Finding MAPs for Belief Networks using Rule-based Constraint Programming

Harriet Beaver and Ilkka Niemelä
Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science
P.O.Box 5400, FIN-02015 HUT, Finland
{Harriet.Beaver,Ilkka.Niemela}@hut.fi

Abstract

The paper studies the question of solving the maximum a-posterior (MAP) assignment problem on a Bayesian belief network using rule-based constraint programming techniques. The starting point is a translation of MAP problems into cost-based abduction presented by Abdelbar. Cost-based abduction can be mapped to a rule-based constraint satisfaction problem. This provides the first approach to solving MAP problems using rule-based techniques. Using rules it is possible to represent also other kinds of constraints than only abductive ones. This opportunity is investigated and translations incorporating symbolic constraints are devised. The computational properties of the translations are studied using an efficient implementation of rule-based constraint programming, the **smodels** system, and two standard examples from belief network literature.

1 Introduction

A Bayesian belief network represents graphically a probability distribution P on a finite set of random variables V [Pearl, 1988]. The network consists of a directed acyclic graph (DAG) $D = (V, E)$, which is a minimal independency map of P, and a set of conditional probability distributions $\{P_v | v \in V\}$. The nodes in D represent the variables V, the edges represent direct causal influence between them, and the conditional probabilities express the strengths of these influences. Each P_v specifies the conditional probabilities of all possible assignments for v given all possible assignments for its parents $\pi(v)$, and thus consists of assignments l for v and $\pi(v)$, and the corresponding probabilities $p(l)$. For a DAG to be a minimal independency map of P, every variable $v \in V$ must be conditionally independent, given its parents $\pi(v)$, of all its nondecendents, and $\pi(v)$ must be the least set for which this is true. Then the probability of an assignment $v_1 = V_1, v_2 = V_2, ..., v_n = V_n$ is

$$P(V_1, V_2, ..., V_n) = \prod_{i=1}^{n} P(V_i \mid \pi(V_i)).$$

The MAP problem is to find, given a belief network $B = (V, E, P)$ and evidence $\mathcal{E}$, an assignment $\mathcal{A}$ for B which maximizes the conditional probability $P(\mathcal{A}|\mathcal{E})$. The evidence is a partial assignment for B representing observations.

In this paper we investigate the possibility of solving MAP problems using an efficient rule-based constraint programming system, **smodels** [Niemelä and Simons, 1996], based on logic programs with the stable model semantics. The approach has provided encouraging results in many application areas, such as planning [Dimopoulos *et al.*, 1997], verification of distributed systems [Heljanko, 1999] and constraint satisfaction [Niemelä, 1998]. Hence, it is interesting to see whether such a system is suitable for solving MAP problems as well. Our starting point is a translation of MAP problems to cost-based abduction presented in [Abdelbar, 1998]. It is straightforward to map abduction to a rule-based constraint satisfaction problem solvable by **smodels**. However, rule-based constraint programming offers a richer language than cost-based abduction for representing symbolic constraints [Niemelä, 1998]. We investigate the possibility of incorporating more symbolic constraints in order to devise computationally more efficient translations from MAPs to rules.

The structure of the rest of the paper is as follows. In the next section we describe the translation of MAP to cost-based abduction in [Abdelbar, 1998]. Then we introduce the rule-based constraint programming approach supported by the **smodels** system. In Section 4 we develop three approaches to translating MAPs to rules and in Section 5 we report some experimental results of using the translations.

2 Translating MAP to Cost-based Abduction

Cost-based abduction is a process of finding the best explanation for a set of observed facts, the evidence, by finding a minimal cost proof for the facts. The cost of a proof is computed by adding the costs of the propositions that must be assumed in order to complete the proof.

Definition 2.1 A cost-based abduction problem is a 4-tuple $(\mathcal{H}, \mathcal{R}, c, \mathcal{E})$, where:

- $\mathcal{H}$ is a set of hypotheses
- $\mathcal{R}$ is a set of implications of the form $(h_1 \wedge h_2 \wedge ... \wedge h_n) \rightarrow h_{n+1}$, where $\{h_1, h_2, ..., h_{n+1}\} \subseteq \mathcal{H}$
- $c : \mathcal{H} \rightarrow \mathbb{R}_+$ is the cost function
- $\mathcal{E} \subseteq \mathcal{H}$ is the evidence

A solution to the problem is a set of hypotheses $\mathcal{S} \subseteq \mathcal{H}$ such that $\mathcal{S} \bigcup \mathcal{R} \models \mathcal{E}$. The cost of a solution is the sum of the costs of the hypotheses in the solution: $c(\mathcal{S}) = \sum_{h_i \in \mathcal{S}} c(h_i)$.

Abdelbar [Abdelbar, 1998] presents the following way to transform a MAP problem on a Bayesian belief network to a cost-based abduction problem.

Definition 2.2 Given a Bayesian belief network $B = (V, E, P)$ and evidence $\mathcal{E}$, the cost-based abduction problem corresponding to the MAP problem is a 4-tuple $(\mathcal{H}, \mathcal{R}, c, \mathcal{G})_{MAP}$, where:

- $\mathcal{H} = \mathcal{H}_{v,t} \bigcup \mathcal{H}_v \bigcup \mathcal{H}_l$, where:
  - $\mathcal{H}_{v,t} = \{h_{v,t} | \ v \in V, t \in D(v)\}$ where $h_{v,t}$ is the hypothesis that v is assigned the value t.
  - $\mathcal{H}_v = \{h_v | \ v \in V\}$ where h_v is the hypothesis that v is assigned some value.
  - $\mathcal{H}_l = \{h_l | \ l \in P_v, v \in V\}$ where h_l is the hypothesis that v and all its parents $u \in \pi(v)$ are assigned values according to l.
- $\mathcal{R} = \mathcal{R}_{v,t} \bigcup \mathcal{R}_l$, where:
  - $\mathcal{R}_{v,t} = \{ \ h_{v,t} \rightarrow \ h_v \mid v \in V, t \in D(v)\}$
  - $\mathcal{R}_l = \{R_l | \ l \in P_v, v \in V\}$, where R_l is the implication $h_l \wedge h_{u_1,t_1} \wedge h_{u_2,t_2} \wedge ... \wedge h_{u_n,t_n} \rightarrow \ h_{v,t}$ with t as the value assigned to v and $t_1, ..., t_n$ the values assigned to the parents $u_1, u_2, ..., u_n$ of v, in l.
- $c : \mathcal{H} \rightarrow \mathbb{R}_+$ such that
  - $c(h_{v,t}) = \infty$ for each $h_{v,t} \in \mathcal{H}_{v,t}$
  - $c(h_v) = \infty$ for each $h_v \in \mathcal{H}_v$
  - $c(h_l) = -\ln p(l) + Q$ for each $h_l \in \mathcal{H}_l$, and the constant Q is chosen so that $Q \geq -ln \prod_v min\{p(l) | l \in P_v\}$.
- $\mathcal{G} = \mathcal{H}_v \bigcup \{h_{v,t} | (v,t) \in \mathcal{E}\}$

Let $(\mathcal{H}, \mathcal{R}, c, \mathcal{G})_{MAP}$ be a cost-based abduction problem generated from $B = (V, E, P)$ and $\mathcal{E}$. If $\mathcal{S}$ is a minimal-cost solution to $(\mathcal{H}, \mathcal{R}, c, \mathcal{G})_{MAP}$, then $\mathcal{S}$ induces a consistent assignment $\mathcal{A}$ for B, such that $\mathcal{E} \subseteq \mathcal{A}$, and $\mathcal{A}$ is a MAP assignment for B. Also, if $\mathcal{A}$ is a MAP assignment for B and $\mathcal{E}$, then there is a minimal-cost solution $\mathcal{S}$ to $(\mathcal{H}, \mathcal{R}, c, \mathcal{G})_{MAP}$ such that $\mathcal{S}$ induces $\mathcal{A}$ [Abdelbar, 1998].

The fact that $\mathcal{A}$ is a full assignment comes from including $\mathcal{H}_v$ in $\mathcal{G}$, and the consistency of $\mathcal{A}$ is obtained by giving the constant Q in the cost-function a relatively large value. In that way, a solution which induces two values for any one variable will always be more expensive than the solutions which induce consistent assignments.

3 Rule-based Constraint Programming

Logic programs with the stable model semantics offer a novel interesting constraint programming paradigm [Niemelä, 1998]. The idea is to bring advantages of logic programming based knowledge representation techniques to constraint programming. The underlying idea in this paradigm is to interpret the rules of a program as constraints on a solution set for the program. A solution set is a set of atoms, and a normal logic program rule of the form

$$A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \qquad (1)$$

is seen as a constraint on this set stating that if $B_1, \ldots, B_m$ are in the solution set and none of $C_1, \ldots, C_n$ are included, then A must be included in the set. A very natural definition for the solution sets is provided by *stable models* [Gelfond and Lifschitz, 1988].

For the purpose of this paper it is enough to explain the approach in the ground (propositional) case, where the stable models are defined as follows. The *reduct* P^S of a program P with respect to a set of atoms S is the (definite) program obtained from P by deleting

(i) each rule that has a negative literal not C in its body with $C \in S$ and

(ii) all negative literals in the bodies of the remaining rules.

The reduct P^S provides the set of potentially applicable rules given a model candidate S, i.e., the rules where the negative body literals are satisfied by S. We can see the reduct P^S as a set of implications and denote by $\mathrm{Cl}(P^S)$ the set of atomic consequences of P^S. Note that $\mathrm{Cl}(P^S)$ can be constructed simply by forward-chaining P^S. A stable model is a set S of atoms that are consequences of the potentially applicable rules P^S.

Definition 3.1 *Let P be a ground (variable-free) program. Then a set of ground atoms S is a* stable model *of P iff $S = \mathrm{Cl}(P^S)$.*

Example 1 *Program P*

$$\begin{aligned} p &\leftarrow \text{not } q, r \\ q &\leftarrow \text{not } p \\ r &\leftarrow \text{not } s \\ s &\leftarrow \text{not } p \end{aligned}$$

has a stable model $S = \{r, p\}$ because the reduct P^S of P with respect to S is

$$\begin{aligned} p &\leftarrow r \\ r &\leftarrow \end{aligned}$$

and $S = \mathrm{Cl}(P^S)$. For instance, $S' = \{p, s\}$ is not a stable model of P because the reduct $P^{S'}$ is $\{p \leftarrow r\}$ and $\mathrm{Cl}(P^{S'}) = \{\} \neq S'$. In fact, P has one other stable model $\{s, q\}$.

The **smodels** system [Niemelä and Simons, 1996] is an implementation supporting this paradigm. Given a program, **smodels** computes all or a given number of stable models of the program or determines that the program has no stable models. It supports programs that extend normal programs considerably. In this paper we employ a couple of these extended features:

- A *compute statement* of the form

$$compute\ \{B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n\}$$

  has the effect is that **smodels** searches only for models satisfying the given set of literals.

- An *integrity constraint*

$$\leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$$

  has the effect of eliminating all stable models satisfying the body of the constraint.

- A *choice rule* of the form

$$n\{A_1, \ldots, A_l\}m \leftarrow B_1, \ldots, B_k$$

  forces a subset A of $\{A_1, \ldots, A_l\}$ to any stable model such that $n \leq |A| \leq m$ (i.e., the cardinality of the subset is between n and m) whenever the body of the rule holds in the model.

- A *cardinality constraint rule* of the form

$$A \leftarrow k\{B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n\}$$

  includes A in any stable model where the number B_is in the model and C_js not in the model is together at least k.

- *Optimize statements* provide a form of optimization in **smodels**: one can assign weights to literals and use a minimize (maximize) statement of the form

$$minimize\ \{B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n\}$$

  to make the system search for a model with the minimal (maximal) weight, where the weight of a model is the sum of the weights of those literals from $\{B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n\}$ which are satisfied by the model.

  In order to avoid complications arising from finite precision of real arithmetic the current implementation supports only integer weights. This means that real-valued weights have to be scaled to integers.

Example 2 *It is straightforward to translate a cost-based abduction problem to a stable model finding problem with a minimization statement in the following way. Given a cost-based abduction problem we build a program* P *such that it includes all rules* $h_{n+1} \leftarrow h_1, \ldots, h_n$*, of the abduction problem, a generate rule* $0\{h_1, \ldots, h_m\} \leftarrow$ *where* $\{h_1, \ldots, h_m\}$ *are all the hypotheses together with a compute and optimize statement*

$$compute\ \{e_1, \ldots, e_l\}$$
$$minimize\ \{h_1, \ldots, h_m\}$$

where $\{e_1, \ldots, e_l\}$ *is the evidence, and costs assigned to each* h_i *as in the abduction problem.*

4 Translating MAP to Rule-based Constraints

Our first translation, denoted by T0, maps a cost-based abduction problem $(\mathcal{H}, \mathcal{R}, c, \mathcal{G})_{MAP}$ to a stable model finding problem as described in Example 2. We take the set of hypotheses $\mathcal{H}$ in $(\mathcal{H}, \mathcal{R}, c, \mathcal{G})_{MAP}$ to be atoms, and the set of rules $\mathcal{R}$ to be a part of the program. In a cost-based abduction problem the hypotheses in $\mathcal{H}_{v,t} \bigcup \mathcal{H}_v$ were given infinite costs to make sure that they are not assumed in any finite-cost solution. To simplify, we give the corresponding atoms no weights, but allow only atoms corresponding to the hypotheses in $\mathcal{H}_l$ to be assumed. The weights of these atoms are the costs of the corresponding hypothesis. They are converted to integer weights supported by **smodels** by multiplying them by a suitable constant in order to obtain sufficient accuracy and then truncating to an integer. The program consists of the following rules and statements:

- The rules $\mathcal{R}$.
- The choice rule $0\{h_{l1}, h_{l2}, ..., h_{ln}\} \leftarrow$, where $\{h_{l1}, h_{l2}, ..., h_{ln}\} = \mathcal{H}_l$. This rule generates a subset of atoms from $\mathcal{H}_l$, to be included in a stable model.
- The statement $minimize\ \{h_{l1}, h_{l2}, ..., h_{ln}\}$, where $\{h_{l1}, h_{l2}, ..., h_{ln}\} = \mathcal{H}_l$. It is satisfied only by the stable models with the minimal sum of the weights of the atoms h_l included in the model.
- The statement $compute\ \{g_1, g_2, ..., g_n\}$, where $\{g_1, g_2, ..., g_n\} = \mathcal{G}$. It ensures that every stable model contains the evidence, and at least one atom $h_{v,t}$ for every $v \in V$.

A stable model satisfying these rules, constructed from a Bayesian belief network B and evidence $\mathcal{E}$, induces an assignment $\mathcal{A}$ for B, by assigning t to v iff it contains the atom $h_{v,t}$. The assignment $\mathcal{A}$ is MAP.

We now develop three alternative ways of translating MAPs into stable model finding problems. In the cost-based abduction problem, and in the previous translation into a stable model finding problem, the consistency of the assignment was a result of the minimization. Another way to ensure the consistency is simply to add the rules:

$$\leftarrow 2\{h_{v,t1}, h_{v,t2}, ..., h_{v,tn}\}$$

where $\{t1, t2, ..., tn\} = D(v)$, for every variable $v \in V$. The rule states that no stable model can contain two atoms of the form $h_{v,t}$. We can now omit the constant Q from the weight function. These are the only alterations for our second translation, which we denote by T1.

In the second alternative translation, denoted by T2, we try generating atoms from $\mathcal{H}_{v,t}$ instead of from $\mathcal{H}_l$. The assignments are then compared using the optimize statement, and the stable model corresponding to the minimal assignment is chosen. The program consists of the following rules and an optimize statement which is the same as in T0 and in T1.

- A choice rule for every variable v

$$1\{h_{v,t1}, h_{v,t2}, ..., h_{v,tn}\}1 \leftarrow$$

Transl.	`alarm`	`Boe92`
T0	0.03s	> 30min
T1	0.03s	1.1s
T2	0.03s	3.1s
T3	0.03s	2.2s

Table 1: Solution times using different translations

where $\{t1, t2, ..., tn\} = D(v)$ stating that every stable model must contain exactly one atom from the set $\{h_{v,t} | t \in D(v)\}$.

- The basic rules for every $v \in V$ and every $l \in P_v$

$$h_l \leftarrow h_{u_1,t_1}, h_{u_2,t_2}, ..., h_{u_n,t_n}, h_{v,t}$$

where t is the value assigned to v, and $t_1, ..., t_n$ those assigned to the parents $u_1, u_2, ..., u_n$ of v, in l.

- The compute statement: *compute* $\{e_1, e_2, ..., e_n\}$ where $\{e_1, e_2, ..., e_n\} = \mathcal{E}$ stating that every stable model must contain the evidence.

The translation T3 tries to make use of the structure of the DAG by assigning a value to a node only after assigning values to its parents. The program consists of the following rules, and the same optimize statements and compute statements as T2.

- A choice rule for every variable $v \in V$ and every possible assignment for its parents $\pi(v) = \{u1, u2, ..., um\}$

$$1\{h_{l1}, h_{l2}, ..., h_{ln}\}1 \leftarrow h_{u1,t1}, h_{u2,t2}, ..., h_{um,tm}$$

where $\{l1, l2, ..., ln\} \subseteq P_v$ is the set of tuples which agree with the assignment for $\{u_1, u_2, ..., u_m\}$.

- The basic rule: $h_{v,t} \leftarrow h_l$, where $(v, t) \in l$, for every $v \in V$ and every $l \in P_v$.

5 Experimental Results

In order to understand better the computational properties of translations T0–T3 from MAP finding to rule-based constraint satisfaction we experimented with two Bayesian networks: `alarm` is the alarm example from [Russel and Norvig, 1995, pp. 437–439] having five nodes and `Boe92` is the network Boerlage92 from http://www.norsys.com/networklibrary.html with 23 nodes. Table 1 contains the solution time for finding MAP for the two examples using the four translations. The tests were run on a Pentium II 450 MHz with 128MB of memory, Linux 2.2.4 operating system, `smodels` version 2.22 and `lparse` version 0.99.14.

6 Conclusions

We have studied the possibilities of solving MAP assignment problems on Bayesian belief networks using rule-based constraint programming. Our starting point was a translation of MAP problems to cost-based abduction presented by Abdelbar [Abdelbar, 1998] which leads straightforwardly to a rule-based constraint satisfaction problem. We showed that computational advantages were gained by employing more symbolic constraints when describing the MAP problem with rules. This indicates that it could be useful to be able to use together both probabilistic knowledge and symbolic constraints. The rule-based constraint programming paradigm supported by the **smodels** system seems to offer one interesting framework where this kind of merging of paradigms is already possible.

Acknowledgements

The work has been funded by the Academy of Finland through Project 43963.

References

[Abdelbar, 1998] A.M. Abdelbar. An algorithm for finding maps for belief networks through cost-based abduction. *Artificial Intelligence*, 104:331–338, 1998.

[Dimopoulos *et al.*, 1997] Y. Dimopoulos, B. Nebel, and J. Koehler. Encoding planning problems in nonmonotonic logic programs. In *Proc. of the Fourth European Conference on Planning*, pages 169–181, Toulouse, France, September 1997. Springer-Verlag.

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of the 5th International Conference on Logic Programming*, pages 1070–1080, Seattle, USA, August 1988. The MIT Press.

[Heljanko, 1999] K. Heljanko. Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets. In *Proc. of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 240–254, Amsterdam, the Netherlands, March 1999. Springer-Verlag.

[Niemelä and Simons, 1996] I. Niemelä and P. Simons. Efficient implementation of the well-founded and stable model semantics. In M. Maher, editor, *Proc. of the Joint International Conference and Symposium on Logic Programming*, pages 289–303, Bonn, Germany, September 1996. The MIT Press.

[Niemelä, 1998] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. In *Proc. of the Workshop on Computational Aspects of Nonmonotonic Reasoning*, pages 72–79. Available at http://www.tcs.hut.fi/pub/reports/A52abstract.html.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[Russel and Norvig, 1995] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.