Two Transformations of Clauses into Constraints and their Properties for Cost-based Hypothetical Reasoning

Yutaka Matsuo[1] and Mitsuru Ishizuka[1]

University of Tokyo, Hongo 7-3-1, Tokyo 113-8656, Japan
matsuo@miv.t.u-tokyo.ac.jp,
WWW home page: http://www.miv.t.u-tokyo.ac.jp/ matsuo

Abstract. This paper describes two ways to transform propositional clauses into mathematical constraints, and gives an overview of mathematical optimization approaches to inference. The first transformation, which translates constraints into linear inequalities, has been applied to cost-based abduction in the past and showed good performance. The second one, which produces nonlinear equalities, is commonly used in other representations, such as SAT. We clarify their differences and advantages, and show the radical performance transition of linear inequalities. We are mainly targeting at cost-based hypothetical reasoning (or abduction), but through preprocessing, the discussion has generality.

1 Introduction

Hypothetical reasoning (or 'abduction') is a useful framework for knowledge-based systems that allows to find explanations for observations [1, 2]. In *cost-based abduction* [3, 4], hypotheses have associated costs, and the cost of a proof is simply the sum of the costs of the hypotheses required to complete that proof. It has been shown in [3] that belief revision in Bayesian networks can be accurately modeled by cost-based abduction. Unfortunately, the computational complexity of computing the minimal cost explanation is NP-hard, even for very basic forms of propositional abduction [5].

Though it is more natural to describe systems in a more expressive language such as first-order logic, propositional logic is still of importance. Prendinger and Ishizuka compile a first-order system description of cost-based abduction to a propositional representation, mainly focusing on model-based diagnosis [6]. An efficient engine for propositional cost-based abduction plays an crucial role in such systems.

Recently, significant progress towards highly efficient inference mechanisms can be seen for propositional cost-based abduction [4, 7, 8]. These methods employ search mechanisms borrowed from mathematical programming, a programming paradigm that may exploit the continuous rather than the discrete value domain. The main idea is to transform Horn clauses into linear constraints and seek the optimal point using linear programming techniques, e.g., the simplex method.

Besides cost-based abduction, several efficient mechanisms have also been developed for other kinds of propositional reasoning, such as satisfiability problems (SAT) or constraint satisfaction problems (CSP). In the SAT case, starting from GSAT [9], a set of powerful new local search heuristics has been developed [10–12]. Especially,

DLM-2000 (discrete Lagrangian method) [13] can solve some of the hardest satisfiability problems of DIMACS benchmark problems.

In this paper, we will describe two types of transformation from propositional clauses to mathematical constraints. The first one, called *transformation* **L**, has been used widely in the context of cost-based abduction, which transforms clauses into linear inequalities. The second one, called *transformation* **NL**, translates clauses into nonlinear equalities, which can be considered as underlying some algorithms for SAT. These two transformations have different characteristics; transformation **L** is likely to find a low-cost solution, while transformation **NL** efficiently searches for a feasible solution. We will show an example in which transformation **L** performs well, but by adding clauses, the performance deteriorates. We also show that transformation **NL** doesn't cause such a radical transition.

The rest of the paper is organized as follows. In the following section, we first briefly describe the preprocessing of a hypothetical reasoning problem in order to obtain a knowledge base with a small number of clauses. Then we explain two transformations of clauses into constraints. Section 3 is devoted to a detailed discussion of the characteristics and the differences of two transformations, including the performance transition of transformation **L**. Finally, we discuss future works and conclude this paper.

2 How to Make Constraints?

A *hypothetical reasoning problem* (HRP) is characterized by a set $\mathcal{G}$ of goals (e.g., observations) to be explained, given background knowledge Σ, e.g., the behavioral model of some device. A *solution* to a HRP is a set of hypotheses $H \subseteq \mathcal{H}$ which, if assumed, would explain the observations.

Definition 1. *A* hypothetical reasoning problem *(HRP) is a quadruple* $HRP = \langle T, I, \mathcal{H}, \mathcal{G} \rangle$*, such that*
(i) T *is a propositional Horn theory, i.e., a set of clauses* t *of the form "*$q \leftarrow p_1 \wedge \ldots \wedge p_n$*" where* $q, p_1, \ldots, p_n$ *are atomic formulas. If* $n > 0$ *then* C *is called a* rule *(or* non-unit *clause), else (*$n = 0$*)* C *is called a* fact *(or* unit *clause). The atom* q *is called the* head *of the clause, the conjunction* $p_1 \wedge \ldots \wedge p_n$ *is called the* body *of the clause.*
(ii) I *is a set of* inconsistency constraints*, i.e., a set of Horn clauses of the form "*$inc \leftarrow h_1 \wedge \ldots \wedge h_n$*" where each* $h_i (1 \leq i \leq n)$ *denotes a hypothesis, and the symbol* inc *denotes the logical constant* falsum*.*
(iii) $\mathcal{H}$ *is a set of atoms that denote element hypotheses ('assumable' states).*
(iv) $\mathcal{G}$ *is a set of goal atoms denoting, e.g., observations.*

Definition 2. *Let* $HRP = \langle T, I, \mathcal{H}, \mathcal{G} \rangle$ *be a hypothetical reasoning problem. A set* $H \subseteq \mathcal{H}$ *is a* solution hypotheses set *(or* solution*) for HRP iff (i) for each* $g \in \mathcal{G}$*:* $T \cup H \vdash g$*, (ii)* $I \cup H \nvdash inc$*.*

Cost-based hypothetical reasoning employs the *minimum cost solution* criterion. It assumes that a numerical weight (or cost) $c(h)$ is assigned to each element hypothesis $h \in \mathcal{H}$ (c is a function from $\mathcal{H}$ to the natural numbers). Then the cost of a solution,

$C(H)$, can be defined as

$$C(H) = \sum_{h \in H} c(h). \tag{1}$$

2.1 Preprocessing

Before applying the transformation, we first compile the HRP, including goal atoms, to a small set of conjunctive normal form (CNF) formulas. The following steps are executed sequentially.

Relevance reasoning Construct the *query-tree* [14] in a top-down (from a goal to leaves) fashion and get T, I, and $\mathcal{H}$ reduced.

Completion Apply completion for each rule $t \in T$ and get augmented T in order to make backward inference possible. We change "$q \leftarrow p$" to "$q \leftarrow p$" and "$p \leftarrow q$," and refer the original rule as a *top-down* rule and the added rule as a *bottom-up* rule, following [3].

Eliminating top-down rules Eliminate top-down rules and get reduced T. This is possible because in case of cost-based abduction, an optimal solution for *semi-induced problems*, which consist of bottom-up rules without top-down rules, can be transformed into the best explanation for the original problem in the linear order of steps with respect to the number of rules under a weak assumption [4]. Thus the original rules in theory T are only utilized in the form of bottom-up rules.

Merging goal atoms and eliminating unit clauses Assign $g \in \mathcal{G}$ `true`, eliminate unit clauses and get reduced T and $\mathcal{H}$. If we assign goal atoms `true`, some clauses in T may turn to be unit. We assign the atom `true` in a unit clause $t \in T$ and eliminate t from T. This process is repeated until no unit clauses are detected in T, or inconsistency detected. This is a simple version of eliminating one-literal clauses in the Davis-Putnam procedure [15].

We now have a new propositional theory T' and a new set of inconsistency constraints I'. We create a new knowledge base *KB*′ of the CNF formulas, consisting of $t \in T'$ and $ic \in I'$. In order to find the (minimum cost) solution to a HRP, we should find the (minimum cost) satisfying assignment to *KB*′. The set of atoms in *KB*′ is denoted as N. Note that the clauses in *KB*′ are not necessarily Horn.

In [3, 4], *WAODAGs* (or, weighted AND/OR directed acyclic graphs) are used to represent cost-based abduction. However, we allow clauses in *KB*′ to be any CNF formulas, thus the following discussion can be applied also to SAT problems under minimum cost solution criterion.

2.2 Transformation into Linear Constraints

First we associate the variable x_p with $p \in N$. We use $x_p = 1$ for "p is `true`," and $x_p = 0$ for "p is `false`."

Assume *KB*′ has the clause,

$$p_1 \vee \neg p_2 \vee \neg p_3. \tag{2}$$

In order for formula (2) to be true, at least one of the literals p_1, $\neg p_2$, and $\neg p_3$ has to be true, equivalently, the mathematical constraint $x_{p1} + (1 - x_{p2}) + (1 - x_{p3}) \geq 1$ has to be satisfied. We define *transformation* **L** as follows.

Definition 3. Transformation L *is a transformation from a clause to a linear inequality constraint, constructed as follows:*
(i) replace the literals p and $\neg p$ by x_p and $(1 - x_p)$, respectively,
(ii) generate the LHS (left-hand side) of an inequality by replacing disjunction ($\vee$) by the arithmetic operation "$+$,"
(iii) generate the inequality "$LHS \geq 1$."

Together with the cost function (1) and 0-1 relaxation, transformation **L** defines *problem L*.

Definition 4. Problem L *is a continuous linear programming problem, where (i) the objective function to be minimized is defined by (1), and (ii) subject to the constraints derived from transformation* **L** *for each clause $c \in KB'$, and (iii) relaxed 0-1 constraints $0 \leq x_p \leq 1$ for each atom $p \in N$.*

Problem L can be applied to other representations than cost-based abduction, such as SAT. However, as Selman describes [16], in most formulations the solution to the linear relaxation of any SAT problem simply sets all the variables to the value "1/2," thus yielding no guidance at all.

But still solving *problem L* is useful in the case of cost-based abduction. Santos, Jr. revealed that 97% of the randomly generated WAODAGs were solved using only linear programming without supplementary branch-and-bound [4]. Although randomly generated problems have some pitfalls [17], we discuss later that there is a region where transformation **L** does work well.

2.3 Transformation into Nonlinear Constraints

Consider (2) again. It is logically equivalent to $\neg(\neg p_1 \wedge p_2 \wedge p_3)$, which means $\neg p_1 \wedge p_2 \wedge p_3$ should be `false`, thus at least one of $\neg p_1$, p_2, and p_3 should be `false`, so that the formula is true. Equivalently we can write down a constraint as $(1 - x_{p1})x_{p2}x_{p3} = 0$. We define *transformation* **NL** as follows.

Definition 5. Transformation NL *is a transformation from a clause to a nonlinear equality constraint, constructed as follows:*
(i) negate the clause,
(ii) replace the literals p and $\neg p$ by x_p and $(1 - x_p)$, respectively,
(iii) make the LHS of an equality replacing conjunction ($\wedge$) by the arithmetic operation "$\times$,"
(iv) make the equality "$LHS = 0$."

Combining the cost function (1) and 0-1 relaxation, we obtain *problem NL*.

Definition 6. Problem NL *is a continuous nonlinear programming problem, where (i) the objective function to be minimized is defined by (1), and (ii) subject to the constraints derived from transformation* **NL** *for each clause* $c \in KB'$*, and (iii) relaxed 0-1 constraint* $0 \leq x_p \leq 1$ *for each atom* $p \in N$.

Problem NL is one of the most simple form of transformations, and it can be considered as underlying some algorithms. For example, Gu has developed a SAT problem model, called *UniSAT*, which transforms a SAT problem into an unconstrained optimization problem on real space [10, 18]. UniSAT7 transforms a CNF $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3)$ into $f = (x_1 - T)^p(x_2 + F)^p + (x_1 + F)^p(x_2 - T)^p(x_3 - T)^p$, where T, F, and p are constants. This method in the case of $T = 1$ and $F = 0$ is nothing but the p-norm penalty method [19] to solve *problem NL*.

In DLM, or discrete Lagrangian method [20], a SAT problem (with n variables and m clauses) is formulated as a discrete constrained optimization problem;
$\text{Minimize}_{x \in \{0,1\}^n} \sum_{i=1}^{m} U_i(x)$, subject to $U_i(x) = 0 \quad \forall i \in \{1, 2, \ldots, m\}$, where $U_i = 0$ if clause i is satisfied, and $U_i = 1$ if clause i is not satisfied. We can easily see that the constraint constructed by transformation **NL** satisfies the requirements of U_i. Thus, applying continuous Lagrangian methods to *problem NL*, similar method to DLM can be derived (though some devices are needed e.g., to force the updated value of x should be 0 or 1 in each iteration). The overview of such subgradient algorithms are seen in [21].

In summary, transformation **L** is efficiently used in the context of cost-based abduction, while transformation **NL** underlies some SAT algorithms to solve highly constrained problems. The approach here follows the treatment of pseudo-boolean optimization problems [22].

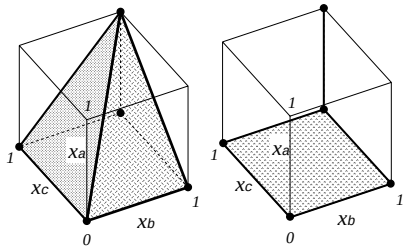
3 Discussion of Two Transformations

3.1 Difference of Two Transformations

One major difference between transformation **L** and transformation **NL** is that transformation **L** gives a necessary condition for variables to be a satisfying assignment, whereas transformation **NL** gives a necessary and sufficient condition.

Given a truth assignment to atoms, obviously we can satisfy constraints of either transformation **L** or transformation **NL** by substituting 1/0 to the corresponding variables according to the truth value. Conversely, given values of variables which satisfy constraints of transformation **NL**, we can construct a truth assignment as follows.

Theorem 1. *Given a set of clauses* KB' *and a set of atoms* N*, if the variables* x_p $(p \in N)$ *satisfy constraints derived from transformation* **NL** *of the clauses* $c \in KB'$*, we can construct a truth assignment satisfying the clauses* $c \in KB'$ *as follows. For all the atoms* $p \in N$*, (i) if* $x_p = 1$*, then set* p `true`*, (ii) if* $x_p = 0$*, then set* p `false`*, (iii) otherwise, set* p *arbitrarily* `true` *or* `false`.



Transformation L **Transformation NL**

Fig. 1. Feasible region of "$(\neg a \vee b) \wedge (\neg a \vee c)$."

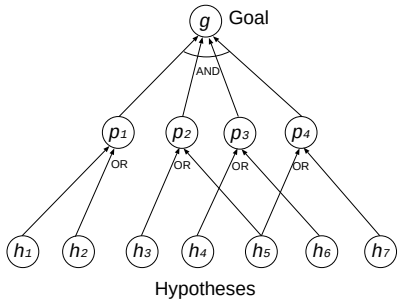


Fig. 2. Set covering problem.

Proof. The proof is obvious by Definition 5 □.

On the other hand, this is not the case for transformation **L**, even if we round variables, e.g., if $x_p \geq 0.5$ then set p `true`, otherwise set p `false`. Despite of this disadvantage, *problem L* has some merits. The most important one is that it may produce the optimal solution quite rapidly, only by simplex method. Besides, (i) it may provide the guidance for the 0-1 optimal solution, (ii) it provides the lower bound of cost of the 0-1 optimal solution, and (iii) it shows the unsatisfiability of *KB′* if *problem L* is infeasible. The feasible region of transformation **L** is a convex polygon as shown in Fig. 1, thus the search point proceeds easily toward the gradient direction of the objective function. However, transformation **NL** produces the tightly constrained feasible region, thus to proceed toward the direction to decrease the objective function (not Lagrangian function) is harder than transformation **L**.

3.2 Performance Transition of Transformation L

It has been believed that *problem L* works well on HRPs, however, the performance depends considerably on the 'hardness' of the problem. Here we pick up a very simple but sufficiently suggestive example to show the transition of performance by transformation **L**.

Fig. 2 is a set covering problem represented by a HRP. A solution for the HRP is a set of hypotheses which covers all the intermediate propositions. For example,

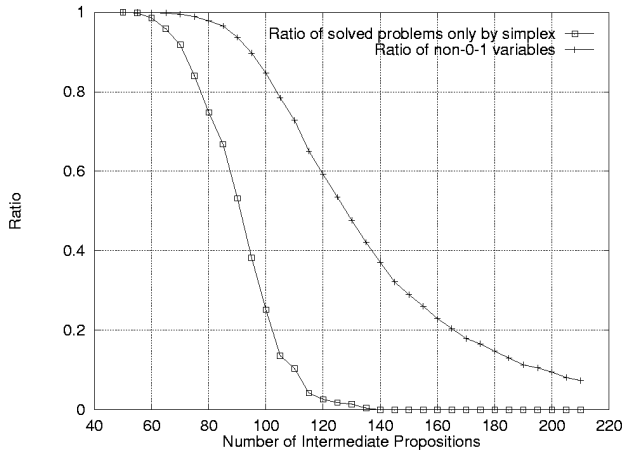


Fig. 3. Ratio of solved problems with 100 hypotheses by solving *problem L*.

$\{h_2, h_5, h_6\}$ is a solution to prove the goal. Here, the cost of each hypothesis is set to be 1, and each intermediate proposition has two OR children randomly chosen. If we add intermediate propositions while keeping the number of hypotheses constant, then the number of solutions decreases, i.e., the problem gets harder. For example, if we add a new intermediate proposition p_5, such that "$p_5 \leftarrow h_4 \vee h_7$ " and "$g \leftarrow p_1 \wedge \ldots \wedge p_4 \wedge p_5$," then $\{h_2, h_5, h_6\}$ is not a solution anymore. Yet this problem has always at least one solution, to set all hypotheses `true`. In this manner, we get series of problems gradually getting more tightly constrained.

In the experiment, starting from 100 hypotheses and 50 intermediate propositions, we gradually add the intermediate propositions. Fig. 3 shows the ratio of solved problems only by solving *problem L* (using simplex method). One dot represents the average of 500 instances. Almost all the problems can be solved only by simplex method when the number of intermediate propositions m is less than 60, however few problems can be solved when m exceeds 140. At $m = 140$, 63% of the variables are set to 1/2, and provide no guidance at all. In the case of 1000 hypotheses starting from 500 intermediate propositions, the drop is even steeper as shown in Fig. 4.

Not only for this set covering problem but also for other problem instances, such as shortest path problem and assignment problem, we have found the same phenomenon. By adding clauses, HRPs are less likely to be solved only by solving *problem L*.

Lastly, the curves in Fig. 3 and 4 remind us of the *phase transition* phenomenon [23, 24]. For example, in the case of random 3-SAT instances with 50 variables, if the ratio of clauses-to-variables is less than 3.5, the formula is almost certainly satisfiable, while if the ratio is over 5.2, almost all formulas are unsatisfiable. This phase transition focuses on the likelihood of satisfiability, however, ours focuses on the likelihood of solvability by solving *problem L*. We expect our performance transition has some correspondence with the phase transition of satisfiability, but further study is needed to derive some conclusions.

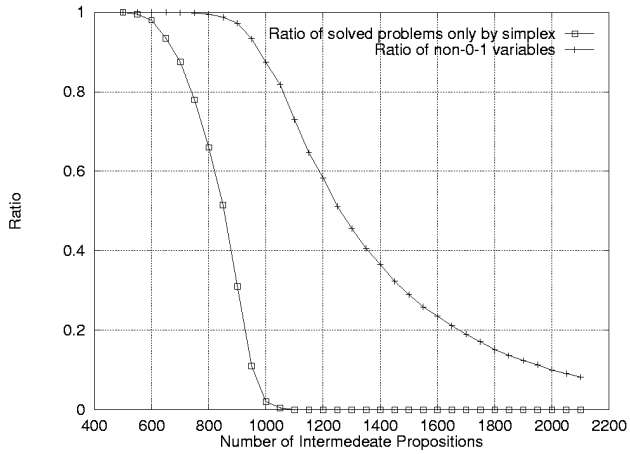


Fig. 4. Ratio of solved problems with 1000 hypotheses by solving *problem L*.

3.3 Performance of Transformation NL

As for *problem NL*, to satisfy the constraints is equivalent to find a satisfying assignment of the clauses from Theorem 1. Therefore the performance can't be evaluated in the same way as in the case of *problem L*. Instead, we implemented DLM to solve *problem NL* and measured the computational time.

Fig. 5 and 6 show the computational time of DLM to solve the set covering problem with different numbers of intermediate propositions. In every case, a solution is found. The computational time grows gradually, and no rapid change of performance is observed. Therefore we can conjecture transformation **NL** is suitable even for seriously constrained problems. This supports the fact that DLM and other algorithms show good performance on SAT problems.

4 Conclusion

The relation between logic and 0-1 integer programming, or OR (operations research), has been studied widely and intensively in the recent years. Especially, the latest good overviews might be [25] and [26], both emphasizing the possibilities of the integration of AI and OR.

In this paper, we have clarified two transformations from clauses to constraints. Transformation **NL** is appropriate for finding a feasible solution. Transformation **L** has the merit in that it may produce the optimal solution quite rapidly, however, it can be applied efficiently only to the "easy" problem instances because the performance rapidly deteriorates as the problem gets hard. Future works will combine the two transformations and develop an algorithm to seek the optimal solution, mainly targeting on an intermediate region between "easy" and "hard."

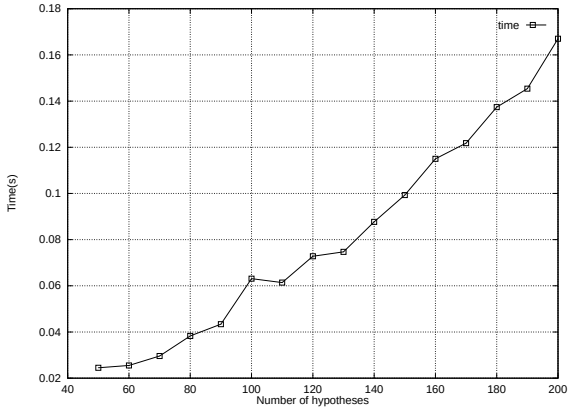


Fig. 5. Computational time to solve problems **NL** with 100 hypotheses by DLM.

References

1. Poole, D.: A methodology for using a default and abductive reasoning system. International Journal of Intelligent Systems **5** (1990) 521–548
2. Kakas, A., Kowalski, R., Toni, F.: The role of abduction in logic programming. In: Handbook of Logic in Artificial Intelligence and Logic Programming. Oxford University Press (1995)
3. Charniak, E., Shimony, S.E.: Cost-based abduction and MAP explanation. Artificial Intelligence **66** (1994) 345–374
4. Santos, Jr., E.: A linear constraint satisfaction approach to cost-based abduction. Artificial Intelligence **65** (1994) 1–27
5. Eiter, T., Gottlob, G.: The complexity of logic-based abduction. Journal of the Association for Computing Machinary **42** (1995) 3–42
6. Prendinger, H., Ishizuka, M.: Qualifying the expressivity / efficiency tradeoff: Reformation-based diagnosis. In: Proceedings 16th National Conference on Artificial Intelligence (AAAI–99). (1999) 416–421
7. Ohsawa, Y., Ishizuka, M.: Networked bubble propagation: A polynomial-time hypothetical reasoning method for computing near-optimal solutions. Artificial Intelligence **91** (1997) 131–154
8. Ishizuka, M., Matsuo, Y.: SL method for computing a near-optimal solution using linear and non-linear programming in cost-based hypothetical reasoning. In: Proceedings 5th Pacific Rim Conference on Artificial Intelligence (PRICAI-98). (1998) 611–625
9. Selman, B., Levesque, H., McAllester, D.: A new method for solving hard satisfiability problems. In: Proceedings 9th National Conference on Artificial Intelligence (AAAI-92). (1992) 440–446
10. Gu, J.: Local search for satisfiability (SAT) problem. IEEE Transactions on Systems, Man, and Cybernetics **23** (1993) 1108–1129
11. Selman, B., Kautz, H., Cohen, B.: Noise strategies for improving local search. In: Proceedings 11th National Conference on Artificial Intelligence (AAAI-94). (1994) 337–343
12. Frank, J.: Learning short-term weights for GSAT. In: Proceedings 15th International Joint Conference on Artificial Intelligence (IJCAI-97). (1997) 384–391

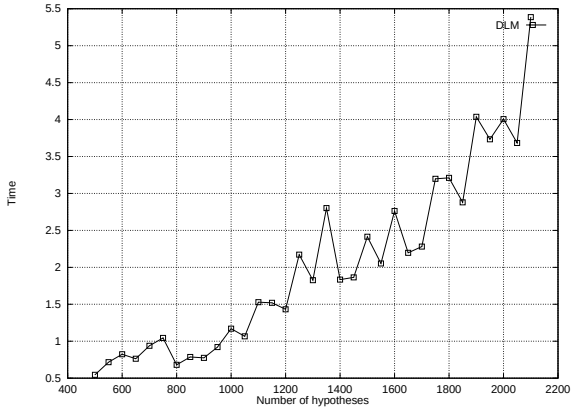


Fig. 6. Computational time to solve problems **NL** with 1000 hypotheses by DLM.

13. Wu, Z., Wah, B.W.: An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems. In: Proceedings 17th National Conference on Artificial Intelligence (AAAI-2000). (2000) 310–315
14. Levy, A.Y., Fikes, R.E., Sagiv, Y.: Speeding up inferences using relevance reasoning: a formalism and algorithms. Artificial Intelligence **97** (1997) 83–136
15. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the Association for Computing Machinary **7** (1960) 201–215
16. Selman, B., Kautz, H., McAllester, D.: Ten challenges in propositional reasoning and search. In: Proceedings 15th International Joint Conference on Artificial Intelligence (IJCAI-97). (1997) 50–54
17. Mitchell, D., Levesque, H.: Some pitfalls for experimenters with random SAT. Artificial Intelligence **81** (1996) 111–125
18. Gu, J.: Global optimization for satisfiability problem. IEEE Transactions on Knowledge and Data Engineering **6** (1994) 361–381
19. Nemhauser, G.L., Kan, A.H.G.R., Todd, M.J., eds.: Optimization: Handbooks in Operations Research and Management Science, Volume 1. North-Holland, Amsterdam (1989)
20. Shang, Y., Wah, B.W.: A discrete Lagrangian-based global-search method for solving satisfiability problems. Journal of Global Optimization **12** (1998)
21. Schuurmans, D., Southey, F., Holte, R.C.: The exponentiated subgradient algorithm for heuristic boolean programming. In: Proceedings 17th International Joint Conference on Artificial Intelligence (IJCAI-01). (2001) 334–341
22. Hammer, P., Simeone, B.: Quadratic functions of binary variables. In: Combinatorial Optimization, Lecture Notes in Mathematics. Volume 1403. (1989)
23. Selman, B., Mitchell, D., Levesque, H.: Generating hard satisfiability problems. Artificial Intelligence **81** (1996) 17–29
24. Gomes, C., Selman, B., Crato, N.: Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning **24** (2000) 67–100
25. Gomes, C.: Structure, duality, and randomization: Common theme in AI and OR. In: Proceedings 17th National Conference on Artificial Intelligence (AAAI-2000). (2000) 1152–1158
26. Hooker, J.N.: Logic, optimization, and constraint programming. (INFORMS journal on Computing) To appear.