

# New Approaches for Efficient Solution of Hitting Set Problem

Amir Fijany and Farrokh Vatan

Ultracomputing Technology Research Group

Jet Propulsion Laboratory, California Institute of Technology

4800 Oak Grove Drive

Pasadena, CA 91109

## Abstract

A new method for solving the hitting set problem is proposed. This method is based on the mapping of the problem onto an integer programming optimization problem. This new approach provides an algorithm with much better performance compare to the algorithms for the hitting set problem that currently are used for solving the diagnosis problem. The relation of this approach with the satisfiability problem and the complexity of the problem are also discussed.

## 1 Introduction

The *Hitting Set Problem*, also known as the *Transversal Problem*, is one of key problems in the combinatorics of finite sets (see [1]) and the theory of diagnosis (see [12, 2]). The problem is simply described as follows. A collection  $S = \{S_1, \dots, S_m\}$  of nonempty subsets of a set  $M$  is given. A hitting set (or transversal) of  $S$  is a subset  $H \subseteq M$  that meets every set in the collection  $S$ ; i.e.,  $S_j \cap H \neq \emptyset$ , for every  $j = 1, \dots, m$ . Of course, there are always trivial hitting sets, for example the background set  $M$  is always a hitting set. Actually we are interested in *minimal hitting sets with minimal cardinality*: a hitting set  $H$  is minimal if no proper subset of  $H$  is a hitting set.

Our primary interest to Hitting Set Problem is its connection with the problem of diagnosis (for more applications of this problem see, e.g., [3, 6]). Here we briefly review this connection. In *model-based diagnosis* approach [12, 2] one starts with a description (model) of the system which is a list of the components, their function, and the interconnections of the components. The diagnosis problem arises when some observations of the system's actual behavior is in contradiction with the system's expected behavior. More formally, a system is represented as  $(SD, COMPONENTS, OBS)$ , where  $COMPONENTS$  is a finite sets of components constituting the system (they are the constants of the logical theory describing the system), and  $SD$  and  $OBS$  are finite sets of first-order sentences describing the system description and observations, respectively. There is a distinguished unary predicate  $AB$ , where  $AB(c)$  is interpreted as "the component  $c$  is abnormal or faulty." A *diagnosis* of the system is a conjunction of the following form determined by a subset  $D \subseteq COMPONENTS$

$$\Delta = \Delta(D) = \left[ \bigwedge_{c \in D} AB(c) \right] \bigwedge \left[ \bigwedge_{c \in COMPONENTS \setminus D} \neg AB(c) \right],$$

such that the set  $SD \cup OBS \cup \{\Delta(D)\}$  is consistent. Thus the conjunction  $\Delta(D)$  is a diagnosis if the assumption of abnormality of the components in the set  $D$  is not in contradiction with the system description and the observations

in hand. The diagnosis  $\Delta(D)$  is *minimal* if for every proper subset  $D' \subset D$ , the conjunction  $\Delta(D')$  is not a diagnosis. Reiter [12] introduced the notion of “conflict” as a technical tool for finding the minimal diagnoses. A subset  $C \subseteq \text{COMPONENTS}$  is a *conflict* of the system if the set

$$SD \cup OBS \cup \{\neg AB(c) : c \in C\}$$

is inconsistent. Equivalently, the set  $C = \{c_1, c_2, \dots, c_k\}$  is a conflict if and only if the disjunction

$$\Gamma(C) = AB(c_1) \vee AB(c_2) \vee \dots \vee AB(c_k)$$

logically follows from the system; i.e.,  $SD, OBS \models \Gamma(C)$ . (To be more precise, we should call  $C$  (or  $\Gamma(C)$ ) a *positive conflict*; in the general definition,  $\Gamma(C)$  may contain negations.) The conflict  $C$  is *minimal* if no proper subset of  $C$  is a conflict. Let  $\mathcal{C}$  be the collection of all minimal conflicts of the system. Then the main theorem in the theory of model-based diagnosis [12, 2] states that the minimal diagnoses of the system are exactly the minimal hitting sets of  $\mathcal{C}$ .

The Reiter’s hitting set algorithm [12] is one the major algorithms for finding minimal hitting sets. The correction of this algorithm is provided by [8] and a modified more efficient version by [15]. The original algorithm and its modifications are based on generating the lattice of the subsets of the background set  $M$  and then extracting a sublattice of it that provides the minimal hitting sets. If the goal is to find a minimal hitting set with *minimal cardinality*, then this algorithm is not efficient by any means; because it requires to save the whole sublattice which leads (in the worst case) to the need of an exponential size memory to save the sublattice. In Section 4, we will show that it is possible to find a minimal hitting set with minimal cardinality with an algorithm that requires a linear size memory (while it still may needs an exponential time to complete the computation).

Our approach for solving the Hitting Set Problem is two-folded. In one hand we map the problem into the *Monotone* Boolean Satisfiability Problem. This provides the opportunity of utilizing the super-polynomial algorithms for finding the prime implicants of monotone functions (see [6, 9]). Also this mapping makes it possible to better understand the complexity of the Hitting Set Problem, by comparing it with the well-studied Boolean function problems. On the other hand, we map the problem onto an integer programming optimization problem. This simple mapping gives us access to vast repertoire of integer programming techniques that in some cases can effectively solve problems with several thousands variables. We would like to mention that mapping of the problem of finding prime implicants (not necessarily prime implicants of monotone formulas) onto the integer programming has already been introduced; see, e.g., [10, 11]. The mappings of the hitting set problem onto monotone satisfiability and integer programming, which is introduced in this paper, provides a new mapping of the problem of finding prime implicants of *monotone* formulas onto the integer programming.

## 2 Complexity of the Hitting Set Problem

Before we describe our algorithm, we would like to look at the complexity of the Hitting Set Problem. Although it is well-known that, in general, this problem is intractable, we show that even the restricted form of this problem, which we are interested in, is also **NP**-complete.

First note that for any system  $\mathcal{S}$  of subsets of the set  $M = \{m_1, m_2, \dots, m_n\}$ , finding *one* minimal hitting set is easy. We define the sequence  $M_0, M_1, \dots, M_{n+1}$  recursively as follows: let  $M_0 = M$ ; suppose that the set  $M_j$  is defined; let  $H = M_j \setminus \{m_j\}$ , i.e., remove the member  $m_j$  from  $M_j$ ; check whether  $H$  is a hitting set, if it is then let  $M_{j+1} = H$ , otherwise let  $M_{j+1} = M_j$ . Then it is easy to see that each set of the sequence  $M_0, M_1, \dots, M_{n+1}$  is a hitting set and the set  $M_{n+1}$  is in fact a minimal hitting set. The more challenging, and more interesting both from

practical and theoretical point of view, is the problem of finding hitting sets of small size. It turns out that this is a hard problem. First let formalize the problem.

### HITTING SET

INSTANCE: A system  $\mathcal{S} = \{S_1, \dots, S_m\}$  of subsets of the set  $M$  and a constant  $\frac{1}{2} < c < 1$ . QUESTION: Is there a hitting set  $H \subseteq M$  such that  $|H| \leq c|M|$ ?

We should mention that it is well-known that the above problem is **NP**-complete if the condition is replaced by  $|H| \leq K$ , where  $K \leq |M|$  (see [7]). It is also known that in this latter form the problem remains **NP**-complete even if  $|S_j| \leq 2$ , for every  $1 \leq j \leq m$ . Here we show that this restricted form of the problem remains **NP**-complete. In [3] the complexity of several other problems related to hitting sets is investigated.

**Theorem 1** *The problem HITTING SET is **NP**-complete. It remains **NP**-complete even if  $|S_j| \leq 3$ , for every  $1 \leq j \leq m$ .*

**Proof.** We use the transformation from the following **NP**-complete problem on monotone satisfiability (see [14]): An  $n$ -variable monotone CNF  $F$  where each clause contains at most 3 variables and a constant number  $c \in [\frac{1}{2}, 1)$  are given; is there a vector  $\mathbf{a} \in \{0, 1\}^n$  such that  $F(\mathbf{a}) = 1$  and  $\text{wt}(\mathbf{a}) \leq cn$ ? (Here  $\text{wt}(\mathbf{a})$  denotes the Hamming weight of the binary vector  $\mathbf{a}$ .)

Suppose that  $F = \bigwedge_{1 \leq j \leq m} C_j$  is a monotone CNF, where  $C_j = x_{j1} \vee x_{j2} \vee x_{j3}$ . Consider the system  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  where  $S_j = \{x_{j1}, x_{j2}, x_{j3}\}$ . Then there is a vector  $\mathbf{a} \in \{0, 1\}^n$  such that  $F(\mathbf{a}) = 1$  and  $\text{wt}(\mathbf{a}) \leq cn$  if and only if the system  $\mathcal{S}$  has a hitting set  $H$  with  $|H| \leq cn$ . ■

## 3 Mapping onto Boolean Satisfiability Problem

In order to describe mapping of the Hitting Set Problem onto Boolean Satisfiability and 0/1 Integer Programming problems, consider a different representation of the problem by describing the attribution of the members to subsets as given by the following matrix:

$$\begin{array}{c|cccc}
 & m_1 & m_2 & \cdots & m_n \\
 \hline
 S_1 & 1 & 0 & \cdots & 0 \\
 S_2 & 0 & 1 & \cdots & 1 \\
 \vdots & \vdots & \vdots & \ddots & \vdots \\
 S_m & 1 & 1 & \cdots & 0
 \end{array} \tag{1}$$

where  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  and  $M = \{m_1, m_2, \dots, m_n\}$  denote the set of nonempty subsets and the set of members (elements), respectively. The  $(i, j)$ <sup>th</sup> entry in this matrix is denoted as  $a_{ij}$  and we have  $a_{ij} = 1$  if  $m_j$  belongs to  $S_i$  otherwise  $a_{ij} = 0$ . To map the problem to Boolean Satisfiability, we introduce the Boolean variables  $x_1, x_2, \dots, x_n$ , where each variable  $x_j$  represents the member  $m_j$ . Then to each subset  $S_i = \{m_{j_1}, m_{j_2}, \dots, m_{j_{n_i}}\}$  (i.e., each row of the matrix (1)) we correspond the disjunction

$$F_i = x_{j_1} \vee x_{j_2} \vee \cdots \vee x_{j_{n_i}}; \tag{2}$$

i.e., for each "1" in the  $i$ <sup>th</sup> row of the matrix (1) the corresponding Boolean variable appears in the disjunction (2). For example, if the  $i$ <sup>th</sup> row of the matrix (1) is  $(0, 1, 1, 0, 0, 1, 0)$  then  $F_i = x_2 \vee x_3 \vee x_6$ . Then the CNF

$$F_{\mathcal{S}} = F_1 \wedge F_2 \wedge \cdots \wedge F_m \tag{3}$$

represents the mapping of the Hitting Set Problem associated with the system  $\mathcal{S}$  onto the Boolean Satisfiability Problem in the sense that every hitting set of the system  $\mathcal{S}$ , in a natural way, corresponds with a satisfying truth-assignment for the CNF  $F_{\mathcal{S}}$ , and vice versa.

We should notice that the CNF (3) is in fact *monotone*. In the case of monotone formulas, the standard form of the Satisfiability Problem should slightly be modified to avoid the trivial cases (see [14] for details). Note that, in the case of the monotone formulas, the all-one vector  $(1, 1, \dots, 1)$  is always a satisfying truth-assignment (equivalently, the background set  $M$  is always a hitting set). Here the correct formulation of the problem is to find the assignments with bounded weight, or in the hitting set setting, the problem is to find hitting sets with bounded number of members. In [14] it is shown that the problem of finding truth-assignments for monotone formulas with weight  $\leq cn$ , for  $\frac{1}{2} \leq c < 1$ , is NP-complete. Also, the problem of finding minimal hitting sets of the system  $\mathcal{S}$  reduces to the problem of finding prime implicants of the monotone function  $F_{\mathcal{S}}$ .

We should mention here a new results [6, 9] which suggests a major breakthrough regarding finding hitting sets in the most general case of the problem. They show that there is an algorithm that produces the list of prime implicants of a monotone Boolean function such that each prime implicant is produced in time  $O(nt + n^{O(\log n)})$ , where  $t$  is the time needed to determine the value of the Boolean function at any point. Also the list that produced by this algorithm has no repetitions. Practical implication of this result for hitting set problem is that for the systems that do not have large number of minimal hitting sets (i.e., there are at most superpolynomially many minimal hitting sets), it is possible to solve the hitting set problem in superpolynomial time, instead of exponential time of a typical NP-complete problem.

## 4 Mapping onto the 0/1 Integer Programming Problem

In order to describe the mapping onto 0/1 Integer Programming Problem, define the  $n \times m$  matrix  $A = (a_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$  associated with the system  $\mathcal{S}$ , as defined in (1). Note that, by this definition, each row of  $A$  corresponds to a subset and each column to a member. The mapping onto 0/1 Integer Programming Problem simply obtained by considering an operator application of  $A$  as follows. Identification of a minimal subset of members, representing a minimal hitting set, is equivalent to finding a minimal subset of columns of the matrix  $A$  whose summation results in a vector with elements equal to or greater than 1. This can be better described in terms of matrix-vector operation as follows. Let the vector  $A_i$ , for  $i = 1, \dots, m$ , denotes the  $i^{\text{th}}$  row of the matrix  $A$ . Also, define a binary vector  $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ , wherein  $x_j = 1$  if the member  $m_j$  belongs to the minimal hitting set, otherwise  $x_j = 0$ . Since at least one member should belong to every  $S_i$ , for every  $i = 1, \dots, m$ , we then have

$$A_i \cdot \mathbf{x} \geq 1.$$

Since, by the definition of the minimal subset, the above equation should be simultaneously satisfied for all  $i = 1, \dots, m$ , we then have the following formulation of the problem as an 0/1 integer programming problem

$$\begin{aligned} & \text{minimize} && \text{wt}(\mathbf{x}) \\ & \text{subject to} && A \mathbf{x}^T \geq \mathbf{b}^T \end{aligned} \tag{4}$$

where  $\mathbf{b} = (1, 1, \dots, 1) \in \mathbb{R}^m$  is the all-one vector, and we denote the *Hamming weight*, i.e., the number of one-components of the binary vector  $\mathbf{x}$ , by  $\text{wt}(\mathbf{x})$ . With this setting, identification of the minimal hitting set is then equivalent to solution for the binary vector  $\mathbf{x}$  from (4), which corresponds to the solution of the 0/1 Integer Programming Problem.

Note that (4) represents a rather special case of the 0/1 Integer Programming Problem since the matrix  $A$  is a binary matrix, i.e., with 1 or 0 elements only. Interestingly, our above derivation also establishes a mapping of the Monotone Boolean Satisfiability Problem onto this special case of 0/1 Integer Programming Problem. To see this, note that any Monotone Boolean Satisfiability Problem, given by the CNF (3), can be equivalently represented by a matrix similar to (1), from which the mapping onto this special case of 0/1 Integer Programming Problem follows immediately.

## 5 Monotonicity

One of the most important characteristics of the Hitting Set Problem is its *monotonicity*. This property reveals itself in different forms. For example, in the Boolean formula formulation of the problem, the formula  $F_S$  in (3) is a monotone Boolean formula (i.e., a formula expressed only with AND and OR operations, with no NOT operation). In the linear optimization formulation of the problem, as expressed in (4), if  $x$  is a solution of  $Ax^T \geq b^T$ , then any binary vector  $y \geq x$  is also a solution (here,  $y \geq x$  means that componentwise we have  $y_j \geq x_j$ ). We believe that incorporating this important property of the Hitting Set Problem with the existing algorithms will result in new algorithms with much more better performances. For example, instead of a generic integer programming method for solving the optimization system (4), a new algorithm utilizing the monotonicity property could be more efficient.

As we mentioned in Introduction, our primary concern with the Hitting Set Problem is its application in the diagnosis problem. Since in most practical applications the number of faulty components of the system is very small, we find that in such cases a structured brute force search algorithm, using the monotonicity property, could provide a solution very fast. Even in some cases faster than a standard integer programming algorithm. The structured brute force search algorithm, first looks in sets of size 1 for a solution, then in sets of size 2, and so on (see [4] for details). For example, if there are two faulty components, then such search could find them in less than a minute in a system with more than 1000 components.

## 6 Conclusion

We have proposed a new method for solving the celebrated hitting set problem. This method is based on the mapping of the problem onto an integer programming optimization problem. There are two important advantages of this approach comparing with the existing algorithms. (1) One important advantage of this method is that once the hitting set problem is mapped onto an integer programming optimization problem, then the inventory of various already existing integer programming algorithms and softwares are available for solving instances of the hitting set problem. (2) This approach makes a great improvement on the algorithms currently used for solving diagnosis problem. The Reiter's hitting set algorithm [12] and its revisions [8, 15] all need exponential size memory to be implemented. In contrast, our integer-programming based algorithm requires a linear size memory (while it still may needs an exponential time to complete the computation). We should also mention that this method provides a valuable theoretical tool for investigating the size of the solutions of the hitting set problem; the details of this finding will be describe in another paper [5].

## Acknowledgement

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA). This work is supported by the JPL InterPlanetary Network and Information Systems Directorate under the State Diagnosis task.

## References

- [1] C. Berge, *Hypergraphs: Combinatorics of Finite Sets*, Elsevier–North Holland, Amsterdam, The Netherlands, 1989.
- [2] J. de Kleer, A. K. Mackworth and R. Reiter, Characterizing diagnoses and systems, *Artificial Intelligence*, **56** (1992), pp. 197–222.
- [3] T. Eiter and G. Gottlob, Identifying the minimal transversal of a hypergraph and related problems, *SIAM J. Computing*, **24** (1995), pp. 1278–1304.
- [4] A. Fijany, F. Vatan, A. Barrett, M. James, C. Williams, and R. Mackey, A novel model-based diagnosis engine: theory and applications, to appear in *Proceedings of 2003 IEEE Aerospace Conference*, 2003.
- [5] A. Fijany and F. Vatan, New bounds on solution of certain NP-complete problems. Submitted to ....
- [6] M. Fredman and L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, *J. of Algorithms*, **21** (1996), pp. 618–628.
- [7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, W. H. Freeman and Company, New York, 1979.
- [8] R. Greiner, B. A. Smith, and R. W. Wilkerson, A correction to the algorithm in Reiter’s theory of diagnosis, *Artificial Intelligence*, **41** (1989), pp. 79–88.
- [9] V. Gurvich and L. Khachiyan, On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean Functions, *Discrete Appl. Math.*, **96–97** (1999) pp. 363–373.
- [10] V. M. Manquinho, P. F. Flores, J. P. M. Silva and A. L. Oliveira, Prime implicant computation using satisfiability algorithms, *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, IEEE Computer Society, Los Alamitos, CA, (1997) pp. 232–239.
- [11] L. Palopoli, F. Pirri and C. Pizzuti, Algorithms for selective enumeration of prime implicants, *Artificial Intelligence*, **111** (1999), pp. 41–72.
- [12] R. Reiter, A theory of diagnosis from first principles, *Artificial Intelligence*, **32** (1987), pp. 57–95.
- [13] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester, 1986.
- [14] F. Vatan, The complexity of diagnosis and monotone satisfiability, submitted to *Discrete Applied Mathematics*, 2001.
- [15] F. Wotawa, A variant of Reiter’s hitting–set algorithm, *Information Processing Letters*, **79** (2001), pp. 45–51.