involving $\{<, >\}$). Each row labeled by $\langle f, p \rangle$ specifies a random subscription with f features and p user precedence constraints with weights selected between 1 and 4. Times are measured in seconds. The column marked `pocsp` corresponds to our Prolog implementation of partial order constraints built on top of MiniSat (average times over 10 random instances[5]). The columns marked `pwmsat`, `cplex` and `cp` are the times taken from Tables 2 and 3 of [4] for their: SAT encoding[6], ILP solver and CP solver. Note that the

[5] The precise instances used may be found at `http://www.cs.bgu.ac.il/~mcodish/Papers/Pages/feature_subscription.html`

[6] The authors of [4] use the SAT4J solver - `http://www.sat4j.org/`.

Table 1. Timings (sec) for two catalog sizes

	catalog $\langle 50, 500, \{<, >, <>\}\rangle$				catalog $\langle 50, 750, \{<, >\}\rangle$			
$\langle f, p\rangle$	`pocsp`	`pwmsat`	`cplex`	`cp`	`pocsp`	`pwmsat`	`cplex`	`cp`
$\langle 30, 20\rangle$	0.18	6.40	1.02	0.65	0.44	5.03	18.46	2.38
$\langle 35, 35\rangle$	0.47	23.95	22.76	7.43	2.30	18.28	126.35	12.88
$\langle 40, 40\rangle$	1.08	282.76	247.14	67.80	6.37	92.11	514.27	42.27
$\langle 45, 90\rangle$	39.36	12638.25	7690.90	1115.51	105.51	2443.23	3780.54	188.83
$\langle 50, 4\rangle$	0.39	195.72	1010.38	413.61	1.00	319.53	3162.08	342.49

random instances for `pocsp` are most likely different than those applied in [4]. The machines are also different. Theirs is a PC Pentium 4 (CPU 1.8 GHz and 768 MB of RAM). Ours is a PC Pentium 4 (CPU 2.4 GHz and 512 MB of RAM). Ours is running SWI Prolog under Linux, kernel 2.6.11-6. With no intention to compare the two machines, the timings are clear enough.

6 Conclusions

Our encoding indicates the clear benefit in choosing the right tool for the problem at hand: Once stating the feature subscription configuration problem as one of partial order constraints, the solution is straightforward. Our results indicate that the application of partial order constraints to the telecommunications feature subscription configuration problem provides a viable alternative to other solution techniques. The partial order approach improves upon the previous SAT approach by avoiding the $O(n^3)$ encoding of partial order, and over the other approaches because of the nogood learning capabilities of SAT.

References

1. M. Codish, V. Lagoon, and P. J. Stuckey. Logic programming with satisfiability. *TPLP*, 8(1):121–128, 2008.
2. M. Codish, V. Lagoon, and P. J. Stuckey. Solving partial order constraints for lpo termination. *Journal on Satisfiability, Boolean Modeling and Computation*, 5:193–215, 2008. An earlier version appears in the proceedings of RTA 2006, LNCS 4098.
3. N. Eén and N. Sörensson. An extensible SAT-solver. In E. Giunchiglia and A. Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003 (Selected Revised Papers)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.
4. D. Lesaint, D. Mehta, B. O'Sullivan, L. Quesada, and N. Wilson. Solving a telecommunications feature subscription configuration problem. In *Proceedings of CP*, 2008. forthcoming (an earlier version appears in the Proceedings of Innovative Applications of Artificial Intelligence (July 2008).
5. J. Wielemaker. An overview of the SWI-Prolog programming environment. In F. Mesnard and A. Serebenik, editors, *Proceedings of the 13th International Workshop on Logic Programming Environments*, pages 1–16, Heverlee, Belgium, Dec. 2003. Katholieke Universiteit Leuven. CW 371.