Using Weighted MAX-SAT Engines to Solve MPE

James D. Park
Computer Science Department
University of California
Los Angeles, CA 90095
jd@cs.ucla.edu

Abstract

Logical and probabilistic reasoning are closely related. Many examples in each group have natural analogs in the other. One example is the strong relationship between weighted MAX-SAT and MPE. This paper presents a simple reduction of MPE to weighted MAX-SAT. It also investigates approximating MPE by converting it to a weighted MAX-SAT problem, then using the incomplete methods for solving weighted MAX-SAT to generate a solution. We show that converting MPE problems to MAX-SAT problems and using a method designed for MAX-SAT to solve them often produces solutions that are vastly superior to the previous local search methods designed directly for the MPE problem.

Introduction

Probabilistic reasoning has a strong relation to logical reasoning. Many problems from one class have natural analogs in the other. The similarities between the problems from probabilistic reasoning and logical reasoning sometimes allows solution techniques to be transfered from one domain to the other. For example, previous results for the Most Probable Explanation (MPE) (Kask & Dechter 1999) have been obtained by noticing the similarity between MPE and satisfiability, and leveraging incomplete methods designed for satisfiability in order to approximately solve the MPE problem. In the other direction, methods used in probabilistic problems have been applied in order to improve the theoretical analysis of logical problems (Rish & Dechter 2000).

The *Most Probable Explanation* (MPE) is the problem of finding the variable instantiation of a Bayesian network that has the highest probability given some evidence. Essentially it provides the most likely state of the system given what has been observed. MPE has a number of applications including diagnosis and explanation. Unfortunately, MPE is an NP-complete problem (Littman 1999), so approximation techniques are necessary. As mentioned above, local search techniques for MPE have been inspired by the relation between satisfiability and MPE. MPE enjoys even closer ties with weighted maximum satisfiability (weighted MAX-SAT), another problem from logic which shares MPE's optimization characteristics. *Weighted MAX-SAT* is the problem of taking a set of clauses with associated weights, and finding the instantiation that produces the largest sum of the weights of satisfied clauses. Weighted MAX-SAT is used for example to resolve con¤icts in a knowledge base. Finding approximate solutions to weighted MAX-SAT has received significant research attention, and novel algorithms have been developed that have proved to be very successful.

This paper investigates using local search algorithms developed for weighted MAX-SAT and applying them to approximately solve MPE. Local search is a general optimization technique which can be used alone or as a method for improving solutions found by other approximation methods. We compare two successful local search algorithms in the MAX-SAT domain (Discrete Lagrangian Multipliers (Wah & Shang 1997), and Guided Local Search (Mills & Tsang 2000)) to the local search method proposed for MPE (Kask & Dechter 1999). For large problems, the MAX-SAT algorithms proved to be significantly more powerful, typically providing instantiations that are orders of magnitude more probable.

The paper is organized as follows: First, we formally introduce the MPE and MAX-SAT problems. Then we present the reduction of MPE to MAX-SAT. We then introduce the MAX-SAT algorithms that will be evaluated. Finally, we provide experimental results comparing the solution quality of MPE approximations using the MAX-SAT methods to the previously proposed local search method developed for MPE.

MPE and MAX-SAT

The variables that appear in the Bayesian networks we will consider are over a finite domain of possible values. Each variable can take on a single value from a finite set of mutually exclusive possibilities.

Definition 1 *An instantiation of a set of variables is a function that assigns a value to each variable in the set. Two instantiations are compatible if they agree on the assignment of all of the variables that they have in common.*

We denote instantiations by the values of each variable to simplify the notation since the variable it is associated with will be clear from the context. Consider, for example, a variable A that can take on values in $\{a_1, a_2, a_3\}$, and variable

A	$Pr(A)$
a_1	.3
a_2	.5
a_3	.2

A	B	$Pr(B\|A)$
a_1	b_1	.2
a_1	b_2	.8
a_2	b_1	1
a_2	b_2	0
a_3	b_1	.6
a_3	b_2	.4

Figure 1: CPTs for a Bayesian network $A \rightarrow B$.

B that can take on values in $\{b_1, b_2\}$. Then instantiation (a_1) is compatible with instantiations (a_1, b_2) and (b_3), but not with (a_2, b_2). For boolean variables, we denote the values by the lowercase variable name or the negated variable name. For example for boolean variables C and D, the instantiation $(c, \overline{d})$ represents the assignment of C to true and D to false.

Definition 2 *A conditional probability table (CPT) T for a variable V with a set of parent variables $\mathbf{P}$ is a function that maps each instantiation of $V \cup \mathbf{P}$ to a real value in $[0, 1]$ such that for any instantiation $\mathbf{p}$ of $\mathbf{P}$, $\sum_v T(\{v\} \cup \mathbf{p}) = 1$ where v ranges over the values of V.*

A CPT provides the probability for each possible value of V given a particular instantiation of the parents. It is called a table since it is often represented in tabular form, enumerating the conditional probability associated with each instantiation. Figure 1 contains example CPTs corresponding to $Pr(A)$ and $Pr(B|A)$

Bayesian networks represent a probability distribution over a set of variables, factored into a specific form. It consists of a directed graph which specifies specific independence relationships between the variables, together with a conditional probability table for each variable. Formally,

Definition 3 *A Bayesian network is a pair $(\mathcal{G}, \mathcal{P})$ where $\mathcal{G}$ is a directed acyclic graph whose nodes are variables, and $\mathcal{P}$ is a set which consists of the CPT of each variable in $\mathcal{G}$, where the parents of each CPT correspond to the parents of the corresponding variable in the graph.*

Because of the way Bayesian networks are factored, computing the probability of a complete instantiation of its variables is very simple. The probability of a complete instantiation is the product of the entry of each CPT that is compatible with the instantiation. For example, in the network in Figure 1 the instantiation (a_3, b_1) has probability $.2 * .6 = .12$

The *MPE* problem is defined as follows: Given a Bayesian Network and an instantiation of a subset of the variables (the *evidence*), find the (not necessarily unique) complete instantiation with the highest probability that is compatible with the evidence.

Now we will consider some related concepts in logical reasoning. Unlike in probabilistic reasoning, logical reasoning deals with propositional variables. We will use the following standard terminology:

Definition 4 *A literal is a variable or its negation. A clause is a disjunction of literals and a weighted clause is a clause, together with a non-negative weight. A weighted CNF formula is a set of weighted clauses.*

We denote clauses using standard CNF formula conventions. Weights are denoted as superscripts. We denote a weighted CNF formula by the weighted clauses conjoined together. For example, the following formula consists of three weighted clauses : $(x \vee \overline{y} \vee \overline{z})^3 \wedge (\overline{x})^{10.1} \wedge (y)^{.5}$. The weight of a complete instantiation of a weighted CNF formula is the sum of the weight of the satisfied clauses. So, for the previous example, the instantiation $(x, y, \overline{z})$ has weight $3 + .5 = 3.5$ since it satisfies the first and third clauses, but not the second.

The *MAX-SAT* problem is defined as follows: Given a weighted CNF formula, find the (not necessarily unique) instantiation which has the highest weight.

Reducing MPE to MAX-SAT

An MPE problem can be converted into a weighted CNF expression whose MAX-SAT solution immediately produces the solution to the corresponding MPE problem. We begin by showing how to reduce a Bayesian network with only binary variables and positive CPT entries, and later show how to extend it to handle zeros, non-binary variables, and evidence.

The basic idea is that each CPT entry induces a weighted clause in the induced MAX-SAT problem. The only instantiations that do not satisfy the clause $l_1 \vee l_2 \vee ... \vee l_n$ are the instantiations in which each literal in the clause evaluates to false. We use this fact as the basis of the conversion. Each row in the CPT generates a weighted clause which contains the negation of each of the variables in the row and is weighted with the negative log of the conditional probability. For example, the network $C \rightarrow D$ with CPTs :

C	$Pr(C)$
c	.3
$\overline{c}$	.7

C	D	$Pr(D\|C)$
c	d	.2
c	$\overline{d}$	.8
$\overline{c}$	d	.1
$\overline{c}$	$\overline{d}$	.9

induces the weighted CNF expression

$$(\overline{c})^{-\log .3} \wedge (c)^{-\log .7} \wedge (\overline{c} \vee \overline{d})^{-\log .2}$$

$$\wedge (\overline{c} \vee d)^{-\log .8} \wedge (c \vee \overline{d})^{-\log .1} \wedge (c \vee d)^{-\log .9}$$

.

Consider, for example, the instantiation $c, \overline{d}$. It satisfies all of the clauses except $(\overline{c})^{-\log .3}$ and $(\overline{c} \vee d)^{-\log .8}$. So the sum of weights of the unsatisfied clauses is $-\log .24$ which is the negative log of the probability of that instantiation. Notice that for any instantiation of the variables, the sum of the weights of the unsatisfied clauses equals the negative