The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Solving Weighted Abduction via Max-SAT Solvers

Yoichi Sasaki,[1 2] **Takanori Maehara,**[2] **Takumi Akazaki,**[2]
Kazeto Yamamoto,[1 2] **Kunihiko Sadamasa**[1 2]
[1]NEC Corporation, [2]RIKEN AIP
yoichisasaki@nec.com, takanori.maehara@riken.jp, takumi.akazaki@riken.jp,
kazeto@nec.com, sadamasa@nec.com

Abstract

Abduction is a form of inference that seeks the best explanation for the given observation. Because it provides a reasoning process based on background knowledge, it is used in applications that need convincing explanations. In this study, we consider weighted abduction, which is one of the commonly used mathematical models for abduction. The main difficulty associated with applying weighted abduction to real problems is its computational complexity. A state-of-the-art method formulates weighted abduction as an integer linear programming (ILP) problem and solves it using efficient ILP solvers; however, it is still limited to solving problems that include at most 100 rules of background knowledge and observations. In this study, we first formulate the weighted abduction problem as a Max-SAT problem whose hard clauses are mostly Horn clauses. Then, we propose to solve the problem using modern Max-SAT solvers. In our experiments, the proposed method solved the problems much faster than the state-of-the-art ILP-based weighted abduction.

1 Introduction

Background *Abduction* is a type of inference that seeks the best explanation for a given observation (Peirce 1883). Abduction provides a reasoning process based on prior knowledge; hence, it is used in tasks that need convincing explanations such as law applications (Anderson, Schum, and Twining 2005), plan recognition (Hobbs et al. 1993), and discourse analysis (Ng and Mooney 1992). The abduction is a very classical problem and has been extensively studied in the early age of artificial intelligence (Josephson and Josephson 1996). Recently, the abduction has attracted attention owing to the progress of knowledge acquisition and improvements in the performance of inference algorithms and computers (Gordon 2016).

The abduction problem is formulated as follows. We are given background knowledge B and an observation O, each of which is a set of first-order formulas. The task is to find a hypothesis H that reasonably explains the observation with the background knowledge, i.e., H is a set of first-order formulas satisfying $H \cup B \models O$ and $H \cup B \not\models \bot$.

There are several models for selecting the best hypothesis, such as probabilistic Horn abduction (Poole 1993), weighted abduction (Hobbs et al. 1993), etcetera abduction (Inoue and Gordon 2017), Markov logic networks based abduction (Kate and Mooney 2009), and Bayesian abductive logic programs (Singla and Mooney 2011). Among them, we consider *weighted abduction*, which is a cost-based abduction method that selects H having the lowest cost induced by backchaining and unification (see Sec. 2). The weighted abduction has been widely used in several tasks (Appelt and Pollack 1992; Blythe et al. 2011; Ovchinnikova et al. 2014) since it is a basic model of abduction and easy to incorporate the uncertainty of the background knowledge and observation (Hobbs 2004). In particular, (Ovchinnikova et al. 2014) observed that the weighted abduction has an advantage over alternative approaches when they are applied to the discourse interpretation tasks.

The main difficulty of the weighted abduction (and other models) is its computational cost. In general, the abduction problem is NP-hard (Bylander et al. 1991); hence, it will be impractical if we have to handle a large amount of background knowledge and observations. To overcome this difficulty, (Inoue and Inui 2011) introduced the *depth-limited weighted abduction problem* (see Sec. 3) and formulated the weighted abduction as an ILP problem. To accelerate the performance of the ILP approach to the weighted abduction, (Inoue and Inui 2012) observed that the most constraints represent the transitivity among equalities of the variables, and proposed a method called cutting-plane inference to improve the performance. (Yamamoto et al. 2015) proposed a method to prune the redundant candidate to improve the performance. (Schüller 2016) proposed an *answer set programming (ASP)*-based approach for the weighted abduction. Similar to the ILP-based approach of (Inoue and Inui 2012), their approach also specifies a parameter that limits the depth of the backchaining, which are incomparable to the depth parameter of (Inoue and Inui 2012).

However, even using these methods, the weighted abduction still cannot be applied to large-scale instances, which typically contain more than hundreds of items in background knowledge and observations. Our purpose is to establish a more efficient algorithm for the weighted abduction.

Contributions We propose an efficient method for the weighted abduction. We make the following contributions.

- We formulate the depth-limited weighted abduction problem as a *maximum satisfiability (Max-SAT) problem* whose majority of the hard clauses are Horn clauses, and the number of soft clauses is small.
- We propose to solve the problem using Max-SAT solvers. Our experiments showed that this approach is 100 times faster than the existing method that uses ILP solver for large-scale real-world instances.

As discussed above, most of the existing methods formulated the problem as an ILP problem to use efficient ILP solvers. However, since abduction is a logic problem, we believe that it is more natural to formulate it as a SAT problem.

Our first contribution shows that our problem is "nearly" Max Horn SAT problem with a small number of soft clauses. For example, in our largest instance, the total number of hard clauses is about 2,000,000, but only 6,000 are non-Horn clauses. Also, the number of soft clauses is 6,000. Since the complexity of the Max Horn SAT is exponential in the number of soft clauses, i.e., not the number of hard clauses, it is beneficial to use our Max Horn-like SAT formulation.

Our second contribution verifies the above analytic contribution via experiments. The Max-SAT and ILP formulation had almost the same number of variables and constraints; however, the former solved the problem much faster than the latter. This verifies that our Max-SAT formulation is much better than the ILP formulation for the weighted abduction problem. Here, we have to thank the recent progress of Max-SAT solvers (Balyo, Heule, and Järvisalo 2017). The performance of SAT solvers improves every year, and many high-performance implementations (e.g., OpenWBO (Martins, Manquinho, and Lynce 2014) and MaxHS (Davies and Bacchus 2011)) are freely available. This might change the situation around solving abduction problems.

2 Preliminaries

We follow the standard notation of first-order logic. We denote by $\vec{x} = (x_1, \ldots, x_k)$ a set of variables with a suitable length, and $p(\vec{x})$ for an atomic formula with variable $\vec{x}$.

2.1 Weighted Abduction

Weighted abduction (Hobbs et al. 1993) is a method of abduction based on the costs. In weighted abduction, background knowledge B is a set of first-order Horn clauses, where all of the variables on the left-hand side are universally quantified with the widest possible scope, and the variables only on the right-hand side are existentially quantified. Each Horn clause in the background knowledge is referred to as a *rule*. Each atomic formula $p(\vec{x})$ on the left-hand side of a rule has a positive *weight* w; this is denoted by $p(\vec{x})^w$. Thus, in general, a rule is represented as

$$\forall \vec{x} \exists \vec{y} (p_1(\vec{x}_1)^{w_1} \wedge \cdots \wedge p_k(\vec{x}_k)^{w_k} \Rightarrow q(\vec{x}_0, \vec{y})), \quad (2.1)$$

where $\vec{x}$ and $\vec{y}$ denote (possibly empty) sets of variables and $\vec{x}_0, \ldots, \vec{x}_k$ are subsets of $\vec{x}$. Observation O is a conjunction of existentially quantified atomic formulas $p(\vec{x})$, each of which has a positive cost c; this is denoted by $p(\vec{x})^{\$c}$. Thus, in general, an observation is represented as

$$\exists \vec{x} (p_1(\vec{x}_1)^{\$c_1} \wedge \cdots \wedge p_k(\vec{x}_k)^{\$c_k}). \quad (2.2)$$

A candidate hypothesis H is also in the same format as the observation. The cost of H is the sum of all costs of atomic formulas in the hypothesis, i.e.,

$$c(H) = \sum_{p(\vec{x})^w \in H} w, \quad (2.3)$$

where $p(\vec{x})^w \in H$ means that $p(\vec{x})$ appears in H.

A candidate hypothesis H is generated as follows. Initially, we set $H := O$. Then, we iteratively apply *backchaining* and *unification* in arbitrary order.

Backchaining We select an atomic formula $q(\vec{x}_0, \vec{y})^{\$c} \in H$ and a rule $\forall \vec{x}' \exists \vec{y}' (p_1(\vec{x}'_1)^{w_1} \wedge \cdots \wedge p_k(\vec{x}'_k)^{w_k} \Rightarrow q(\vec{x}'_0, \vec{y}')) \in B$. Then, we replace H by removing $q(\vec{x}_0, \vec{y})^{\$c}$ and adding $p(\vec{x}_1)^{w_1 \$c} \wedge \cdots \wedge p(\vec{x}_1)^{w_k \$c}$. For example, if $H = \exists x_1, x_2 (p(x_1, x_2)^{\$2} \wedge q(x_2)^{\$1})$ and $\forall x'_1, x'_2, \exists y (r(x'_1)^{\$3} \wedge s(x'_2)^{\$4} \Rightarrow p(x'_1, y)) \in B$ then we obtain $H' = \exists x_1, x_2, y (r(x_1)^{\$6} \wedge s(x_2)^{\$8} \wedge q(y)^{\$1})$.

Unification We select sets $\vec{x}, \vec{y}$ of variables of the same size in H. Then, we replace H by letting $\vec{x} = \vec{y}$. This is allowed only if all of the atomic formula containing $\vec{x}_j$ and $\vec{y}_j$ are the same up to the variables. The cost of the resulting atomic formulas are the smallest costs of the original atomic formulas. For example, if $H = \exists x, y (p(x)^{\$2} \wedge p(y)^{\$3} \wedge q(x)^{\$5} \wedge q(y)^{\$4})$, we have $H' = \exists x (p(x)^{\$2} \wedge q(x)^{\$4})$.

A different inference process gives a different hypothesis. The goal of weighted abduction is to find a hypothesis that has the lowest cost.

2.2 Satisfiability Problems

Given a set of clauses, *satisfiability (SAT) problem* asks to find a truth assignment of the variables such that all clauses are satisfied. If all clauses are Horn clauses, the problem is referred to as the *Horn SAT* problem. The SAT problem is NP-hard (Bylander et al. 1991), but the Horn SAT problem is solved in linear time (Dowling and Gallier 1984).

The *Max-SAT problem* is an optimization variant of the SAT problem. A clause having the infinite cost is called a hard clause, which must be satisfied; otherwise, it is called a soft clause. We are given a set of clauses, each of which is associated with a (possibly infinite) weight. The objective value of an assignment is the total weight of the satisfied clauses. The Max-SAT problem seeks the assignment that maximizes the objective value, which equivalently minimizes the total weight of the unsatisfied clauses.

The *Max Horn SAT problem* is a Max-SAT problem whose clauses are Horn clauses. The Max Horn SAT problem is NP-hard (Jaumard and Simeone 1987) and has no constant-factor approximation algorithm unless P = NP (Lee 2017). Therefore, it seems as difficult as the general Max-SAT problem. However, in reality, we can obtain large benefits. In practice, Max Horn SAT problems can be solved

much faster than general Max-SAT problems using modern Max-SAT solvers (Marques-Silva, Ignatiev, and Morgado 2017; Ignatiev, Morgado, and Marques-Silva 2017). Modern Max-SAT solvers iteratively solve the decision version of the problem, find a core, which is unsatisfied clauses, and extract information from the core. Here, in the Max Horn-SAT case, the first two steps are performed in polynomial time; therefore, we expect that the solvers run efficiently compared with the general Max-SAT problem.

3 Weighted Abduction via Max-SAT

Here, we formulate the problem as a Max-SAT problem whose majority of hard clauses are Horn clauses.

A naive approach is searching hypothesis using backchaining and unification from the observation. However, this approach is impractical because of the combinatorial explosion. To overcome this difficulty, (Inoue and Inui 2011) introduced the *depth-limited weighted abduction problem*, which restricts the depth of backchaining at most d, where d is a positive integer specified as a parameter. This formulation has advantages in both application and computation. In a practical application, we usually hope explanations with shallow backchaining because an explanation with deep backchaining may look like the "butterfly effect." Therefore, it is reasonable to restrict the depth of backchaining. In computation, this removes the combinatorial explosion due to backchaining.

3.1 Backchaining Graph

A *backchaining graph of depth* d is a node-weighted directed hypergraph[1] $G = (P, R) \equiv (P^{(i)}, R^{(i)})_{i=0,\dots,d}$ such that each $\alpha \in P$ corresponds to an atomic formula $p(\vec{x})$ (with different variable names), and each $e \in R$ is associated with a rule in the background knowledge. It is constructed as follows. First, we define $P^{(0)} = \{\alpha_1, \dots, \alpha_k\}$ for observation $O = \exists \vec{x}\, (p_1(\vec{x}_1)^{\$c_1} \wedge \dots \wedge p_k(\vec{x}_k)^{\$c_k})$, where $\alpha_1, \dots, \alpha_k$ are associated with $p_1(\vec{x}_1), \dots, p_k(\vec{x}_k)$, respectively, and the weights are defined by $w(\alpha_1) = c_1$, …, $w(\alpha_k) = c_k$. We define $R^{(0)} = \emptyset$. Then, we perform the following procedure for $i = 1, \dots, d$. We initialize $P^{(i)} = P^{(i-1)}$ and $R^{(i)} = R^{(i-1)}$. For each rule $\forall \vec{x}' \exists \vec{y}'\, (p_1(\vec{x}'_1)^{w_1} \wedge \dots \wedge p_k(\vec{x}'_k)^{w_k} \Rightarrow q(\vec{x}'_0, \vec{y}')) \in B$ and node $\beta \in P^{(i-1)}$ that is associated with $q(\vec{x}_0, \vec{y})$ (i.e., having the same predicate name), we create new nodes $\beta_1, \dots, \beta_k$, which are associated with $\bar{p}_1(\vec{x}'_1), \dots, \bar{p}_k(\vec{x}'_k)$, respectively, and the weights are defined by $w(\beta_1) = w_1 w(\beta)$, …, $w(\beta_k) = w_k w(\beta)$. Here, we introduce new variables $\vec{x}'$ to distinguish them from existing variables. We add these new nodes to $P^{(i)}$ and add new hyperedge $(\{\beta_1, \dots, \beta_k\}, \beta)$ to $R^{(i)}$. We refer to the nodes in P as *latent atomic formula*. The backchaining graph is a hyperforest since each application of backchaining creates new nodes.

The above construction generates all atomic formulas using the applications of backchaining with depth at most d. (Yamamoto et al. 2015) proposed an A^*-search-based heuristics to construct a smaller graph by pruning atomic formulas that do not constitute the optimal hypothesis.

3.2 Max-SAT Formulation

After constructing the backchaining graph, our remaining tasks are to determine (1) which atomic formulas are included in the best hypothesis, and (2) which variables are unified. We solve them by reducing to a Max-SAT problem.

Let V be the set of variables in the atomic formulas corresponding to P, and C be the set of constants in them. Our encoding to Max-SAT is based on (Inoue and Inui 2011). We introduce four types of Boolean variables: h_α, r_α, $u_{\alpha,\beta}$ for all $\alpha \in P$, and $\beta \in P$ associated with the same atomic formula (i.e., with a different variable name) with that of α, and $s_{x,y}$ for all variables or constants $x, y \in V \cup C$. In the following, we design formulas to enforce these variables representing the following events.

- $h_\alpha = \texttt{True}$ if $p(\vec{x})$ corresponding to α can be deduced from the hypothesis (i.e., it is explained by others or included in hypothesis).
- $r_\alpha = \texttt{True}$ if $p(\vec{x})$ corresponding to α does not have to pay the cost even if $p(\vec{x})$ is explained (i.e., it is explained or unified with another atomic formula).
- $u_{\alpha,\beta} = \texttt{True}$ if $p(\vec{x})$ corresponding to α and $p(\vec{y})$ corresponding to β are unified.
- $s_{x,y} = \texttt{True}$ if the variables or constants x and y are equal. We define $s_{x,y} = \texttt{False}$ for all distinct constants.

Then, we can see that $h_\alpha \wedge \neg r_\alpha$ implies that α is included in the solution hypothesis. Note that we use this representation because most of the constraints use h_α instead of $h_\alpha \wedge \neg r_\alpha$; hence, it reduces the lengths of the formulas.

Objective Function (Soft Clauses) The objective of weighted abduction is to find a set of latent atomic formulas such that the total cost is the smallest. By the encoding, α pays a cost if $h_\alpha \wedge \neg r_\alpha$ holds. Thus, we set the negations of these formulas to the soft clauses as

$$h_\alpha \Rightarrow r_\alpha; \;\; \text{weight} = w(\alpha), \quad \alpha \in P. \tag{3.1}$$

By the construction, if an assignment falsifies this clause, the assignment contains α in the solution hypothesis; therefore, it pays the cost $w(\alpha)$. These are Horn clauses.

Constraint 1: Observation must be true We require that the latent observation must be true. Thus, we add

$$h_\alpha, \quad \alpha \in \bar{O} \tag{3.2}$$

to the formula. These are Horn clauses.

Constraint 2: Unification is only performed for explained atomic formulas Two latent atomic formulas can be unified only if both are explained or included in hypothesized. So, we add the following constraint, which are Horn clauses.

$$u_{\alpha,\beta} \Rightarrow h_\alpha, \quad u_{\alpha,\beta} \Rightarrow h_\beta, \quad \alpha, \beta \in P \tag{3.3}$$

[1] A *directed hypergraph* (P, R) is a pair of finite sets P and $R \subseteq 2^P \times P$. Each hyperedge $e \in R$ is denoted by $e = (S, t)$, where $S \subseteq V$ and $t \in V$.

Constraint 3: Unified atomic formulas do not have to pay the costs A latent atomic formula α does not have to pay the cost only if it is explained or unified. To represent this constraint compactly, we first add the constraint:

$$(\bigwedge_{\beta \in S} h_\beta) \vee (\bigwedge_{\beta \in S} \neg h_\beta), \quad (S,t) \in R \text{ for } t \in P. \tag{3.4}$$

This means that all the atomic formulas that are children of $\beta \in P$ have the same value. Since the generated hypergraph is a hyperforest, we can safely add this constraint. (Note that two latent atomic formulas associated with the same atomic formula can have different values). Since these are not Horn clauses, we convert them using the Tseitin transformation (Tseitin 1968). For each S, we introduce a new variable X_S and add the clause $\left(\bigwedge_{\beta \in S} h_\beta \Rightarrow X_S\right) \wedge \left(X_S \Rightarrow \bigwedge_{\beta \in S} h_\beta\right)$ to the formula. Then, $X_S \equiv \bigwedge_{\beta \in S} h_\beta \Rightarrow X_S$. The first conjunction is a Horn clause, and the second conjunction is a conjunction of a Horn clause, i.e., $X_S \Rightarrow \bigwedge_{\beta \in S} h_\beta$ is equivalent to $X_S \Rightarrow h_\beta$ for all $\beta \in S$. For the above event, the original clause is equivalent to $\bigwedge_{\beta \in S} (h_\beta \Rightarrow X_S)$. Therefore, (3.4) is represented by Horn clauses.

Under this constraint, the cost condition is represented by

$$\neg r_\alpha \vee \left(\bigvee_{\beta \in I(\alpha)} h_\beta\right) \vee \left(\bigvee_{\beta \in \text{small}(\alpha)} u_{\alpha,\beta}\right) \tag{3.5}$$

where $I(\alpha) = \{\beta \in P : (S,\alpha) \in R, \beta \in S\}$ is the set of the in-neighborhood of α, which are the nodes that explain α, and $\text{small}(\alpha) = \{\beta \in P : c(\beta) \leq c(\alpha)\}$ is the nodes having a smaller cost than β. Here, we introduce arbitrary tie-breaking to make all of the costs have different values. This cannot be converted into a Horn clause if we use any transformation. So, (3.5) is not represented by Horn clauses.

Constraint 4: Variable transitivity If $x = y$ and $y = z$ then $x = z$. This is represented as follows:

$$\begin{aligned} &s_{x,y} \wedge s_{y,z} \Rightarrow s_{x,z}, \quad s_{x,z} \wedge s_{y,z} \Rightarrow s_{x,y}, \\ &s_{x,y} \wedge s_{x,z} \Rightarrow s_{y,z}, \quad x,y,z \in V \cup C. \end{aligned} \tag{3.6}$$

These are Horn clauses.

Constraint 5: Valid substitution If we unify two latent atomic formulas p and q, all corresponding variables must take the same value. This condition is represented by

$$u_{p,q} \Rightarrow s_{x,y}, \quad p,q \in P, (x,y) \in \text{usub}(p,q), \tag{3.7}$$

where $\text{usub}(p,q)$ is the matching pairs of the variables of p and q that can be unified. For example, given atomic formula $p'(x_1,x_2,x_3)$ and $q'(y_1,y_2,y_3)$ corresponding nodes $p \in P$ and $q \in P$, respectively, we have $\text{usub}(p,q) = \{(x_1,y_1),(x_2,y_2),(x_3,y_3)\}$. These are Horn clauses.

Now all the constraints of the depth-limited weighted abduction problem are encoded as an instance of the Max-SAT problem. We analyze the instance. First, we see that most of the hard clauses come from Constraint 4 (variable transitivity), which are Horn clauses. The non-Horn clauses only come from Constraint 3, whose number is at most the number of the nodes in the backchaining graph, which are much smaller than the total number of the hard clauses. Next, we see that the number of soft clauses is equal to the number of vertices of the backchaining graph. Thanks to the A^*-search technique mentioned in the last of Section 3.1, the size of the backchaining graph is not too large. By summarizing the above discussion, we obtain the following.

Claim 3.1. A depth-limited weighted abduction problem is reduced to a Max-SAT problem with a small number of soft clauses and a small number of non-Horn clauses.

As mentioned in Section 2, modern Max-SAT solvers efficiently solve Max Horn-SAT problems. We expect these also solves our problem efficiently.

4 Experiments

We conducted the following experiments to evaluate our proposed method. We refer to our method as *SAT-WA*.

All experiments were conducted on a computer (Intel Xeon E3-1270 V2 (4C/3.50GHz/8M) with 32GB of memory). The code of our method was implemented in C++ (g++ [version4.8.5] with the -O2 option). We implemented our algorithm (*SAT-WA*) using OpenWBO[2] and MaxHS[3], which are particularly efficient Max-SAT solvers. For comparison, we employed David[4], which is a successor to Phillip (Yamamoto et al. 2015), with the Gurobi optimizer[5] and ASP-based method (ASP-WA)[6] (Schüller 2016) using Gringo[7] for grounding and Wasp[8] for solving. Note that ASP-WA can be used for datasets that do not contain cycles because the method does not support depth-limitation. As a result, we compared ASP-WA with other method only for the ACCEL dataset without the depth limitation (see below).

4.1 Datasets

Fraud We created a new dataset for a real-world event detection problem. The task is to seek the possibility of "fraud" from emails[9]. The background knowledge consists of rules such as *competitor*$(x,\ y) \wedge$ *provide*(*PRICE_INFO*, $x,\ y) \Rightarrow$ *cartel*$(x,\ y)$, which means that "*if x and y are competitors and x provides y the price information, then x and y form a cartel.*" We created 215 rules by hand. The observation is a set of events that are extracted from emails. We selected 10 observations that are extracted using a lexical analysis. Each observation has 331.9 atomic formulas on average. By performing weighted abduction for this instance, we may obtain a possibility of fraud (e.g., cartel) that can explain the

[2] http://sat.inesc-id.pt/open-wbo/

[3] http://www.maxhs.org/docs/papers.html

[4] https://github.com/aurtg/open-david

[5] http://www.gurobi.com/

[6] https://bitbucket.org/knowlp/asp-fo-abduction

[7] http://potassco.sourceforge.net/

[8] https://github.com/alviano/wasp/

[9] The emails are manually and artificially created based on the actual incidents. Thus, there is no issue in law, privacy, and ethics.

d		Graph	Convert	Solve	Total (ratio of solved)
1	David_gurobi	0.067	0.002	0.008	0.077 (10/10)
	SAT-WA_openwbo	0.065	0.002	0.006	0.074 (10/10)
	SAT-WA_maxhs	0.064	0.002	0.006	0.073 (10/10)
3	David_gurobi	5.13	0.053	362.75	367.93 (7/10)
	SAT-WA_openwbo	4.71	0.051	0.176	4.94 (10/10)
	SAT-WA_maxhs	4.51	0.051	77.02	81.59 (7/10)
5	David_gurobi	8.72	0.110	468.06	476.89 (7/10)
	SAT-WA_openwbo	8.25	0.107	0.305	8.66 (10/10)
	SAT-WA_maxhs	8.12	0.107	77.81	86.04 (10/10)

(a) Running time [sec] varying depth value $d = 1, 3, 5$.

	$d = 1$	$d = 3$	$d = 5$
#nodes of graph	564.9	2983.3	4385.5
#valuables	797.9	15372.5	33167.5
#constraints	994.7	26500.2	63285.6

(b) The size of ILP and Max SAT problem

Table 1: Comparison of David and SAT-WA for Fraud. All values are the averages over the instances.

d		Graph	Convert	Solve	Total (ratio of solved)
1	David_gurobi	0.006	0.01	6.167	6.185 (50/50)
	SAT-WA_openwbo	0.01	0.01	0.031	0.056 (50/50)
	SAT-WA_maxhs	0.008	0.01	0.115	0.137 (50/50)
3	David_gurobi	0.48	0.16	159	159.6 (27/50)
	SAT-WA_openwbo	2.78	1.81	6.303	10.89(50/50)
	SAT-WA_maxhs	2.68	1.81	53.60	58.10(50/50)
5	David_gurobi	0.74	0.23	262.8	263.8 (26/50)
	SAT-WA_openwbo	4.24	2.41	9.05	15.70(50/50)
	SAT-WA_maxhs	3.92	2.43	78.19	84.54(50/50)
∞	David_gurobi	0.68	0.23	263.9	264.9 (26/50)
	SAT-WA_openwbo	4.39	2.46	9.22	16.08(50/50)
	SAT-WA_maxhs	4.08	2.46	79.65	86.20(50/50)
	ASP-WA	—	—	—	14.71(50/50)

(a) Running time [sec] varying depth value $d = 1, 3, 5, \infty$.

	$d = 1$	$d = 3$	$d = 5$	$d = \infty$
#nodes of graph	141.38	4853.42	5917.84	5917.84
#valuables	3575.66	540853	723639	723639
#constraints	7830.48	1248961	1672480	1672480

(b) The size of ILP and Max SAT problem

Table 2: Comparison of David and SAT-WA for ACCEL. All values are the averages over the instances.

observation. We assign the uniform weights to the rules so that the total weight equals 1.2. We assign a cost of 10 for each observation. This dataset contains cycles; thus, ASP-WA method cannot be applied for this dataset.

ACCEL The ACCEL dataset is the dataset used in (Schüller 2016). It is extracted from the dataset originally developed for the abductive plan recognition system ACCEL (Ng and Mooney 1992) and one rule that makes the background knowledge cycle is removed from it.

The background knowledge and observations of this dataset were obtained by extracting 189 background axioms and 50 plan recognition problems, respectively. The plan recognition problems provide agents' partial actions as a conjunction of atomic formulas. The number of rules in the background knowledge is 189 and the average number of atomic formulas for 50 observations is 12.58. This contains no cycles. Thus, we can apply ASP-WA for this dataset.

4.2 Evaluation

We measured the number of solved problems in 1000 seconds and the average running times of David and SAT-WA on the Fraud and ACCEL datasets with the depth of 1, 3, and 5. We also measured the average running time of ASP-WA and the above methods on ACCEL with the depth of ∞. The average is taken over the instances in the datasets (i.e., 50 for ACCEL and 10 for Fraud). The running time is separated as the following three parts: i) Graph: the running time for generating the backchaining graph, ii) Convert: the running time for converting to the ILP or Max Horn-SAT problem and iii) Solve: the running time for finding an optimal hypothesis by the solvers. SAT-WA and David performed the same procedure for Graph part and almost the same for Convert part. The difference will appear in Solve part.

4.3 Results

Tables 1a and 2a present the running times for the Fraud and ACCEL datasets that could be solved in 1000 seconds, respectively. For all cases, SAT-WA outperformed David significantly. For the Fraud dataset with $d = 5$, the all of the problems were solved by SAT-WAs with all SAT solvers, while the 30% (3/10) of the problems were not solved by David with ILP solver. For the ACCEL dataset with $d = 5$, all problems were solved by SAT-WAs with all SAT solvers, while the 48% (24/50) of the problem could not be solved by David with ILP solver. This implies that SAT-WAs were much efficient than David, and they efficiently worked for reasonable depths. Besides, for depth $= \infty$ SAT-WA with OpenWBO efficiently ran as well as ASP-WA.

To understand the reason why SAT-WA was so efficient, we looked into the details of the ILP and Max-SAT problems. Tables 1b and 2b show the size of these problems converted from the weighted abduction problem. As expected by Claim 3.1, the number of soft clauses, which equals to the number of nodes of the backchaining graph, is small compared with the other parameters. Hence, we can expect that the SAT solver can solve the problem efficiently.

5 Conclusion

We proposed a Max-SAT formulation of the weighted abduction problem and proposed to solve the problem using Max-SAT solvers. The Max-SAT problems obtained by the formulation have a small number of soft clauses compared with the hard clauses, and the hard clauses are mostly Horn clauses. This indicates that the problems can be solved efficiently using modern SAT solvers. Experimental results supports this observation. Our method outperformed the existing methods by a factor of 100 for real-world instances.

There are several possible directions. One promising direction is to develop a tailored solver, instead of general Max-SAT solvers. Our Max-SAT problems have many Horn clauses; thus, a progress on the Max Horn-SAT solvers can be incorporated. Another promising direction is to learn the

parameters (i.e., the cost and weight) for the weighted abduction problem from dataset. A learning process may require to solve the weighted abduction problem several times; hence, our efficient method will help such the process.

Acknowledgements The authors are grateful to Yuzuru Okajima for his comments and discussion.

References

Anderson, T.; Schum, D.; and Twining, W. 2005. *Analysis of evidence*. Cambridge University Press.

Appelt, D. E., and Pollack, M. E. 1992. Weighted abduction for plan ascription. *User modeling and user-adapted interaction* 2(1-2):1–25.

Balyo, T.; Heule, M.; and Järvisalo, M. 2017. *Proceedings of SAT Competition 2017: Solver and Benchmark Descriptions*. University of Helsinki, Department of Computer Science.

Blythe, J.; Hobbs, J. R.; Domingos, P.; Kate, R. J.; and Mooney, R. J. 2011. Implementing weighted abduction in markov logic. In *Proceedings of the 9th International Conference on Computational Semantics*, 55–64.

Bylander, T.; Allemang, D.; Tanner, M. C.; and Josephson, J. R. 1991. The computational complexity of abduction. *Artificial intelligence* 49(1-3):25–60.

Davies, J., and Bacchus, F. 2011. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Principles and Practice of Constraint Programming - CP 2011*, 225–239.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming* 1(3):267–284.

Gordon, A. S. 2016. Commonsense interpretation of triangle behavior. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3719–3725.

Hobbs, J. R.; Stickel, M. E.; Appelt, D. E.; and Martin, P. 1993. Interpretation as abduction. *Artificial intelligence* 63(1-2):69–142.

Hobbs, J. R. 2004. Abduction in natural language understanding. *Handbook of pragmatics* 724–741.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2017. On tackling the limits of resolution in sat solving. In *International Conference on Theory and Applications of Satisfiability Testing*, 164–183.

Inoue, N., and Gordon, A. S. 2017. A scalable weighted Max-SAT implementation of propositional Etcetera Abduction. In *Proceedings of the International Conference of the Florida AI Society*, 62–67.

Inoue, N., and Inui, K. 2011. Ilp-based reasoning for weighted abduction. In *Plan, Activity, and Intent Recognition, Papers from the 2011 AAAI Workshop*.

Inoue, N., and Inui, K. 2012. Large-scale cost-based abduction in full-fledged first-order predicate logic with cutting plane inference. In *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Proceedings*, 281–293.

Jaumard, B., and Simeone, B. 1987. On the complexity of the maximum satisfiability problem for horn formulas. *Information Processing Letters* 26(1):1–4.

Josephson, J. R., and Josephson, S. G. 1996. *Abductive inference: Computation, philosophy, technology*. Cambridge University Press.

Kate, R. J., and Mooney, R. J. 2009. Probabilistic abduction using markov logic networks. In *Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition*.

Lee, E. 2017. *Optimal Approximabilities beyond CSPs*. Ph.D. Dissertation, Carnegie Mellon University Pittsburgh.

Marques-Silva, J.; Ignatiev, A.; and Morgado, A. 2017. Horn maximum satisfiability: Reductions, algorithms and applications. In *Portuguese Conference on Artificial Intelligence*, 681–694. Springer.

Martins, R.; Manquinho, V. M.; and Lynce, I. 2014. Openwbo: A modular maxsat solver,. In *Theory and Applications of Satisfiability Testing - SAT 2014*, 438–445.

Ng, H. T., and Mooney, R. J. 1992. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, 499–508.

Ovchinnikova, E.; Montazeri, N.; Alexandrov, T.; Hobbs, J. R.; McCord, M. C.; and Mulkar-Mehta, R. 2014. Abductive reasoning with a large knowledge base for discourse processing. In *Computing Meaning*, 107–127.

Peirce, C. S. 1883. A theory of probable inference. *The Johns Hopkins Studies in Logic* 126–181.

Poole, D. 1993. Probabilistic horn abduction and bayesian networks. *Artificial intelligence* 64(1):81–129.

Schüller, P. 2016. Modeling variations of first-order horn abduction in answer set programming. *Fundamenta Informaticae* 149(1-2):159–207.

Singla, P., and Mooney, R. J. 2011. Abductive markov logic for plan recognition. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*.

Tseitin, G. 1968. On the complexity of derivation in propositional calculus. *Studies in constructive mathematics and mathematical logic* 115–125.

Yamamoto, K.; Inoue, N.; Inui, K.; Arase, Y.; and Tsujii, J. 2015. Boosting the efficiency of first-order abductive reasoning using pre-estimated relatedness between predicates. *International Journal of Machine Learning and Computing* 5(2):114–120.