Computing Minimal Hitting Sets with Genetic Algorithm

Lin Li[1, 2] and **Jiang Yunfei**[1]

Abstract. A set S that has a non-empty intersection with every set in a collection of sets *C* is called a hitting set of *C*. If no element can be removed from *S* without violating the hitting set property, *S* is considered to be minimal. Several interesting problems can be partly formulated as ones that a minimal hitting set or more ones have to be found. Many of these problems are required for proper solutions, but sometimes the approximate solutions are enough. A genetic algorithm and advantaged algorithms were devised for computing minimal hitting sets. An improvement makes them get most minimal hitting sets efficiently. Furthermore, they are smaller, i.e. fewer rules.

1 INTRODUCTION

A lot of theoretical and practical problems, e.g., [1~8], can be partly reduced to an instance of the minimal hitting set or one of its relatives, such as the minimum set cover problem, model-based diagnosis [1~5,7~8], and teachers and courses problem.

Normally speaking, it is a problem of selecting a minimal set (e.g., of teachers) that has a non-empty intersection with each set (e.g., list of courses), That is to say, there is, at least, one teacher who can teach any courses, This is a formulation of the minimal hitting set problem, which, in general, is NP-hard [6].

Generally, there are a number of hitting sets, but sometimes we only need one or some of them. There are some algorithms [1~8] for computing all of the minimal hitting sets, the space and time efficiency are not ideal. We present a novel method based on the Genetic Algorithm (in short GA here) for calculating minimal hitting sets.

Definition 1. (Hitting sets)

Given a collection $C=\{S_i \mid i \square \mathbb{N}\}$ of sets of elements from some universe *U*, a hitting set is a set $S \subseteq U$ such that $S \cap S_i \neq \varnothing$, for all *i*, i.e., a set which contains, at least, one element from all sets in *C*. Let HS(*C*) denote the collection of all hitting subsets in HS(*C*). These are called the minimal hitting sets of *C*.

We introduce a minimizing operator μ [5], MHS(*C*)= μ (HS(*C*)). We will use μ to get minimal conflict(/hitting) sets from conflict(/hitting) sets.

Determining a minimal cardinality element of MHS(*C*) is called the minimal hitting set problem.

Example 1. Model-based diagnosis [1], as shown in Figure 1. Suppose conflict sets are {M1, M2, A1}, {M1, A1, A2, M3}. The minimal hitting sets (diagnosis) are {M1}, {A1}, {M2, A2}, {M2, M3}.

{M1}, {A1} are of minimal cardinality.

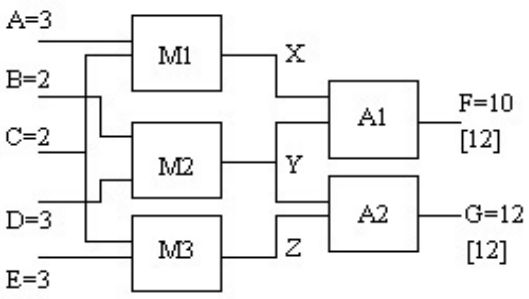

Figure 1 A simple circuit with 3 multipliers and 2 adders.

A minimal cardinality hitting set is a minimal hitting set of minimal cardinality.

In case of large sets of conflicts, the computation of the hitting sets will result both time and space consumption. Shown in Figure 2.

There are about millions of components, For example, in vehicles, computer systems, power plants, aircrafts, etc,. Therefore, we developed a novel efficient GA to compute minimal hitting sets. When the scale of conflicting sets is getting large, the GA method can still be used for computing the minimal hitting sets in a very short time.

[1] Sun Yat-sen university, Guangzhou, China.
[2] Jinan university, Guangzhou, China. email:linlionline@21cn.com.

2 GENETIC ALGORITHM

Genetic algorithm is a heuristic for the function optimization, where the extreme of the function (i.e., minimal or maximal) cannot be analytically established. A population of potential solution is refined iteratively by employing a strategy inspired by Darwinist evolution or natural selection. Genetic algorithms promote "survival of the fittest". This type of heuristic has been applied in many different fields, including construction of neural networks and multi-disorder diagnosis.

For the minimal hitting set problem, a straightforward choice of population is a set P of elements from 2^U, encoded as bit-vectors, where each bit indicates the presence of a particular element in the set.

Example 2. (Teacher and course problem) Let C denote a set cluster containing,

S_1={1, 2, 3, 4}, S_2={1, 2, 4}, S_3={1, 2}, S_4={2, 3}, S_5={4}.

It means that there are 5 courses {S_1, S_2, S_3, S_4, S_5 } and 4 teachers 1, 2, 3, 4. Teachers 1, 2, 3 and 4 can teach course S_1, teachers 1, 2, 4 can teach course S_2, … , teacher 4 can teach course S_5. We want to find the least teachers who can teach all of the 5 courses. This is a minimal hitting sets problem, and the minimal hitting sets are: H_1={1, 3, 4}, H_2={2, 4}.

We use bi-vectors to represent the sets and their hitting sets, these bi-vectors are called "chromosomes", each bit is called "gene", and all of the "chromosomes" are called "population".

If we use chromosome to represent the sets, they are represented as follow:

S_1={1, 1, 1, 1}, S_2={1, 1, 0, 1}, S_3={1, 1, 0, 0}, S_4={0, 1, 1, 0}, S_5={0, 0, 0, 1}.

The hitting sets are: H_1={1, 0, 1, 1}, H_2={0, 1, 0, 1}.

Here, $|S_i| \le |\square S_j|$, $|H_i| \le |\square S_j|$, so, the length of chromosomes equals to $|\square S_j|$.

Genetic operations include: "crossover", "mutation", "inversion", "selection" and "obtain".

Suppose that minimal conflict sets cluster is C={S_1, S_2, … , S_n}, $n=|\square S_k|$.

"Crossover" operator. Suppose that S_1={s_{11}, s_{12}, … , s_{1n}}, S_2={s_{21}, s_{22}, … , s_{2n}}, are two chromosomes, select that a random integer number $0<r<n$, S_3, S_4 is offspring of crossover(S_1, S_2),

S_3={s_i | if $i \le r$, $s_i \square S_1$, else $s_i \square S_2$},

S_4={s_i | if $i \le r$, $s_i \square S_2$, else $s_i \square S_1$}.

"Mutation" operator. Suppose that a chromosome S_1={s_{11}, s_{12}, … , s_{1n}}, selecting a random integer number $0<r \le n$, S_3 is mutation of S_1,

S_3={s_i | if $i \ne r$, then $s_i=s_{1i}$, else $s_i=1-s_{1i}$}.

"Inversion" operator. Suppose that chromosome S_1={s_{11}, s_{12}, … , s_{1r} , $s_{1, r+1}$, … , $s_{1, r+l}$, $s_{1,r+l+1}$, … , s_{1n} }, r, l are random numbers, S_2 is the inversion of S_1.

S_2={s_{11}, s_{12}, … , s_{1r} , $s_{1, r+l}$, … , $s_{1, r+1}$, $s_{1,r+l+1}$, … , s_{1n} }.

"Selection" operator. Suppose that there are m sets, we select [m/2] sets and eliminate other sets, the sets we selected are both "fitness" and "minimal", i.e. first, they intersect more sets than the other, and second, their cardinality is smaller.

"Obtain" operator. Suppose that there is a singleton set in the set cluster, then all hitting sets must hits this set, i.e. the gene stands for this set must be always kept as "1" , we refer to this operator as "obtain":

"Obtain" operator has no any influence on the result, it can improve the efficiency, such as a giraffe obtains "long neck". So they can be competed under the " long neck" condition.

Genetic algorithm.

1. InitializePopulation: Obtain $k*|C|*|\square S_i|$ population randomly, each chromosome is an n-length array, k is a const coefficient.

2. Testing if one of the stopping criteria (time, fitness, etc) holds. If it is yes, the procedure can be stopped, here,100 generations are gotten

3. Selection: Selecting one of chromosome; testing its fitness, here, being the number of sets it hits. Keeping the most fitness ones and deleting the bad ones.

4. Applying the genetic operator: such as "crossover", "mutation", "inversion" and "obtain" to the selected parents to form offspring.

5. Recombining the offspring and current population to form a new population with "selection" operator.

6. Repeating steps 2-5.

Also, we can use Genetic Algorithm to compute MINIMAL hitting sets from hitting sets.

In step 3. If we get hitting sets, we can undergo mutation operator just to change s_r from "1" to "0" in order to get its offspring, else, we undergo mutation operator just to change s_r from "0" to "1" in order to get its offspring. In the next selection operator we will go on keeping hitting sets because they are more fitting.

In the end, we will get 4 sets as follow:

1. Minimal hitting sets;

2. Both minimal hitting sets and their super-hitting sets; we will use operator μ to delete the super-hitting sets;

3. Hitting sets, but not minimal, their sub-hitting sets are not gotten;

4. No hitting sets, these sets will be deleted by "selection" operator.

But, in fact, the situation 3 is never gotten by GA test program.

We can get about 95 percent minimal hitting sets with GA. (shown in Figure 2)

3 COMPARISON

We have written a program to compare among HS-tree, BHS-tree [8] and GA; the result is shown in Figure 3 and Figure 4. The elements of every conflict sets are between 1 and 20.

In general, GA can get more than 95 percent minimal hitting sets in 100 generations, when the set cluster is big, then the HS-tree and BHS-tree can not run because of "Out of memory", but, GA can get almost all minimal hitting sets efficiently.

The space complexity of HS-tree is about $O(m^n)$, m is the average of $|S_i|$, n is $|C|$, That of BHS-tree is about $O(2^{|\cup S_i|})$,

that of GA is about $O(n|\sum S_i|)$. So the efficiency of GA is better than that of HS-tree and BHS-tree.

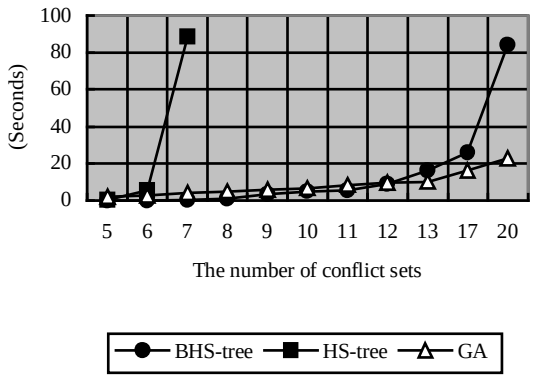

Figure 2 Running time among BHS-tree, HS-tree and GA. (CPU-PII 667, 128M main memory, C++, Windows'98)

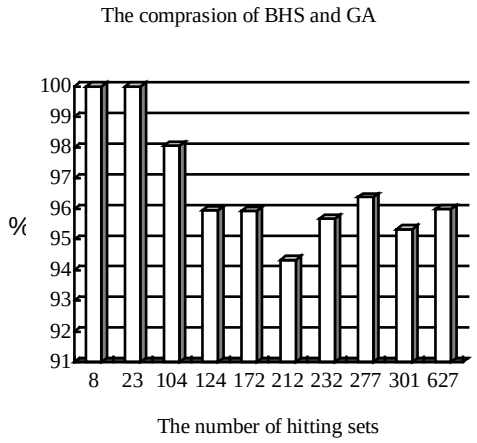

Figure 3 The hitting sets number and the percentage of GA gets.

4 CONCLUSIONS

In this paper, we have improved,

1. When the conflict sets scale gets big, This GA algorithm may get most of minimal hitting sets in a relative short time and small memory, but the other algorithm can't get the hitting sets because of "out of memory".

2. The GA algorithm can also get MINIMAL hitting sets. If a chromosome is not a hitting set, and the "mutation" operator just changes a random gene from "0" into "1", else change a random gene form "1" into "0" so that we can get minimal hitting set.

Example 3. (Continue to Example 2)

If we get H_3={1, 1, 0, 1} and know that it is a hitting set, then we undergo "mutation" operator to it, however, we only change "1" into "0" here.

H_3={1, 1, 0, 1}→{0, 1, 0, 1}, (minimal hitting set)
→{1, 0, 0, 1}, (no hitting set)
→{1, 1, 0, 0}. {no hitting set}

Underlined genes stand for "mutation" from parent genes.

3. Although this algorithm can't get all of the minimal hitting sets, but after we replace or repair these components we have computed, we can do next diagnosis step by step. The next research is GA used in choice of a repair/replace action on the set of suspects or choice of a next measurement.

This GA can be used in many other fields, e.g. a librarian can decide what kind of journals referred by researchers will be purchased under lack of funds. [6, pp124].

ACKNOWLEDGEMENTS

We are grateful to the referees for their comments, which helped us improve this paper.

REFERENCES

[1] R. Reiter, A theory of diagnosis from first principles. Artificial Intelligence, 1987, 32(1): 57~96.

[2] Greiner R, Smith B. A, Wilkerson R. W, A correction to the algorithm in Reiter's theory of diagnosis (research note). Artificial Intelligence, 1989, 41(1): 79~88.

[3] J. de Kleer J., A. K. Mackworth, R. Reiter, Characterizing diagnoses and systems. Artificial Intelligence, 1992, (56): 197~222.

[4] J. A. Reggia, D.S. Nau, P.Y. Wang, Diagnostic expert systems based on a set covering model, International Journal of Man-Machine Studies. 1983, (19): 437~460.

[5] Rolf Haenni, Generating diagnosis from conflict sets. 1997. http://www.aaai.org/

[6] Staal Vinterbo, Aleksander Ohrn, Minimal approximate hitting sets and rule templates, International Journal of Approximate Reasoning 2000, (25): 123~143

[7] Franz Wotawa, A variant of Reiter's hitting set algorithm. Information Processing Letters. 2001, (79): 45~51.

[8] Jiang Yunfei, Lin Li, Computing minimal hitting set from first principles with BHS-tree. Journal of software. Spring 2002. (Coming soon, In Chinese).

[9] A.E. Andreev, A.E.F. Clementi, J.D.P. Rolim, Hitting sets derandomize BPP, In: Automata, languages and programming, number 1099, Lecture Notes in Computer Science, Springer, Berlin, 1996, pp. 357~368.

[10] J.A. Reggia, D.S. Nau, P.Y. Wang, Diagnostic expert systems based on a set covering model, International Journal of Man-Machine Studies 19 (1983) 437~460.

[11] S. Vinterbo, L. Ohno-Machado, A genetic algorithm approach to multidisorder diagnosis, Artificial Intelligence in Medicine, 18 (2) (2000) 117~132.
[12] J.A. Swets, R.M. Pickett, Evaluation of Diagnostic Systems. Methods from Signal Detection Theory, Academic Press, New York, 1982.
[13] F.M. Brown, Boolean Reasoning, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
[14] J. Wroblewski, Finding minimal reducts using genetic algorithms, in: Proceedings of the Second Annual Join Conference on Information Sciences, October 1995, pp. 186~189.
[15] M. Mitchell, An Introduction to Genetic Algorithms, MIT press, Cambridge, 1996.
[16] D. whitley. An overview of evolutionary algorithm: practical issues and common pitfalls, Information and Software Technology. 2001(43): 817~831.
[17] Benjamin Han, Shie-Jue Lee. A genetic algorithm approach to measurement prescription in fault diagnosis. Information Science. 1999(120):223~237.