Finding MAPs Using High Order Recurrent Networks

Emad A.M. Andrews and Anthony J. Bonner

Department of Computer Science
University of Toronto
Toronto, ON
emad@cs.toronto.edu

Abstract. Belief revision is the problem of finding the most plausible explanation for an observed set of evidences. This has many applications in various scientific domains like natural language understanding, medical diagnosis and computational biology. Bayesian Networks (BN) is an important probabilistic graphical formalism used widely for belief revision tasks. In BN, belief revision can be achieved by setting the values of all random variables such that their joint probability is maximized. This assignment is called *the maximum a posteriori* (MAP) assignment. Finding MAP is an NP-Hard problem. In this paper, we are proposing finding the MAP assignment in BN using High Order Recurrent Neural Networks through an intermediate representation of Cost-Based Abduction. This method will eliminate the need to explicitly construct the energy function in two steps, objective and constraints, which will decrease the number of free parameters to set.

1 Introduction

Belief revision is the problem of finding the most plausible explanation for an observed set of evidences. Belief revision falls under the broader domain of reasoning under uncertainty where information is not complete or contradictory; thus, probabilistic handling seemed the best candidate for those tasks. However, probabilistic reasoning was described as being "*epistemologically inadequate*" by McCarthy and Hayes in their basic paper in 1969 [1]. They showed that the number of parameters needed to compute the joint probability distribution is exponentially proportional to the size of the given dataset which yields the whole mathematical computations intractable. As a result, researchers avoided using probabilistic reasoning until the notion of independence assumption appeared.

In 1988, Pearl standardized the independence assumption notion and formally presented Bayesian Network (BN) where each variable is conditionally independent of its ancestors given its parents [2]. BN is fully specified by two components: a Directed Acyclic Graph (DAG), whose vertices represent random variables, and a set of parameters that describe the conditional probability distribution of each variable given its parents. These two components together fully specify a unique joint distribution over all random variables in the graph. Let *G* be a DAG, and let $X_1, \ldots, X_n$ denote the set of random variables, vertices of G. The BN encodes the *Markov assumption*: Each

C.S. Leung, M. Lee, and J.H. Chan (Eds.): ICONIP 2009, Part I, LNCS 5863, pp. 100–109, 2009.

variable is independent of all its non-descendants variables given its parents. Thus, the full joint distribution can be composed of the product form:

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i|\pi(X_i)) \tag{1}$$

$\pi(X_i)$ is the set of parents of X_i in G. For a specific assignment, A, over all nodes, (1) can be rewritten as:

$$P(A) = \prod_{i=1}^{n} P(A(X_i)|A(\pi(X_i))) \tag{2}$$

BN saves a considerable amount of memory and calculations which enables us to calculate joint distributions otherwise impossible to calculate. For example, to specify the full joint distribution for 10 binary random variables we need $2^{10} = 1024$ values to be stored and used during computation. If we used BN with each variable depending on no more than three other variables, we end up having $10\times2^3=80$ parameters only.

Given a BN with an observed set of evidence nodes $\mathcal{E}$ we are looking for values assignment *A* for the rest of the network nodes, such that $P(A|\mathcal{E})$ is maximized, using Bayes rule:

$$P(A \mid \mathcal{E}) = \frac{P(A)P(\mathcal{E} \mid A)}{P(\mathcal{E})} \tag{3}$$

Because we have observed the values of evidence nodes $\mathcal{E}$, $P(\mathcal{E})$ is constant, so it ends up maximizing *P(A)* that represents the joint probability distribution in (1) and (2). This assignment is called *the maximum a posteriori* assignment (MAP). Once this assignment is found, we can do all kinds of probabilistic inference needed.

Finding MAP is shown to be NP-hard [4]. For multiply-connected BN, existing algorithms suffer from exponential complexity, so new heuristics and algorithms are always needed. In this paper, we propose finding MAP using High Order Recurrent Neural Networks (HORN) through an intermediate representation of Cost-Based Abduction (CBA). We first transform BN into CBA system using the algorithm mentioned in [6]. To our knowledge, this is the only algorithm in literature that does such a transformation, so it is crucial to analyze and discuss it step by step. Then, we will fill in the gap between BN and HORN by solving the resultant CBA system using HORN through the method mentioned in [7]. Finally, we will provide the mathematical framework to derive the energy functions equivalent to logical rules with more than 3 hypotheses and present our results.

1.1 Cost-Based Abduction (CBA)

CBA was first introduced by Charniak et al [9]. Formally, a CBA system is a 4-tuple (*H*, *R*, *c*, *G*), where *H* is a set of hypotheses or propositions, *c* is a function from *H* to a nonnegative real *c(h)* called the assumability cost of $h \in H$, *R i*s a set of rules of

the form: $R:(p_{i_1} \wedge p_{i_2} \wedge ... \wedge p_{i_n}) \rightarrow p_{i_k}$ for all $p_{i_1}, ..., p_{i_n} \in H$, and $G \in H$ is the goal or the evidence set [6].

Objective: finding the least cost proof (LCP) of the goal. Proof cost is the sum of all costs of the hypotheses needed to be assumed to complete the proof. Any given hypothesis $p_i \in H$ can be true either by proving it or by assuming it to be true and paying its assumability cost. Hypotheses that can be assumed have assumability costs less than ∞, we call them *"assumables"*. Consequent hypotheses that are proven through the assumables are called *"provables"*.

Finding the optimal solution for a CBA system is proven to be NP-hard [11] [14]. Previous approaches to CBA can be found in [12] [13] [14]. The only Neural Networks (NN) approach to CBA was introduced in [7], where the authors found the optimal solution of CBA system by transforming it into HORN through an intermediate representation of Penalty Logic (PL).

Finding the LCP in CBA system is equivalent to finding the MAP in BN [11] [14]. Despite their equivalency, it is believed that finding LCP is more efficient than finding MAP and it may be easier to find heuristic for CBA system than finding one for BN [6] [11]. Santos found the necessary and sufficient conditions under which CBA is polynomially solvable [15]. On the other hand, polynomial solvability for finding MAP in BN is not available even with applying restrictions on the graphical representation [4] and even for trying to find an alternative next-best explanation [5].

1.2 High Order Recurrent Networks (HORN)

A recurrent NN is one whose underlying inter-neural connections contain at least one cycle. The Hopfield network is perhaps the most famous recurrent NN [16]. The underlying topology is a graph and each weighted connection is either a binary connection, T_{ij}, that connects two neurons (i, j) or a unary connection I_i which is the bias of a single neuron i. Each neuron is trying to minimize the energy function which is usually composed of two energy functions:

$$E = E^{Obj} + \beta E^{Const} \tag{4}$$

E^{Obj} describes the objective function to be either maximized or minimized while the E^{Const} ensures the feasibility of the optimized solution by enforcing the set of the constraints. β is a problem dependant free parameter to be experimentally tuned. We can think of β as a tradeoff knob between solution optimality and solution feasibility. Depending on the NN order, E can be either quadratic or higher order.

HORN is a recurrent network whose underlying topology is a hypergraph, allowing weighted hyperedges that connect more than two neurons. The degree of the edge is the number of neurons it connects. The order of the network is the highest degree in the topology. In a K^{th}-order HORN a neuron with an activation level u_i and an output V_i is governed by:

$$\frac{du_i}{dt} = \sum_{d=1}^{k} \sum_{s \in S_d, i \in s} T_s^{(d)} \prod_{j \in s, j \neq i} V_j \tag{5}$$

Where $V_i = g(u_i)$, and g is typically a sigmoidal activation function. k is the order of the network. $T_s^{(d)}$ is the weight of the d^{th}-degree edge connecting neurons $i_1 ... i_d$. S_d denotes the set of all neuron sequences $J_1, ..., J_d$, such that $1 \le j_{1,...,d} \le n$; where n is the number of neurons and $J_a \neq J_b$ if $a \neq b$ [3]. Each unit minimizes the following K^{th}-order energy function:

$$K = -\sum_{1 \le i_1 < i_2 < ... < i_k \le n} T_{i_1 ... i_k}^{(k)} V_{i_1} ... V_{i_k} - \sum_{1 \le i_1 < i_2 < ... < i_{k-1} \le n} T_{i_1 ... i_{k-1}}^{(k-1)} V_{i_1} ... V_{i_{k-1}} - ... - \sum_{1 \le i \le n} T^{(1)} V_i \tag{6}$$

1.3 Related Work

To our knowledge, the only attempt to find MAP using HORN is in [3]. However, this method requires deriving the energy function in two steps: E^{Obj} and E^{Const}; then, the two functions need to be combined into one function as in (4). This method requires extensive experimentation to set the network parameter β among other parameters [3]. In this work, we will create the full energy function in one step from the CBA system equivalent to the given BN. That will eliminate the need for creating and combining two energy functions and the need for tuning the free parameter β.

2 Transforming BN into CBA

In this section we will follow and analyze the linear time transformation algorithm mentioned in [6] to transform the multiply-connected BN in fig.1 into an equivalent CBA system. This example can be found in Murphy's BN tutorial [17].

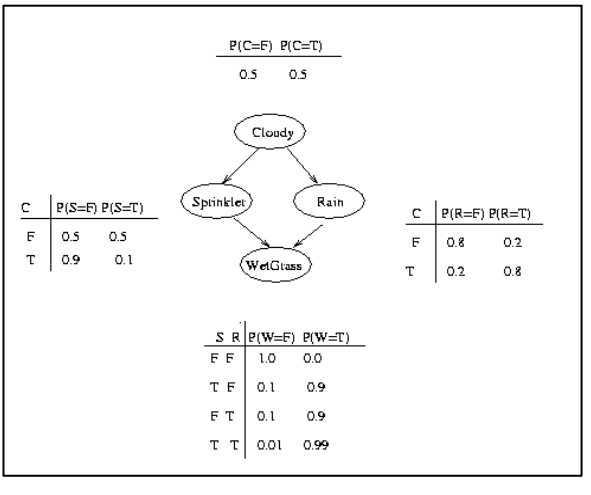

Fig. 1. Multiply-connected BN example

Our objective is to explain why the grass is wet. So we want to reach an assignment A for the random variables such that $P(A|W)$ is maximized. Using Bayesian inference we can see that:

$$P(S = 1 | W = 1) = 0.430$$

$$P(R = 1 | W = 1) = 0.708$$

So, the best explanation for the wet grass is because it is raining rather than because of having the sprinkler on.

The transformation to CBA goes as follows:

1. Transform the CPT of each random variable v to a linear table P_v, tables 1 to 4. Each line $l \in P_v$ is a hypothesis that the premises of that line are satisfied. *V(l)* denotes the probability corresponding to line *l* in the table. The cost of the hypothesis h_l that represents each line is : $c(h_l) = -\log V(l) + Q$ where: $Q = -\log \prod_{v \in V} Q_v$ and $Q_v = \min\{V(l) \mid l \in P_v\}$ So: $Q_C = 0.5, Q_S = 0.1, Q_R = 0.2, Q_W = 0.9$
When calculating Q, we ignore the line $P(W \mid S, R) = 0$ because *W* is our goal and we are trying to explain why the grass is wet, so $Q = 2.0458$

Table 1. Cloudy , C

Value	*P(C)*	h_l	$c(h_l)$
0	0.5	$h_C l_1$	2.3468
1	0.5	$h_C l_2$	2.3468

Table 2. Sprinkler , S

Value		$P(S \mid C)$	h_l	$c(h_l)$
C	S			
0	0	0.5	$h_S l_1$	2.3468
0	1	0.5	$h_S l_2$	2.3468
1	0	0.9	$h_S l_3$	2.0915
1	1	0.1	$h_S l_4$	3.0458

Table 3. Rain , R

Value		$P(R \mid C)$	h_l	$c(h_l)$
C	R			
0	0	0.8	$h_R l_1$	2.1427
0	1	0.2	$h_R l_2$	2.7447
1	0	0.2	$h_R l_3$	2.7447
1	1	0.8	$h_R l_4$	2.1427

Table 4. Wet grass, W. The goal

Value			$P(W \mid S,R)$	h_l	$c(h_l)$
S	R	W			
0	0	1	0.0	$h_W l_1$	∞
1	0	1	0.9	$h_W l_2$	2.0915
0	1	1	0.9	$h_W l_3$	2.0915
1	1	1	0.99	$h_W l_4$	2.0501

2. For every variable $v \in V$, create h_v representing that proposition *v* is assigned some truth value; $c(h_v) = \infty$. Result is a set of hypothesis $\{h_C, h_S, h_R, h_W\}$
3. For every $v \in V$ and for every $t \in D(v)$, where $D(v)$ is domain of *v,*
   - Construct a hypothesis h_{v_t} denoting that proposition *v* is assigned a value *t*. Add h_{v_t} to the system hypothesis and assign $c(h_{v_t}) = \infty$;

Result is a set of new hypothesis: $\{h_{C_t}, h_{C_f}, h_{S_t}, h_{S_f}, h_{R_t}, h_{R_f}, h_{W_t}\}$, we ignore h_{W_f}.

- Construct a rule R_{v_t} with $R^A_{v_t} = \{h_{v_t}\}$ and $R^C_{v_t} = \{h_v\}$

Where R^A refers to the set of *R's* antecedents and R^C refers to *R's* consequent. Result is this set of rules:

$$R_{C_t} : h_{C_t} \to h_C \;,\; R_{C_f} : h_{C_f} \to h_C; \quad R_{S_t} : h_{S_t} \to h_S \;,\; R_{S_f} : h_{S_f} \to h_S$$
$$R_{R_t} : h_{R_t} \to h_R \;, R_{R_f} : h_{R_f} \to h_R; \quad R_{W_t} : h_{W_t} \to h_W \;, R_{W_f} : h_{W_f} \to h_W$$

4. For every $v \in V$ and every $l \in P_v$:
   - Construct a rule R_l where: $R^A_l = \{h_l\}$
   - For every $\{u \to t'\} \subseteq l, where\, u \in \pi(v)\, and\, t' \in D(u)$, set $R^A_l = R^A_l \bigcup \{h_{u_{t'}}\}$
   - Let $t \in D(v)$ be the value from *v's* domain that satisfies $\{v \to t\} \subseteq l$, set $R^C_l = \{h_{v_t}\}$. Result is the following sets of rules:

$$\begin{array}{l|l|l}
R_c l_1 : h_c l_1 \to h_{C_f} & R_c l_2 : h_c l_2 \to h_{C_t} & R_W l_1 : h_W l_1 \wedge h_{S_f} \wedge h_{R_f} \to h_{W_t} \\
R_R l_1 : h_R l_1 \wedge h_{C_f} \to h_{R_f} & R_S l_1 : h_S l_1 \wedge h_{C_f} \to h_{S_f} & R_W l_2 : h_W l_2 \wedge h_{S_t} \wedge h_{R_f} \to h_{W_t} \\
R_R l_2 : h_R l_2 \wedge h_{C_f} \to h_{R_t} & R_S l_2 : h_S l_2 \wedge h_{C_f} \to h_{S_t} & R_W l_3 : h_W l_3 \wedge h_{S_f} \wedge h_{R_t} \to h_{W_t} \\
R_R l_3 : h_R l_3 \wedge h_{C_t} \to h_{R_f} & R_S l_3 : h_S l_3 \wedge h_{C_t} \to h_{S_f} & R_W l_4 : h_W l_4 \wedge h_{S_t} \wedge h_{R_t} \to h_{W_t} \\
R_R l_4 : h_R l_4 \wedge h_{C_t} \to h_{R_t} & R_S l_4 : h_S l_4 \wedge h_{C_t} \to h_{S_t} &
\end{array}$$

Finally, the goal set $G = \{h_C, h_W, h_R, h_S, h_{W_t}\}$ and $R_G : h_C \wedge h_W \wedge h_R \wedge h_S \wedge h_{W_t} \to G$

As discussed above, finding the LCP for this derived CBA system will be the same as finding MAP for BN in fig.1. In other words, the values assigned to CBA variables to reach the LCP are the same values that achieve MAP for the equivalent BN. Section 3 will illustrate how HORN can be used to find LCP for the CBA which will be the same as finding MAP for BN.

2.1 Discussion

The algorithm did a linear time transformation from BN to CBA. However, we have the following comments:

1. There is no analysis in terms of the size ratio between the BN as an input to the CBA system as an output.
2. While the algorithm generates an equivalent CBA system in terms of solution, it might not be the most optimal system in terms of size. We did an experiment where we shrank the R_G to be $h_{W_t} \to G$ and we obtained the same LCP, but with more computational effort.

3. The algorithm does not remove the redundant CPT entries to save memory space. Also, we do not need to create rules for the entire CPT of the evidence nodes. We only need to create rules for the values observed.
4. It is not clear from the algorithm how we should deal with CPT entries with probability equal to 0. Considering such probabilities will cause all assumables to have costs of ∞, which means we cannot afford explaining our goal.

3 Finding LCP Using HORN

Here, we will summarize how to solve the CBA system using HORN. The reader is directed to [7] [8] for full details. The process goes as follows:

1. Without loss of generality, we start by processing the CBA system such that all consequents are provable. Also, we make sure that every provable appears only once as a consequent in the system.
2. Given the preprocessed CBA, we reverse the implication direction of all rules to avoid the null antecedent proofs.
3. We transform the CBA into PL pairs. PL is an extension of propositional logic; the reader is directed to [10] for a complete review of PL.
4. We generate the equivalent energy function for all PL formulas using the following characteristic function provided by Pinkas [10]:

$$H_\sigma = \begin{cases} x_i & \text{if } \sigma = x_i \text{ is atomic proposition} \\ 1 - H_{\sigma'} & \text{if } \sigma = \neg\sigma' \\ H_{\sigma_1} \times H_{\sigma_2} & \text{if } \sigma = \sigma_1 \wedge \sigma_2 \\ H_{\sigma_1} + H_{\sigma_2} - H_{\sigma_1} \times H_{\sigma_2} & \text{if } \sigma = \sigma_1 \vee \sigma_2 \end{cases} \tag{7}$$

This function maps every propositional sentence σ into a characteristic algebraic term H_σ that has its maximal points at the truth assignments that satisfy the clause. The equivalent energy function for a given proposition sentence σ is the characteristic function of the sentence negation $H_{\neg\sigma}$. The Energy function for PL pairs $\psi = \wedge_i^n \sigma_i$ is defined by (8); this energy function fully specifies the equivalent HORN.

$$E_\Psi = \sum_i^n H_{\neg\sigma} \tag{8}$$

3.1 Deriving Energy Functions for Logical Rules with More Than 3 Variables

In this section we provide derivations of the energy functions for logical rules with more than 3 variables. We start by OR rule; consider $\beta = x_n \rightarrow x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_{n-1}$:

$$\begin{aligned} \neg\beta &= x_n \wedge \neg(x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_{n-1}) \\ &\equiv x_n \wedge \neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \ldots \wedge x_{n-1} \end{aligned}$$

$$\therefore H_{\neg\beta} = x_n[(1-x_1)(1-x_2)(1-x_3)...(1-x_{n-1})]$$

$$\therefore E_\beta = x_n - x_n\left(\sum_{k=1}^{n-1}(-1)^{k+1}\left(\sum_{1\le i_1<....<i_k\le n-1} x_{i_1}x_{i_2}...x_{i_k}\right)\right) \tag{9}$$

For AND rule, consider $\beta = x_n \rightarrow x_1 \wedge x_2 \wedge x_3 \wedge ... \wedge x_{n-1}$

$$\begin{aligned}\neg\beta &\equiv x_n \wedge \neg(x_1 \wedge x_2 \wedge x_3 \wedge ... \wedge x_{n-1}) \\ &\equiv x_n(1 - x_1x_2x_3...x_{n-1})\end{aligned}$$

$$\therefore H_{\neg\beta} = x_n - x_nx_1x_2x_3...x_{n-1}$$

$$\therefore E_\beta = x_n - \prod_{i=1}^{n} x_i \tag{10}$$

4 Solution Quality and Size Complexity

For the example traced in this manuscript, it is clear that the network reached the LCP and assigned values which give the maximum joint probability for the random variables of the BN in Fig.1. In general, we judge if the network reached the global minima by benchmarking the solution against the results obtained by the popular public domain *lp-solve* engine which solves the CBA system after converting it to the equivalent Linear Programming (LP) form.

The example above showed that HORN solved a problem of size 26-hypothesis, 22-rule. However, previous work [7] [8] showed that HORN constantly found feasible solutions for a CBA system with 300-hypothesis, 900-rule with high difficulty. Problem size is not the only factor which determines the CBA instance difficulty and its search space complexity. Other factors, like solution depth, rules length, and ratio between the number of rules to the total number of hypotheses are taken into consideration when considering a CBA instance difficulty level.

5 Results Summary

Using the previously mentioned transformations, we constructed the energy function which represents the CBA system derived in section 2. Then, we used HORN simulator to minimize this energy function. The LCP was found through the following assignments $\{C \rightarrow T, R \rightarrow T, S \rightarrow F, W \rightarrow T\}$. Total cost of 8.6725 by assuming the following hypotheses $h_C l_2$, $h_S l_3$, $h_R l_4$ and $h_W l_3$. This is the same solution we reached using Bayesian inference for BN in fig.1. Table 5 summarizes the results of the HORN which solved this example.

We can also find the LCP by backtracking the rules starting from the goal rule. We only need to calculate towards h_{W_t} because all other hypotheses in the goal rule are provables with assumability cost of ∞. By backtracking rules R_{W_t} 's, it is clear that we cannot use $R_W l_1$ which costs us ∞, because it explains that the grass is wet while there is neither rain nor sprinkler. That leaves us with rules $R_W l_2$, $R_W l_3$ and $R_W l_4$ with costs : 8.9278, 8.6725 and 9.4884, respectively. That means the best explanation for the observation that the grass is wet is $R_W l_3$ which assumes that the sprinkler is off and there is rain. The LCP assignment of the constructed CBA system is the same assignment for the variables in the BN to achieve MAP.

Table 5. Results summary

R_G	network order	network iterations	cost
$h_C \wedge h_W \wedge h_R \wedge h_S \wedge h_{W_t} \rightarrow G$	9	98489	8.6725
$h_{W_t} \rightarrow G$	9	136128	8.6725

6 Concluding Remarks and Future Work

In this paper we showed how to find MAP in BN using HORN through an intermediate representation of CBA. This method creates the full integrated energy function directly without explicitly setting the objective and constraint functions. We traced and analyzed the only algorithm in literature that transforms BN to CBA. Future work would be to invent new algorithms that transform BN to CBA while taking care of the size ratio between both systems. Finding MAPs for BN with continuous probability distribution will be an interesting follow up for this work. Also, we can study which classes of BN can create polynomially solvable CBA systems.

References

1. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 4, pp. 463–502. Edinburgh University Press (1969)
2. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
3. Abdelbar, A.M.: Designing high order recurrent networks for Bayesian belief revision. In: Medsker, L.R., Jain, L.C. (eds.) Recurrent Neural Networks: Design and Applications, pp. 77–98. CRC Press, Boca Raton (1999)
4. Shimony, S.E.: Finding MAPs for belief networks is NP-hard. Artificial Intelligence 68, 399–410 (1994)
5. Abdelbar, A.M., Hedetniemi, S.M.: Approximating MAPs for belief networks is NP-hard and other theorems. Artificial Intelligence 102, 21–38 (1998)

6. Abdelbar, A.M.: An algorithm for finding MAPs for belief networks through cost-based abduction. Artificial Intelligence 104, 331–338 (1998)
7. Abdelbar, A.M., Andrews, E.A.M., Wunsch II, D.C.: Abductive Reasoning with Recurrent Neural Networks. Neural Networks 16(5-6), 665–673 (2003)
8. Abdelbar, A.M., El-Hemely, M.A., Andrews, E.A.M., Wunsch II, D.C.: Recurrent Neural Networks with Backtrack-Points and Negative Reinforcement Applied to Cost-Based Abduction. Neural Networks 18(5-6), 755–764 (2005)
9. Charniak, E., Shimony, S.E.: Probabilistic semantics for cost-based abduction. In: AAAI National Conference on Artificial Intelligence, pp. 106–111. AAAI, Boston (1990)
10. Pinkas, G.: Reasoning, nonmonotonicity and learning in connectionist networks that capture prepositional knowledge. Artificial Intelligence 77, 203–347 (1995)
11. Charniak, E., Shimony, S.E.: Cost-based abduction and MAP explanation. Artificial Intelligence 66, 345–374 (1994)
12. Den, Y.: Generalized Chart Algorithm: An Efficient Procedure for Cost-Based Abduction. In: 32nd annual Meeting of the Association for Computational Linguistics, pp. 218–225. New Mexico Association for Computational Linguistics, Las Cruces (1994)
13. Ishizuka, M., Matsuo, Y.: SL Method for Computing a Nearoptimal Solution Using Linear and Non-Linear Programming in Cost-Based Hypothetical Reasoning. Knowledge-based systems 15, 369–376 (2002)
14. Santos, E.: A Linear constraint satisfaction approach to Cost-Based Abduction. Artificial Intelligence 65(1), 1–27 (1994)
15. Santos, E.J., Santos, E.S.: Polynomial Solvability of Cost-Based Abduction. Artificial Intelligence 86, 157–170 (1996)
16. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. National Academy of Science 79, 2554–2558 (1982)
17. Murphy, K.: A Brief Introduction to Graphical Models and Bayesian Networks, `http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`