Proc. 10th Canadian Conf. on AI, Banf, pp.176-186 (1994)

A Polynomial-time Hypothetical Reasoning employing an Approximate Solution Method of 0-1 Integer Programming for Computing Near-optimal Solution

Mitsuru Ishizuka and **Tomoki Okamoto**+
Dept. of Information & Communication. Eng.
Faculty of Engineering, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113, Japan

+Presently with Tokyo Electric Power Co.

Abstract

A hypothetical reasoning is an important knowledge system's framework because of its theoretical basis and its usefulness for practical problems including diagnosis, design, etc. One crucial problem with the hypothetical reasoning is, however, its slow inference speed. In order to achieve practical or tractable inference speed, we apply an approximate solution method of 0-1 integer programming to a weight-based hypothetical reasoning, where a numerical weight is assigned to each possible element hypothesis and the optimal solution hypothesis set with minimal sum of its element hypotheses' weights is searched. In this method, we regard all described knowledge as constraints. To narrow down the search space, we first extract restricted knowledge relevant to the proof of a given goal. Then, we transform the restricted knowledge into inequatlities to apply 0-1 integer programming. While the computational complexity of the hypothetical reasoning is NP-complete or NP-hard, an approximate solution method of 0-1 integer programming allows polynomial inference time for finding a near-optimal solution hypothesis.

1. Introduction

A hypothetical reasoning is an important framework for advanced knowledge-based systems because of its theoretical basis and its usefulness for many practical problems including diagnosis, design, etc. [Poole 87, 88, Ishizuka 90, Makino 90]. It is an abductive inference mechanism for finding consistent solution hypotheses satisfying given constraints. One crucial problem with the hypothetical reasoning is its slow inference speed because a non-monotonic inference is needed due to the use of hypothetical or defeasible knowledge.

In order to overcome this problem, the authors' group has developed so far several fast hypothetical reasoning methods, e.g., 1) fast hypothetical reasoning using inference-path network [Ito 91, Ishizuka 91] which includes ATMS mechanism [deKleer 86], 2) fast hypothetical reasoning for predicate-logic knowledge-base [Kondo 91] which employs a deductive database technique, 3) fast hypothetical reasoning using analogy [Ishizuka 93], and 4) knowledge-base compilation method [Tsuruta 91, 92]. Since the computational complexity of the hypothetical reasoning is NP-complete or NP-hard [Kautz 89, Bylander 89, Stillman 90], we cannot overcome the wall of exponential inference time with respect to problem size as long as we use ordinary inference methods. The above methods 2) and 3), for example, use analogical reasoning and knowledge-base compilation, respectively, to overcome this inference speed limit.

In this paper, we present another efficient hypothetical reasoning method based on an approximate solution method of 0-1 integer programming. Here, we consider a weight-based or cost-based hypothetical reasoning [Charniak 90], where a numerical weight is assigned to each possible element hypothesis and an optimal solution hypothesis set is searched which has the minimum weight sum of the element hypotheses. This framework is useful, for example, for finding the most possible diagnosis or the least expensive design satisfying given constraints.

We regard the described logical knowledge as constraints and transform them into inequalities to apply the 0-1 integer programming method. Before this application, we find the restricted portion of knowledge relevant to the proof of a given goal so as to narrow down the search space. As a result, we show that the approximate solution method of 0-1 integer programming is very effective to find a near-optimal solution hypothesis in polynomial time.

Recently, it is recognized that the combination of mathematical programming techniques with knowledge-based processing is useful to achieve an efficient inference [Hooker 88, Dhar 90, Charniak 92]. However, the use of an approximate solution method was not considered in [Chaniak 92]. While its use was considered in [Dhar 90], a preprocessing of restricting the scope of knowledge being introduced in this paper had not been incorporated. Since the computational complexity of 0-1 integer programming is still NP-complete, it is necessary to incorporate a preprocessing of reducing the number of variables for 0-1 integer programming and to apply an approximate solution method, for achieving an efficient and tractable inference of the hypothetical reasoning.

Probabilistic search methods based on simulated annealing(SA) [Kirkpatrik 83] or genetic algorithm(GA) [Goldberg 89] are also exploited recently in AI area for efficiently finding near-optimal solutions. The search of our inference method is different from these probabilistic methods; our method uses guiding information obtained by the efficient simplex method for a corresponding problem relaxed from 0-1 integer domain to real domain, and excecutes a local search for a 0-1 optimal solution around the optimal solutuion in real domain.

Also, the efficiency of the local search for CSP (constraint satisfaction problem) or SAT (satisfiability testing) is recently recognized such as in a heuristic repair method [Minton 92] and GSAT [Selman 93a, 93b]. Our method differs from these methods in that the efficient simplex method is used to obtain a good initial guess and analog-value points between (or sometimes outside) 0-1 vertex points are considered in the local search process. In addition, our method can compute a near-optimal solution rather than a simple single solution satisfying the constraints. The use of unconstrained nonlinear programming for CSP or SAT is shown in [Gu 93]. This local search process is conceptually similar to our method; however, it does not provide any mechanism to determine a good initial search point. Thus our method, while it is described in the context of the hypothetical reasoning in this paper, may indicate a new efficient search for wider problem solving under declarative knowledge.

2. Logic-based Hypothetical Reasoning

The hypothetical reasoning in this paper is a logic-based one [Poole 87, 88, Ishizuka 91, Ito 91], where knowledge is divided into two categories, i.e., background knowledge (or fact in [Poole 87, 88]) and hypothesis. Background knowledge denoted by Σ has no possibility of inconsistency, whereas the hypothesis denoted by H has the possibility of contradiction with other hypotheses, and thus is defeasible knowledge.

As illustrated in Fig.1, the basic behavior of this hypothetical reasoning is as follows. When a goal G is given, the system first tries to prove this goal from background knowledge. If it fails, then the system selects a subset of the hypotheses so that the given goal is proved from the union of bacground knowledge and this hypothesis subset, which should be consistent with background knowledge. This consistent subset of hypotheses becomes a solution for the given goal in the hypothetical reasoning system. The generation of this consistent hypothesis subset can be viewed as abduction.

The structure of above hypothetical reasoning can be summarized to find a solution h satisfying

- $h \subseteq H$ (h is a subset of H)
- $\Sigma \cup h \vdash G$ (G can be derived from $\Sigma \cup h$), and
- $\Sigma \cup h \nvdash \Box$ ($F \cup h$ is consistent, $\Box$: empty clauses),

where Σ, H and G are background knowledge, possible hypotheses and a given goal, respectively.

In this paper, we restrict the knowledge representation to propositinal Horn clauses, since our

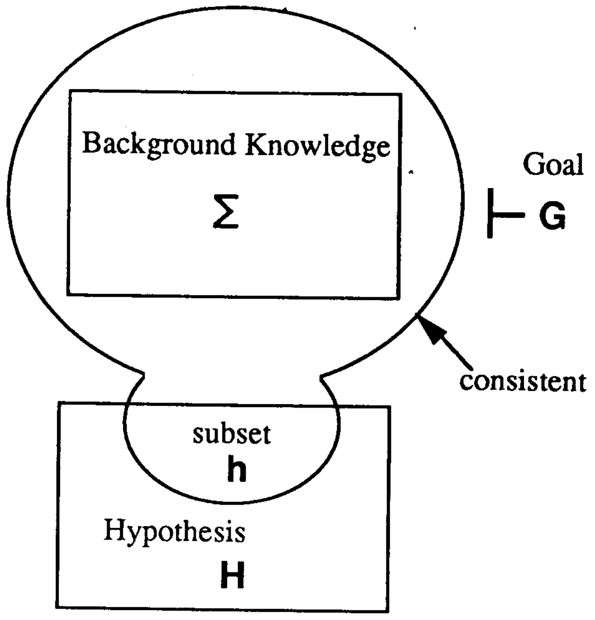

Fig. 1 Basic structure of logic-based hypothetical reasoning.

main concern here is an efficient inference mechanism. Since the logical negation of an atom cannot be expressed with Horn clauses, we introduce an atom called "inc" to denote inconsistency among hypotheses, such as,

$\text{inc} \leftarrow h_1, h_2.$,

which says that h_1 and h_2 cannot be coexist in an environment.

Rule-type hypotheses are allowed in general in our hypothetical reasoning system. They are, however, transformed by a preprocessing into newly introduced single-atom hypotheses and modified background knowledge. For example, a rule-type hypothesis '$p \leftarrow q.$' will be transformed by introducing a new atom 'r' into,

background knowledge $p \leftarrow q, r.$, and
hypothesis $r.$

In this case, the hypothesis '$p \leftarrow q.$' can be interpreted as being included in a solution hypothesis set if 'r' is included in the solution hypothesis set. With this preprocessing, all the hypotheses become unit clauses (single atoms).

There are often cases that a goal with a non-Horn clause such as,

$s_1 \,\&\, \text{---} \,\&\, s_m \leftarrow t_1, \text{---}, t_n.$

is given to the system; t_1,---, t_n and s_1, ---, s_m may be, for example, input and output observations, respectively, in a fault diagnosis problem, or an input-output specification in a circuit design problem. In these cases, by introducing an atom 'g' indicating an inference goal, we add

$g \leftarrow s_1, \text{---}, s_m$
$t_1.$
:
$t_n.$

into the knowledge-base as background knowledge, and then try to prove 'g'.

There exist plural solution hypotheses sets in general. In many cases, an optimal solution hypothesis set with the minimum weight sum of its element hypotheses is required as the solution. We consider this type of a weight-based or cost-based hypothetical reasoning in this paper. A weight for each element hypothesis is defined in our system , for example, as,

hyp (h_1, 2),

where 2 is a numerical weight assigned to hypothesis h_1.

Figure 2 dipicts the functions of the hypothetical reasoning system described in this paper.

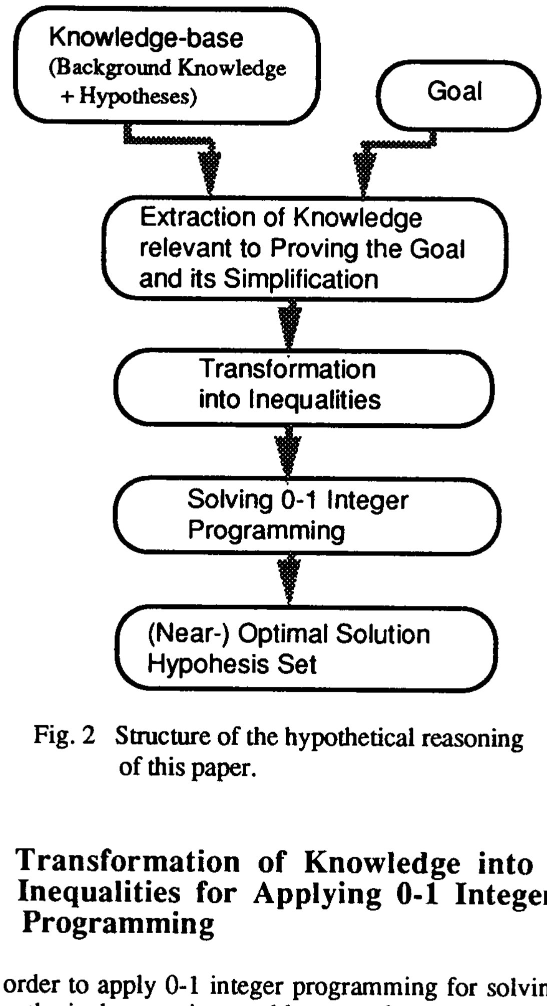
Knowledge-base
(Background Knowledge + Hypotheses)
Goal
Extraction of Knowledge relevant to Proving the Goal and its Simplification
Transformation into Inequalities
Solving 0-1 Integer Programming
(Near-) Optimal Solution Hypohesis Set

Fig. 2 Structure of the hypothetical reasoning of this paper.

3. Transformation of Knowledge into Inequalities for Applying 0-1 Integer Programming

In order to apply 0-1 integer programming for solving a hypothetical reasoning problem, we have to transform described knowledge into linear inequalities. We first describe this transformation method employed in our system.

We assume closed world assumption (CWA) [Clark 78] for the knowledge-base; i.e., we regard knowledge not explicitly described in the knowledge-base as false. In this situation, we can interpret a fact unable to be deductively proved from the knowledge-base as false; this is called 'negation as failure'. For example, suppose that

a ← b, c., a ← d., e ← f., b., f.,

are described in a knowledge-base. In this case, since 'a' cannot be proved to be true from this knowledge-base, we interpret '¬a' is true.

To give a model theoretic semantics to the negation as failure, we introduce the concept of completion [Clark 78, Lloyd 84]. Simply speaking, we rewrite 'e ← f.' into 'e ↔ f.' ('e' is true iff 'f' is true). Also, if an atom 'd' appears in the body of a Horn clause and there exist no Horn clause with the head of 'd', we add '¬d' to the knowledge-base. By this completion procedure, the above-mentioned illustrative knowledge-base becomes

$a \leftrightarrow ((b \wedge c) \vee d)$, b, ¬c,
¬d, $e \leftrightarrow f$, f.

In the case of propositional Horn clauses, it is known that the completion and the closed world assumption are equivalent.

In the framework of our hypothetical reasoning, we can assume the closed world assumption for background knowledge and apply the completion. Since it is unknown whether a hypothesis is true or not in a certain environment, we regard it as a variable to be determined for satisfying a given goal under the background knowledge. That is, we regard the hypothesis as a 0-1 variable in 0-1 integer programming.

Generally speaking, logic formulae in the knowledge-base can be regarded as constraints. Transforming these constraints into inequalities, we can apply 0-1 integer programming for obtaining a solution. Here, we consider a method of transforming the completed knowledge-base into inequalities. (Another transformation method for uncompleted knowledge-base is shown in [Hooker 88].)

Firstly, modeling the propositional logic formulae by Boolean algebraic equations, we rewrite the truth value (true/false) of an atom into 1/0, and equivalence symbol (↔) into equality symbol (=). Then, we can reduce all the knowledge except hypotheses into one of the following forms.

(1) $p = q_1 \vee \cdots \vee q_n$,
(2) $p = (q_1 \wedge \cdots \wedge q_n) \vee r$
where $p, q_i, r \in \{0,1\}$, $i = 1,2, \cdots, n$,
(3) $p = 1$ if p is defined,
(4) $p = 0$ if $\neg p$ is defined.

Next, we transform above Boolean equations of (1) and (2) into equivalent linear inequalities. These equivalent linear inequalities are not unique; however, we adopt here,

$$(1')\ \frac{q_1 + \cdots + q_n}{n} \leq p \leq q_1 + \cdots + q_n$$

$$(2')\ \frac{q_1 + \cdots + q_n + nr - (n-1)}{2n} \leq p \leq \frac{q_1 + \cdots + q_n + nr}{n}$$

for (1) and (2), respectively. All the rule-type knowledge can be expressed by these linear inequalities, if necessary, by introducing supplementary variables. The atom 'inc' indicating inconsistency is transformed into 'inc=0', and the atom indicating the goal is assigned 1 since it is a constraint to be satisfed.

With above procedures, the hypothetical reasoning with the propositional logic expression can be reformulated into 0-1 integer programming; among 0-1 integer solutions satisfying all the constraints, the variables with 1 become to represent a solution hypothesis set. Setting the weight sum of element hypotheses to the objective function of the 0-1 integer programming, we can obtain the optimal solution hypothesis set by calculating the optimal solution of the 0-1 integer programming.

4. Extraction of Knowledge Relevant to Proving a Goal and its Simplification

With above-mentioned method, the optimal solution hypothesis set in the hypothetical reasoning can be computed in principle by the 0-1 integer programming method. However, if we transform all knowledge in the knowledge-base into inequalities and apply 0-1 integer programming, it becomes quite inefficient because the number of 0-1 variables becomes large. Practical performance cannot be attained by this simple application of 0-1 integer programming for practical-scale knowledge-bases, since, different from linear programming in real domain, the speed of integer programming is not sufficient.

For the hypothetical reasoning with propositional Horn clauses, it is possible to efficiently extract limited knowledge relevant to proving a given goal, while leaving the synthesis of necessary hypotheses to a later process, by the same method as one used in a fast hypothetical reasoning using inference-path network [Ito 91, Ishizuka 91]. The extracted knowledge can be further simplified or compiled efficiently. These procedures can be constructed on the basis of a linear-time algorithm of satisfiability testing for propositional Horn clauses [Dowling 84]. Since the computational complexity of 0-1 integer programming is NP-complete and its computational time increases exponentially with respect to the number of variables, this type of preprocessing for reducing the variables is necessary for achieving a highly efficient computation.

Our preprocessing consists of three processes. The first process is the formation of a goal-directed initial inference-path, and the second is its simplification. The third one is the extraction of relevant constraint knowledge indicating inconsistent combinations among hypotheses (hereinafter, inconsistency knowledge). For illustration purpose, we consider a knowledge-base including hypotheses $h_1 \sim h_{13}$ with numerical weights and a goal (a, b) shown in Fig. 3.

An initial inference-path network can be formed by a backward inference originated from the given goal. This inference-path network becomes to contain all relevant knowledge possibly to contribute to the proof of the goal. The unit clauses of background knowledge and element hypotheses are placed at the leaf nodes of this network. For the knowledge-base and the goal of Fig.3, the initial inference-path network of Fig.4 can be formed, for example. Knowledge such as $p \leftarrow e, h_{11}.$, $q \leftarrow k, l.$ ----, and the element hypotheses of $h_9 \sim h_{13}$ are not included in this network, since they are irrelevant to proving the goal (a, b) in this case.

In the simplification process of the inference-path network, 'true' state at the nodes corresponding to unit-clause background knowledge, and 'false' state at the nodes with no possibility of turning into 'true' because of lacking their child hypothesis nodes are propagated upward. That is, we assign 'true-by-hypothesis' state to all the intermediate nodes in the inference-path network except ones corresponding to the unit clauses of background knowledge. We also assign 'true-by-hypothesis' state to the hypothesis nodes. Then, the 'true' and 'false' states are propagated upward by changing the 'true-by-hypothesis' state of an AND node to 'true' state if all its AND-connected child nodes are in 'true' state and to false' state if one of them is in 'false' state, and by changing the 'true-by-hypothesis' state of an OR node to 'true' state if one of its OR-connected child nodes is in 'true' state and to 'false' state if all of them are in 'false' state.

In the illustrative initial inference-path network of Fig.4, since node 'f' is in 'true' state, node 'c' having this 'f' as its OR-connected child node turns into 'true' state. As a result, at node 'a' having this node 'c' as its AND-connected child node, we don't need to consider this node 'c' and its child nodes any more in the synthesis process of necessary hypotheses and need to consider only child node 'd'. Furthermore, node 'n' in Fig.4 can be identified as 'false' state, since it doesn't have any child hypothesis nodes and thus has no possibility of turning into 'true' state. Then, node

Background Knowledge		Hypotheses with weights	Goal
$a \leftarrow c, d.$	$l \leftarrow h_7.$	hyp (h_1, 1)	(a, b)
$b \leftarrow e.$	$l \leftarrow h_8.$	hyp (h_2, 2)	
$b \leftarrow i.$	$m \leftarrow h_9.$	hyp (h_3, 3)	
$c \leftarrow f.$	$m \leftarrow h_{10}.$	hyp (h_4, 1)	
$c \leftarrow j.$	$p \leftarrow e, h_{11}.$	hyp (h_5, 3)	
$c \leftarrow l.$	$q \leftarrow k, l.$	hyp (h_6, 1)	
$d \leftarrow h_1, k.$	$r \leftarrow q, h_{12}.$	hyp (h_7, 2)	
$e \leftarrow h_2.$	$r \leftarrow m, h_{13}.$	hyp (h_8, 4)	
$e \leftarrow l.$		hyp (h_9, 1)	
$i \leftarrow m, n.$	$inc \leftarrow h_1, h_4.$	hyp (h_{10}, 2)	
$f.$	$inc \leftarrow h_2, h_6.$	hyp (h_{11}, 1)	
$j \leftarrow h_3.$	$inc \leftarrow h_5, h_7.$	hyp (h_{12}, 2)	
$j \leftarrow h_4.$	$inc \leftarrow h_9, h_{11}.$	hyp (h_{13}, 3)	
$k \leftarrow h_5.$	$inc \leftarrow h_{10}, h_{13}.$		
$k \leftarrow h_6.$	$inc \leftarrow h_{11}, h_{12}.$		

Fig. 3 An example of knowledge-base with hypotheses and an inference goal to be satisfied.

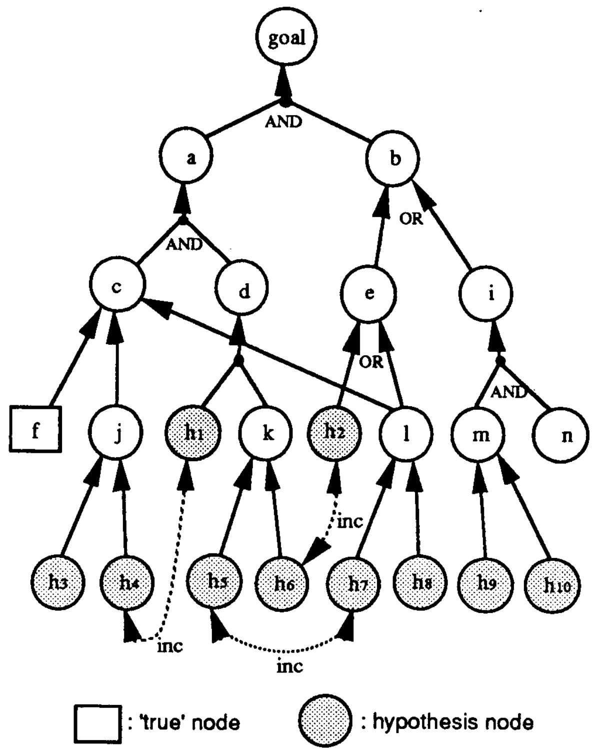

Fig.4 Agoal-directed initial inference-path network.

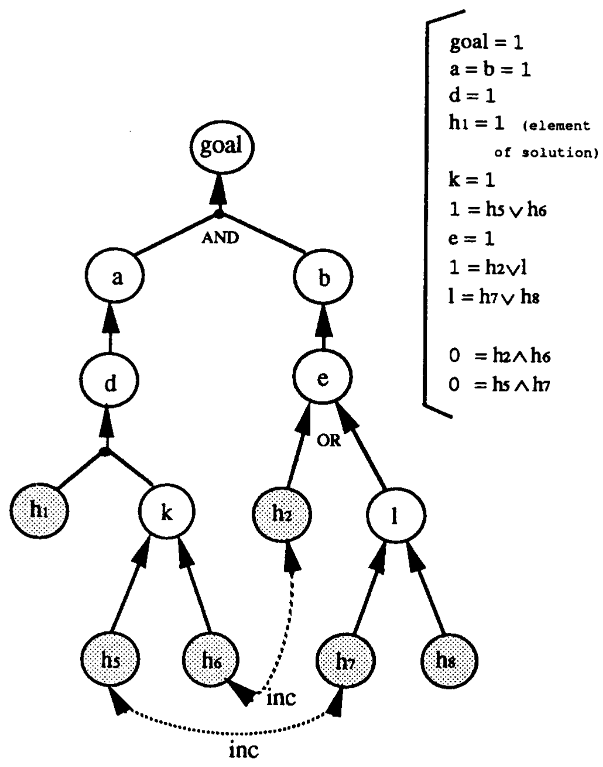

Fig. 5 A simplified inference-path network and extracted constraint Boolean equations (right side).

'i' also becomes 'false' state because it has this node 'n' as its AND-connected child node. Thus, at node 'b' having this node 'i' as its OR-connected child node, we don't need to consider node 'i' any more in the synthesis process of necessary hypotheses and need to consider only child node 'e'. As a result, a simplified inference-path network shown in Fig.5 is obtained.

Moreover, since it is a constraint that the goal becomes 'true' if a solution exists, we can determine the state of AND-connected child nodes by propagating this constraint downward. For example, in the inference-path network of Fig.5, nodes 'a' and 'b' are required to be 'true'; as a consequence, node 'd', node 'k' and hypothesis node 'h_1' are also required to be 'true'. In the same manner, it is necessary for node 'e' to be 'true'. Thus we can obtain simplified constraint equations shown in the right side of Fig.5. Here, '$h_1=1$' means that the element hypothesis 'h_1' should be adopted to satisfy the given goal; otherwise, the goal cannot be satisfied. We can thus reduce the number of variables for 0-1 integer programming.

The extraction of relevant inconsistency knowledge can be performed by selecting only inconsistency knowledge in which every body atom is appeared in the simplified inference-path network as hypothesis or intermediate node. We can ignore other inconsistency knowledge in the case of the given goal.

By the above procedures, the simplified constraint (Boolean) equations shown in the right side of Fig.5 are obtained for the case of Fig.3. Excluding the variables already determined to 0 or 1, we can have simplified constraint equations with only five variables as,

$$1 = h_5 \vee h_6,$$
$$1 = h_2 \vee h_7 \vee h_8,$$
$$0 = h_2 \wedge h_6,$$
$$0 = h_5 \wedge h_7.$$

We transform these constraint equations into linear inequalities according to the method described in section 3. The objective function z for integer programming can be set based on the weights of the element hypotheses, for example, in this case as,

$$z = h_1 + 2h_2 + 3h_5 + h_6 + 2h_7 + 4h_8$$
$$= 1 + 2h_2 + 3h_5 + h_6 + 2h_7 + 4h_8,$$

Then, we can apply 0-1 integer programming for solving the weight-based hypothetical reasoning.

5. Applying Exact and Approximate Solution Methods of 0-1 Integer Programming and their Evaluation

For solving 0-1 integer programming, we have applied two exact methods and one approximate method, i.e., 1) all integer method, 2) implicit enumeration method [Balas

65], and 3) pivot and complement method [Balas 80]. See [Greenberg 71, Garfinkel 72, Konno 81], for example, for general discussion on integer programming methods.

There are two main approaches in the exact solution methods for 0-1 integer programming; they are cutting plane method and branch-and-bound method. The all integer method is a variant of the cutting plane method. The implicit enumeration method is based on the branch-and-bound method. Both methods partially employ an efficient linear programming method, i.e., simplex method in their processes. It is recognized in general that the implicit enumeration method is the most efficient among currently available exact solution methods.

On the other hand, the pivot and complement (P&C) method is an efficient approximate solution method for finding a near-optimal solution in polynomial time. Integer constraint is first relaxed in this method to find an optimal solution in real domain by employing the simplex method. Then, by repeating the change of bases (pivot operation) so as to decrease the degree of non-integer index and by rounding the assignments to variables into 0 or 1 while allowing the slight increase of the objective function value, the P&C method finds a feasible integer solution. In the next step, the P&C method executes a local search around this feasible integer solution for finding a better 0-1 integer solution (complement operation).

For example, the problem illustrated in section 4 and in Figs.3-5 becomes to the following 0-1 integer programming problem after removing unnecessary equations because of already détermined variables.

$$h_5 + h_6 \geq 1$$
$$h_2 + h_7 + h_8 \geq 1$$
$$-h_2 - h_6 \geq -1$$
$$-h_5 - h_7 \geq -1$$

where

$$h_2, h_5, h_6, h_7, h_8 \in \{0,1\},$$

and the objective function

$$z = 1 + 2h_2 + 3h_5 + h_6 + 2h_7 + 4h_8$$
$$\Rightarrow \text{ minimize.}$$

In this example, together with already determined element hypothesis h_1, a final solution will be obtained as (h_1, h_6, h_7) with the minimum value 4 of the objective function.

Hypothetical reasoning systems employing above-described 0-1 integer programming methods have been implemented in C language and their performance was evaluated. Fault diagnosis problems of digital circuits were used as examples in experiments. The CPU time on Sun4/370 was measured with respect to several sizes of the digital circuits. The CPU time here includes the extraction of relevant knowledge, its simplification, transformation into inequalities, and 0-1 integer programming. Table 1 shows the experimental results.

The time expressed as simplification in Table 1 is the time spent for the extraction of knowledge relevant to a given goal and its simplification; this is within 0.1 sec in the used examples and is very fast. Figure 6 depicts these measurements against the number of possible

Table 1 Experimental performance of hypothetical reasoning systems employing 0-1 integer programming methods. (- indicates the cases that a solution was not obtained in a pre-determined time limit.)

Example No.	Number of Element Hypotheses Before Simplificatin → After Simplification	CPU time [sec]				Value of Objective Function	
		Simplification	All Integer Method	Implicit Enumeration Method	Pivot and Complement Method	Optimal Value	Value by Pivot and Complement Method
(1)	15→10	0.01	0.02	0.01	0.04	9	9
(2)	30→24	0.02	75.53	9.37	2.25	14	14
(3)	45→38	0.03	-	1301.04	12.01	19	19
(4)	60→52	0.06	-	-	35.77	?	24
(5)	75→66	0.09	-	-	87.57	?	29
(6)	15→7	0.01	0.01	0.01	0.02	10	11
(7)	30→24	0.02	9.73	40.60	1.63	15	15
(8)	45→38	0.04	13094.15	8172.21	11.86	20	20
(9)	60→52	0.05	-	-	35.27	?	25
(11)	15→11	0.01	0.02	0.10	0.08	9	9
(12)	30→25	0.02	83.75	55.03	1.98	14	15
(13)	45→39	0.03	-	8598.74	11.96	19	21
(14)	60→53	0.07	-	-	50.18	?	25
(16)	15→11	0.01	0.03	0.05	0.12	4	7
(17)	30→25	0.02	2.56	18.15	2.16	8	15
(18)	45→39	0.04	1850.41	1877.5	12.01	12	17
(19)	60→53	0.06	-	-	39.67	?	25

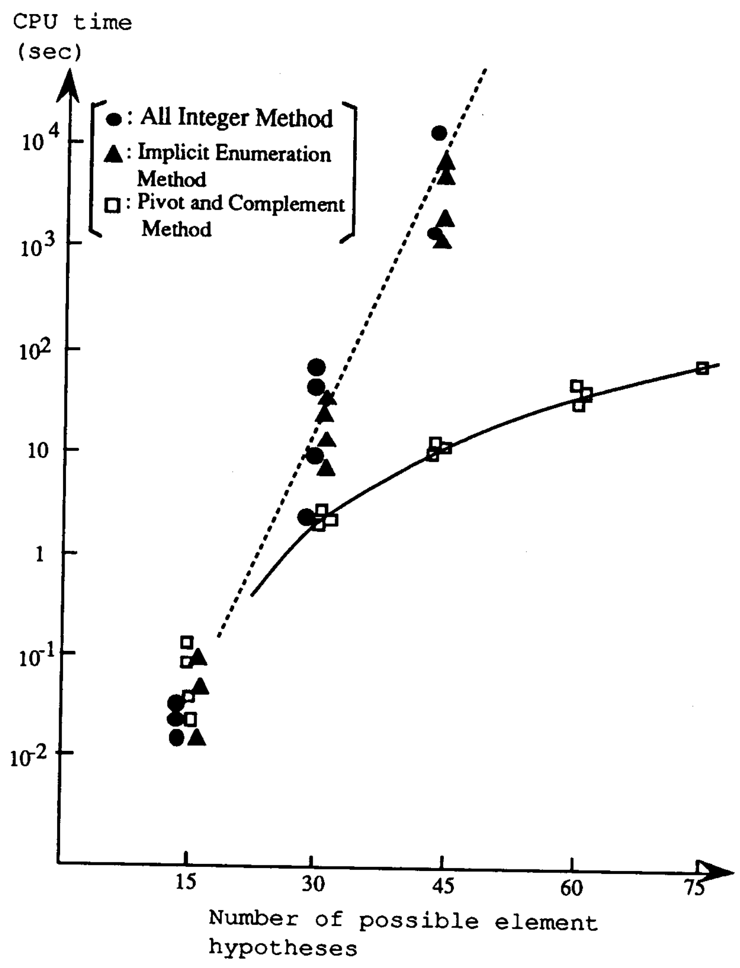

Fig. 6 Inference time (CPU time) of hypothetical reasoning systems employing 0-1 integer programming methods.

element hypotheses which indicates the scale of knowledge-base. Since the 0-1 integer programming is NP-complete problem, the all integer method and the implicit enumeration method which are exact solution methods show exponential-time performance against the scale of knowledge-base, though these methods were applied after the preprocessing including the extraction of relevant knowledge, etc.

On the other hand, Table 1 and Fig.6 show that the pivot and compliment (P&C) method which is an efficient approximate solution method can find a near-optimal solution in polynomial time. The regression analysis of the experimental data reveals that the CPU time is approximately 4.7th power of the number of possible element hypotheses in the knowledge-base. The obtained solutions coincide with the optimal solutions in many cases, or are good near-optimal solutions as seen in the original paper [Ballas 80] and Table 1. (It appears that the solution by the P&C method for example No.17 in Table 1 is not good, since its objective function value is 8 whereas the value of the optimal solution is 15. It becomes clear, however, by a detailed analysis that this approximate solution is the second best solution in this case.)

As seen in the above experiments, the pivot and complement (P&C) method allows a practical polynomial-time hypothetical reasoning. It is, however, not necessarily suitable from its algorithm for the following cases.

- The constraints in 0-1 integer programming is strong.
- The optimal real-number solution and the optimal integersolution are located far away to each other.

In other words, since this method emphasizes the near-optimality of the solution rather than reliably finding a feasible solution, there are cases that the method fails to find a 0-1 solution even if the solution exists. There may be possibilities of improving its performance by considering each problem structure particularly in the hypothetical reasoning problem.

6. Conclusions

This paper has presented a polynomial-time hypothetical reasoning which employs an approximate solution method of 0-1 integer programming. A preprocessing including the extraction of restricted knowledge relevant to a given goal and its simplification is incorporated to effectively reduce the number of variables of 0-1 integer programming. Unlike existing probabilistic search methods such as simulated annealing(SA) and genetic algorithm(GA), and other local search methods such as the heuristic repair method [Minton 92], GSAT [Selman 93a, 93b] and Gu's method based on the unconstrained nonlinear programming [Gu 93], the salient feature of our method is the efficient local search arround the optimal real-domain solution obtained by the efficient simplex method. A further improvement may be possible if we take account of a specific knowledge structure of the hypothetical reasoning.

References

[Balas 65] E. Balas: An Additive Algorithm for Solving Linear Programs with Zero-One Variables, Opsearch, Vol.13, pp.517-546 (1965)

[Balas 80] E. Balas and C. Martin: Pivot and Complement -- A Heuristic for 0-1 Programming, Management Science, Vol.26, pp.86-96 (1980).

[Bylander 89] T. Bylander, D. Allemang, et al.: Some Results Concerning the Complexity of Abduction, Proc. Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'89), (1989).

[Charniak 90] E. Charniak and S. Shimony: Probabilistic Semantics for Cost Based Abduction, Proc. AAAI'90 (1990).

[Charniak 92] E. Charniak and E. Santos Jr.: Dynamic MAP Calculation for Abduction, Proc. AAAI'92 (1992).

[Clark 78] K. L. Clark: Negation as Failure, in Logic and Databases (H. Gallaire and J. Minker (eds.)), Plenum Press, N.Y., pp.293-322 (1978).

[deKleer 86] J. deKleer: An Assumption-based TMS, Artifi. Intelli., Vol.28, pp.127-167 (1986).

[Dhar 90] V. Dhar and N. Ranganathan: Integer Programming vs. Expert Systems: An Experimental Comparison, Comm. ACM, Vol.33, No.3, pp.323-336 (1990).

[Dowling 84] W. F. Dowling and J. H. Gallier: Linear-time Algorithm for Testing the Satisfiability of Propositional Horn Formulae, Jour. of Logic Programming, Vol.3, pp.267-284 (1984).

[Garfinkel 72] R. Garfinkel and G. L. Nemhauser: Integer Programming, Jon Wiley and Sons (1972)

[Goldberg 89] D. E. Goldberg: Genetic Algorithm in Search, Optimization & Machine Learning, Addison-Wesley (1989).

[Greenberg 71] H. Greenberg: Integer Programming, Jon Wiley and Sons (1971)

[Gu 93] J. Gu: Local Search for Satisfiability (SAT) Problem, IEEE Tran. SMC, Vol.23, No.4, pp.1108-1129 (1993).

[Hooker 88] J. N. Hooker: A Quantitative Approach to Logic Inference, Decision Support Systems, Vol.4, No.1, pp.45-69 (1988).

[Ishizuka 90] M. Ishizuka and T. Matsuda: Knowledge Acquisition Mechanisms for a Logical Knowledge Base including Hypotheses, Knowledge-Based Systems, Vol.3, No.2, pp.77-86 (1990).

[Ishizuka 91] M. Ishizuka and F. Ito: Fast Hypothetical Reasoning System using Inference-Path Network, Proc. Int'l Conf. on Tools for AI (TAI'91), San Jose (1991).

[Ishizuka 93] M. Ishizuka and A. Abe: Fast Hypothetical Reasoning using Analogy on Inference-path Networks, Proc. Int'l Conf. on Tools with AI (TAI'93), Boston (1993).

[Ito 91] F. Ito and M. Ishizuka: Fast Hypothetical Reasoning System using Inference-Path Network (in Japanese), Jour. Japanese Soc. for AI, Vol.6, No.4, pp.501-509 (1991).

[Kirkpatrik 83] S. Kirkpatrik, et al.: Optimization by Simulated Annealing, Science, No.220, pp.671-681 (1983).

[Kondo 91] A. Kondo, T. Makino and M. Ishizuka: An Efficient Hypothetical Reasoning System for Predicate-logic Knowledge-base, Proc. Int'l Conf. on Tools for AI (TAI'91), San Jose (1991).

[Konno 81] H. Konno: Integer Programming (in Japanese), Sangyo-Tosho (1986)

[Makino 90] T. Makino and M. Ishizuka: A Hypothetical Reasoning System with Constraint Handling Mechanism and its Application to Circuit-Block Synthesis, Proc. PRICAI'90, Nagoya (1990).

[Minton 92] S. Minton, et. al.: Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems, Artif. Intelli., Vol.58, pp.161-205 (1992).

[Poole 87] D. Poole, R. Aleliunas and R. Goebel: Theorist: A Logical Reasoning System for Defaults and Diagnosis, in The Knowledge Frontier: Essays in The Knowledge Representation (N. J. Cercone and G. McCalla (eds.)), Springer-Verlag, N.Y. (1987).

[Poole 88] D. Poole: A Logical Framework for Default Reasoning, Artif. Intelli., Vol.36, pp.27-47 (1988).

[Selman 93a] B. Selman and H. Kautz: An Empirical Study of Greedy Search for Satisfiability Testing, Proc. AAAI-93 (1993).

[Selman 93b] B. Selman and H. Kautz: Domain-Independent Extensions to GSAT: Solving Large Stractured Satisfiability Problems, Proc. IJCAI-93 (1993).

[Tsuruta 91] S. Tsuruta and M. Ishizuka: A Compiling Method of Propositional Knowledge Base for Abductive Generation of Lacked Knowledge (in Japanese), Jour. Japanese Soc. for AI, Vol.6, No.1, pp.117-123 (1991).

[Tsuruta 92] S. Tsuruta and M. Ishizuka: A Compiling Method of Predicate Knowledge Base for Efficient Abductive Hypothesis Synthesis (in Japanese), Jour. Japanese Soc. for AI, Vol.7, No.1, pp.130-137 (1992).