Information-Theoretical Mining of Determining Sets for Partially Defined Functions

Dan A. Simovici, Dan Pletea, Rosanne Vetro
Univ. of Massachusetts Boston,
Dept. of Comp. Science,
100 Morrissey Blvd.
Boston, Massachusetts
02125 USA
{dsim,dpletea,rvetro}@cs.umb.edu

Abstract

This paper describes an algorithm that determines the minimal sets of variables that determine the values of a discrete partial function. The algorithm is based on the notion of entropy of a partition and is able to achieve an optimal solution. A limiting factor is introduced to restrict the search, thereby providing the option to reduce running time. Experimental results are provided that demonstrate the efficiency of the algorithm for functions with up to 24 variables. The effect of the limiting factor on the optimality of the algorithm for different sizes of partial functions is also examined.

1 Introduction

Partially defined finite functions are studied by both mathematicians and engineers due to their many technical applications, particularly in designing switching circuitry. They model such diverse circuits as logical programmable arrays, or content addressable memory. The performance of such circuits (including wiring complexity, power dissipation, etc.) is heavily influenced by the number of arguments on which the function implemented by the circuit depends effectively.

The goal of this paper is to present an algorithm to generate various sets of input variables on which a partial function depends using the conditional entropy between sets of attributes.

This problem has been addressed in T. Sasao's seminal paper [Sas08] using an algebraic minimization algorithm that is applied to functions that depend on small number of variables. Our approach is distinct and involves techniques inspired by data mining. Additionally, it has the advantage of being independent of the values of the input or output radix of the partial function f.

In [SPV09] we developed an Apriori-like algorithm [AIS93, MT97, ZH05] that computes the entire collection of determining sets of a partially defined function by traversing the entire lattice of subsets of the set of variables, which is usually a rather large search space. The current approach applies techniques based on entropy of partitions and limits the search space by using a limiting factor. This improves considerably the running time. The algorithm presented in this paper is based on the partial order that is naturally defined on the set of partitions of a set and on the properties of conditional entropy of partitions of finite sets. Determining sets are computed by evaluating the conditional entropy of the output variable relative to subsets of the set of input variables of a partial function. Experimental results show the effectiveness of the algorithm.

The paper is organized as follows. In Section 2, the notion of determining set for a partial function is introduced and a few properties of these sets that play a role in our algorithm are examined. Section 3 introduces the conditional entropy of partitions that is used to find determining sets. The algorithm is presented in Section 4. Section 5 discusses experimental work related to the algorithm. Finally, Section 6 presents our conclusions.

2 Determining Sets for Partially Defined Functions

We denote the finite set $\{0, 1, \ldots, n-1\}$ by $\mathbf{n}$. The partial functions that we study have as domain a subset of the finite set $\mathbf{r}^n$ and as range a subset of the finite set $\mathbf{p}$ for some positive natural numbers r and p, referred to as the *input radix* and the *output radix* of the function, respectively. The set of all such partial functions is denoted by $\mathsf{PF}(\mathbf{r}^n, \mathbf{p})$. If $f \in \mathsf{PF}(\mathbf{r}^n, \mathbf{p})$ we denote by $\mathrm{Dom}(f)$ the set of all n-tuples $(a_1, \ldots, a_n)$ in $\mathbf{r}^n$ for which $f(a_1, \ldots, a_n)$ is defined.

Table 1. Tabular Representation of a Partial Function

x_1	x_2	x_3	y
0	1	1	0
0	1	2	1
0	2	1	2
0	2	2	2
1	0	1	3
1	0	2	3
2	0	1	3
2	0	2	3
1	1	0	2
1	2	0	2
2	1	0	1
2	2	0	0

A partial function $f \in \mathsf{PF}(\mathbf{r}^n, \mathbf{p})$ is specified as a table T_f having columns labelled by the argument variables $x_1, \ldots, x_n$ and by the output variable y. The set of variables that occur in a table T_f is denoted by Var_f.

If $f(a_1, \ldots, a_n) = b$ we have in the table T_f the $(n+1)$-tuple $t = (a_1, \ldots, a_n, b)$. For example, in Table 1 we show a partial function defined on all triplets in $\mathbf{3}^3$ that contain at least two non-zero elements, and ranging in the set $\mathbf{4}$: The number of rows of the table that represents a partial function defined on $\mathbf{r}^n$ can range between 0 and r^n. Usually, the number of rows of such a function is smaller than r^n and, often this number is much smaller. Tuples $(a_1, \ldots, a_n)$ that do not belong to the definition domain of f are considered as "don't care" tuples, that is, as input sequences that are unlikely to occur as inputs of the functions, or the output of the function for such inputs is indifferent to the designer.

For a tuple t in T_f and a set of variables $U \subseteq \mathrm{Var}_f$ we denote by $t[U]$ the *projection* of t on U, that is, the restriction of t to the set U. If $U = \{u_1, u_2, \ldots, u_m\}$ is a set of variables we will use the alternative notation $U = u_1 u_2 \ldots u_m$.

DEFINITION 2.1. *A set of variables $V = \{x_{i_0}, \ldots, x_{i_{p-1}}\} \subseteq \mathrm{Var}_f$ is a* determining set *for the partial function f if for every two tuples t and s from T_f, $t[V] = s[V]$ implies $t[y] = s[y]$.*

In other words, V is a determining set for the partial function f if $t = (a_0, \ldots, a_{n-1}, b)$ and $s = (c_0, \ldots, c_{n-1}, d)$ in T_f such that $a_{i_k} = c_{i_k}$ for $1 \leq k \leq p$ implies $b = d$. The collection of determining sets for f is denoted by $\mathsf{DS}(f)$.

V is a *minimal determining set for f* if V is a determining set for f and there is no strict subset of V that is a determining set for f. The set of minimal determining sets of f is denoted by $\mathsf{MDS}(f)$. Our main purpose is to present an algorithm that extracts the minimal determining sets for a partially specified function.

We introduce a partial order relation "$\sqsubseteq$" on the set $\mathsf{PF}(\mathbf{r}^n, \mathbf{p})$ by defining $f \sqsubseteq g$ if $\mathrm{Dom}(f) \subseteq \mathrm{Dom}(g)$ and $f(a_1, \ldots, a_n) = g(a_1, \ldots, a_n)$ for every $(a_1, \ldots, a_n)$. In other words, we have $f \sqsubseteq g$ if g is an extension of f.

The following simple statement is crucial to the proposed algorithm.

THEOREM 2.1. *Let f and g be two partial functions in $\mathsf{PF}(\mathbf{r}^n, \mathbf{p})$. If $V \in \mathsf{DS}(f)$ and $V \subseteq W$, then $W \in \mathsf{DS}(f)$. Furthermore, if $f \sqsubseteq g$, then $\mathsf{DS}(g) \subseteq \mathsf{DS}(f)$.*

Note that if $f \sqsubseteq g$ and $V \in \mathsf{MDS}(g)$, then there exists $Z \in \mathsf{MDS}(f)$ such that $Z \subseteq V$.

3 Entropies Associated with Partial Functions

Entropy is a probabilistic concept that lies at the foundation of information theory. The entropy of a partition takes advantage of the partial order that is naturally defined on the set of partitions of a set. In [SD08] a generalized notion of entropy for partitions, with Shannon entropy as a special case, is introduced.

A *partition* on a set S is a collection π of nonempty, pairwise disjoint sets

$$\pi = \{B_1, \ldots, B_m\}$$

such that $\cup_{i=1}^m B_i = S$. The sets B_i are referred to as the *blocks* of π. The set of partitions of S is denoted by $\mathsf{PART}(S)$. If $\pi, \rho \in \mathsf{PART}(S)$ we say that $\pi \leq \rho$ if every block of π is included in a block of ρ; equivalently, $\pi \leq \rho$ if every block of ρ is a union of blocks of π. For $\pi, \rho \in \mathsf{PART}(S)$ we say that ρ *covers* π if $\pi \leq \rho$ and there is no $\sigma \in \mathsf{PART}(S)$ distinct from π and ρ such that $\pi \leq \sigma \leq \rho$. This is denoted by $\pi \prec \rho$. It can be shown that $\pi \prec \rho$ if the blocks of ρ coincide with the blocks of π, with the exception of one block of ρ which is the union of two blocks of π.

The partially ordered set $(\mathsf{PART}(S), \leq)$ is actually a lattice. If $\pi, \rho \in \mathsf{PART}(S)$, $\pi = B_1, ..., B_m$ and $\rho = (C_1, \ldots, C_n)$, the greatest lower bound of π and ρ is the partition $\pi \wedge \rho$ given by

$$\pi \wedge \rho = \{B_i \cap C_j \mid B_i \cap C_j \neq \emptyset\}.$$

If C is a subset of S and $\pi = \{B_1, \ldots, B_m\} \in \mathsf{PART}(S)$, the *trace of π on C* is the partition $\{C \cap B_1, \ldots, C \cap B_m\} \in \mathsf{PART}(C)$. Unless stated otherwise, all logarithms are in base 2.

DEFINITION 3.1. *Let S be a finite set and let $\pi = B_1, ..., B_n$ be a partition of S. The Shannon entropy of*

π *is the number:*

$$\mathcal{H}(\pi) = \sum_{i=1}^{m} \frac{|B_i|}{|S|} \log \frac{|B_i|}{|S|}.$$

The Shannon entropy can be used to evaluate the uniformity of the distribution of elements of S in the blocks π since the entropy value increases with the uniformity of the distribution of the elements of S.

In [SJ02] we have shown that the entropy of a partition is a dually monotonic function. In other words, if $\pi \leq \rho$, we have $\mathcal{H}(\rho) \leq \mathcal{H}(\pi)$.

If $\pi, \rho \in \mathsf{PART}(S)$, $\pi = B_1, ..., B_n$ and $\rho = (C_1, \ldots, C_n)$, then the *conditional entropy of* π *on* ρ is the number

$$\mathcal{H}(\pi|\rho) = \sum_{j=1}^{n} \frac{|C_j|}{|S|} \mathcal{H}(\pi_{C_j}),$$

that is, the weighted average of the entropies of the traces of π on the blocks of ρ.

An equivalent expression of the conditional entropy can be obtained after elementary transformations as

$$\mathcal{H}(\pi|\sigma) = \mathcal{H}(\pi \wedge \rho) - \mathcal{H}(\sigma).$$

Our algorithm uses the conditional entropy of partitions of sets of variables defined as follows. If U, V are two sets of attributes, the entropy of U conditioned upon V is the difference

$$\mathcal{H}(U|V) = \mathcal{H}(UV) - \mathcal{H}(V). \tag{3.1}$$

The monotonicity of $\mathcal{H}$ implies that the function $\mathcal{H}(\cdot|\cdot) : \mathcal{P}(\mathrm{Var}_f) \longrightarrow \mathbb{R}$ is monotonic in its first argument U.

THEOREM 3.1. *Let S be a set and let $\pi, \sigma \in \mathsf{PART}(S)$. The conditional entropy $\mathcal{H}(\pi|\sigma)$ is monotonic relative to σ.*

DEFINITION 3.2. *Let f be a partial function, $f \in \mathsf{PF}(\boldsymbol{r}^n, \boldsymbol{p})$ and let V be a set of variables of f, $V \subseteq \{x_1, \ldots, x_n, y\}$. Define the partition π^V of $\mathrm{Dom}(f)$ by its corresponding equivalence $\sim_V$, where $u \sim_V w$ if $u[V] = w[W]$.*

The entropy of V $\mathcal{H}(V)$ *is the entropy $\mathcal{H}(\pi^V)$ of the partition π^V.*

Note that if V and V' are two sets of variables such that $V \subseteq V'$, then $\pi^{V'} \leq \pi^V$. Thus, $V \subseteq V'$ implies $\mathcal{H}(V) \geq \mathcal{H}(V')$, so the entropy is monotonic with respect to inclusion of attribute sets.

The role of the conditional entropy in detecting determining sets is highlighted by the next statement.

THEOREM 3.1. *Let f be a partial function, $f \in \mathsf{PF}(\boldsymbol{r}^n, \boldsymbol{p})$ and let U be a set of variables of f. Then, X is a determining set of f if and only if $\mathcal{H}(y|X) = 0$.*

Table 2. Partially Defined Function Example

x_1	x_2	x_3	x_4	y
0	1	0	1	1
1	2	1	2	3
0	3	1	0	2
0	0	0	1	1
2	2	1	3	3
1	3	1	2	3
0	1	1	0	2
2	0	0	1	2
0	1	0	2	1
0	1	0	2	1

4 An Algorithm for Mining MDSs using Entropy

The algorithm uses the conditional entropy $\mathcal{H}(y|X)$ described in the previous section to find determining sets. It starts evaluating $\mathcal{H}(y|X)$ for single element subsets $X = x$ and increases the size of the subsets with each successive iteration. In this way, all possible subsets of variables with size α are evaluated before any subset of S with size $\beta > \alpha$. The algorithm is not redundant because it does not evaluate $\mathcal{H}(y|X)$ for a subset of variables X more than once. For instance, if both subsets $X_1 = \{1\}$ and $X_2 = \{2\}$ are expanded in a subsequent iteration, then the subset $s_3 = \{1, 2\}$ common to both expansions will be evaluated once.

The proposed algorithm takes as input a partially defined function f and a limiting factor ℓ in the range $(0, 1]$ used to reduce the search space. The output is a collection of determining sets for f. The algorithm performs a search on the power-set of the set of variables $V = x_1 x_2 \cdots x_n$ of f. The limiting factor ℓ defines the subsets of variables that are expanded among all subsets within a given size. Namely, ℓ corresponds to a fraction of subsets X with equal size and lowest $\mathcal{H}(y|X)$ value. When the limiting factor equals 1, all the possible subsets are evaluated until one or more solution sets with equal size are found. The search stops when $\mathcal{H}(y|X) = 0$ for all the possible subsets X with a given size; these sets are referred to as *determining sets* for f. The minimum number found corresponds to the size of the first solution set since the search proceeds with increasing order of the subsets size and all the remaining subsets that have not been checked have a greater size.

Example. Table 2 shows another example of a partially defined function. The values of conditional entropies involved are given by:

$$\begin{aligned}
\mathcal{H}(y|x_1) &= \mathcal{H}(yx_1) - \mathcal{H}(x_1) = 7.229,\\
\mathcal{H}(y|x_2) &= \mathcal{H}(yx_2) - \mathcal{H}(x_2) = 12.381,\\
\mathcal{H}(y|x_3) &= \mathcal{H}(yx_3) - \mathcal{H}(x_3) = 6.703,\\
\mathcal{H}(y|x_4) &= \mathcal{H}(yx_4) - \mathcal{H}(x_4) = 7.229,\\
\mathcal{H}(y|x_3x_1) &= \mathcal{H}(yx_3x_1) - \mathcal{H}(x_3x_1) = 0,\\
\mathcal{H}(y|x_4x_1) &= \mathcal{H}(yx_4x_1) - \mathcal{H}(x_4x_1) = 0.
\end{aligned}$$

When the limiting factor takes a value equal to 0.25, the algorithm only indicates x_1x_3 as a determining set for the partially defined function presented in Table 2. Note that x_1x_4 is also a determining set for the given function and is part of the solution when the limiting factor takes a value that is at least 0.5.

Next, we discuss the algorithm Computing $\mathsf{MDS}(f, \ell)$ given in Figure 1, where f is a partial function and ℓ is the limiting factor.

We use a list of sets of variables L. The set of variables of f is denoted by S. The variable $SETLIST$ contains a list of sets of variables. A list of sets of variables with the lowest entropy values among the ones in $SETLIST$ is contained by $LOW\_ENTROPY\_SETLIST$.

The method ADD(L, X) inserts set X in L, while REMOVE(L, idx) deletes the set at index idx from L. The roles of CLEAR(L), NEXT(L), SIZE(X) and GET(L, idx) are clear. INDEX_MIN(E) obtains the index of the first element on the list E with lowest conditional entropy value. COLLECTIONADD($\mathcal{D}, X$) inserts set X in the collection of determining sets $\mathcal{D}$. The core of the algorithm is the method COMPUTE_ENTROPY(X) that evaluates the entropy value of a set of variables X.

The core of the algorithm is the function COMPUTE_ENTROPY(f, X) that has as input arguments a partial function f and a set of variables X. This function returns the Shannon Entropy corresponding to X. The function presented in Figure 2 defines the methods and variables used in function COMPUTE_ENTROPY(f, X).

We use two hash maps, MAP_X and MAP_YX, which store the keys corresponding to the values of the set of variables X and of the set of variables $X \cup \{y\}$ and the number of their occurrences, respectively. The function ENTROPY(M) evaluates the entropy of the sets of variables in map M.

The method GET_XKEYS(v, X) gets the values assigned in a registered vector v of a partial function f to the set of variables X; similarly, GET_YXKEYS(v, X) gets the values assigned in a registered vector v of a partial function f to the set of variables $X \cup \{y\}$. ADD(M, k, o) inserts into M the record with key k and number of occurrences o.

Figure 1. Computing $\mathsf{MDS}(f, \ell)$

Input: A partially defined function and a limiting factor

Result: A collection $\mathcal{D}$ of determining variables sets

```
begin
    $\mathcal{D} \longleftarrow \emptyset$
    $set\_size \longleftarrow 1$
    $dset\_found \longleftarrow false$
    foreach $var \in S$ do
        $X \longleftarrow var$
        ADD(SETLIST,$X$)
        $entropy \longleftarrow$ COMPUTE_ENTROPY($X$)
        ADD(ENTROPY_LIST,$entropy$)
        if $entropy = 0$ then
            COLLECTIONADD($\mathcal{D}$,$X$)
            $dset\_found \longleftarrow true$
    while $set\_size \leq$ SIZE($S$) and
    $dset\_found = false$ do
        $lf \longleftarrow \ell *$ BINOMIAL(SIZE($S$), $set\_size$)
        repeat
            $index \longleftarrow$
            INDEX_MIN(ENTROPY_LIST)
            $X \longleftarrow$ GET(SETLIST,$index$ )
            ADD(LOW_ENTROPY_SETLIST,$X$)
            REMOVE(SETLIST,index)
            REMOVE(ENTROPY_LIST,index)
            $lf \longleftarrow lf - 1$
        until $lf = 0$
        CLEAR(SETLIST)
        CLEAR(ENTROPY_LIST)
        $set\_size \longleftarrow set\_size + 1$
        while LOW_ENTROPY_SETLIST $\neq \emptyset$ do
            $X \longleftarrow$
            NEXT(LOW_ENTROPY_SETLIST)
            foreach $var \in S$ do
                if $X \cup var \ni$ SETLIST and
                $var \ni X$ then
                    ADD(SETLIST, $X \cup var$)
                    $entropy \longleftarrow$
                    COMPUTE_ENTROPY($X \cup$
                    $var$)
                    ADD(ENTROPY_LIST,$entropy$)
                    if $entropy = 0$ then
                        COLLECTIONADD($\mathcal{D}$,$X \cup$
                        $var$)
                        $dset\_found \longleftarrow true$
        CLEAR(LOW_ENTROPY_SETLIST)
end
```

Figure 2. COMPUTE_ENTROPY(f, X)

```
Input: A partially defined function, a subset $X$ of
       the complete set of variables of the partially
       defined function given
Output: An entropy value
1  begin
2      foreach $v \in F$ do
3          $keyX \longleftarrow$ GET_XKEYS($v, X$)
4          if $keyX \in MAP\_X$ then
5              $val \longleftarrow$ GET(MAP_X,$keyX$)
6              ASSIGN(MAP_X,$keyX$,val + 1)
7          else
8              ADD(MAP_X,$keyX, 1$)
9          $keyYX \longleftarrow$ GET_XYKEYS($v, X$)
10         if $keyYX \in MAP\_YX$ then
11             $val \longleftarrow$ GET(MAP_YX,$keyYX$)
12             ASSIGN(MAP_YX,$keyYX$,val + 1)
13         else
14             ADD(MAP_YX,$keyYX, 1$)
15     $entropyX \longleftarrow$ ENTROPY(MAP_X)
16     $entropyYX \longleftarrow$ ENTROPY(MAP_YX)
17     $entropy \longleftarrow entropyYX - entropyX$
18     return entropy
19 end
```

5 Experimental Results

We carried out experiments on a Windows Vista 64-bit machine with 8Gb RAM and 2 $\times$ Quad Core Xeon Proc E5420, running at 2.50 GHz with a 2$\times$6Mb L2 cache. The algorithm was written in Java 6.

We analyze the results in terms of running time and minimum number of variables of a determining set found as a function of the number of tuples in T_f and the limiting factor ℓ.

A program that randomly generates comma separated text files representing partially defined functions with a prescribed number of variables was developed. These values were chosen based on the experiments made in the related work of T. Sasao [Sas08] and in [SPV09].

One hundred files were randomly generated for each type of partially defined function (with 8 and 24 variables) using an input radix $r = 3$ and an output radix $p = 5$.

Note that a totally defined function with 8 variables and $r = 3$ has $3^8 = 6561$ tuples. In our experiments, we randomly generated 1000 tuples for each of the partially defined functions with 8 variables. For functions that depend on 24 arguments we generated 5000 tuples because the number of tuples for completely defined functions with 24 variables is much higher.

In the experiments, we evaluate the performance of

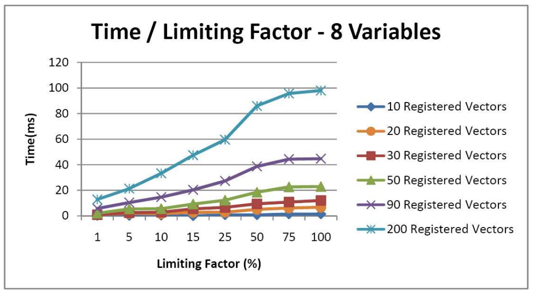

Figure 3. Dependency of average time on number of tuples and limiting factor for 8 variables.

the algorithm with a varying number of tuples and limiting factor. By Theorem 2.1, if $(f_1, f_2, \ldots, f_k)$ is a sequence of functions such that

$$f_1 \sqsubseteq f_2 \sqsubseteq \cdots \sqsubseteq f_k,$$

we have

$$\mathsf{DS}(f_k) \subseteq \cdots \subseteq \mathsf{DS}(f_2) \subseteq \mathsf{DS}(f_1).$$

In other words, when we start with a partial function f_1 with a small specification table T_{f_k} and we expend sequentially the specification of the functions, the number of determining sets will decrease. The experiments compare the results for files with 8 and 24 variables and they contain averages of the values corresponding to time and number of variables the function depends on as a function of the number of tuples and limiting factor. In our case, $k \in \{10, 20, 30, 50, 90, 100, 200\}$. The averages are evaluated over 100 functions within each group of generated functions (8 and 24 variables).

As shown in Figures 3 and 4, the running time increases with the number of tuples because in most cases, the larger the subset of variables X, the greater is the conditional entropy $\mathcal{H}(y|X)$. Likewise, the running time increases with the limiting factor ℓ since the search space increases as ℓ increases. Also, the time increases exponentially with the number of variables. It is clear that the number of subsets evaluated during the search depends on the original number of variables of a partial function f_1.

Finally, Figures 5 and 6 show that the number of variables the function depends on is related to the number of tuples k. As k increases, the constraints imposed on the problem become more extensive, and the number of variables that determines the value of the function increases.

We observed that the limiting factor ℓ has a remarkable insignificant impact on the size of determing sets. Therefore, the algorithm provides solutions that are optimal or

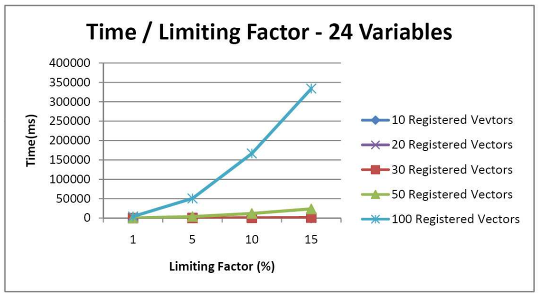

Figure 4. Dependency of average time on number of tuples and limiting factor for 24 variables.

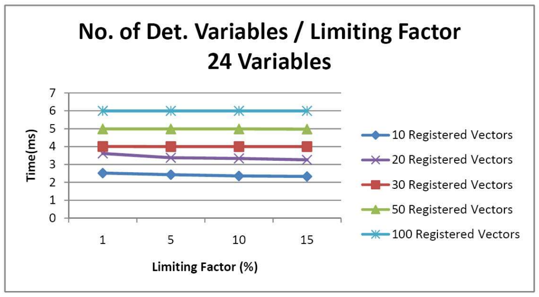

Figure 6. Average size of minimal determining set for 24 variables, as a function of the number of tuples and limiting factor.

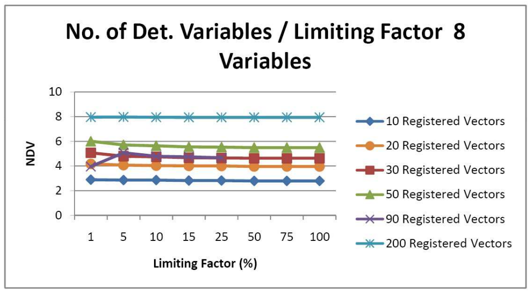

Figure 5. Average size of minimal determining set for 8 variables, as a function of the number of tuples and limiting factor.

near optimal, even when ℓ takes an extremely low value. This is an important result since as previously mentioned, the running time for the algorithm to find a solution is highly affected by the limiting factor chosen.

6 Conclusions

We developed an algorithm that identifies the sets of variables that determine a partially defined function and operates within a reasonable time by using partition conditional entropy. The search and evaluation proceed in increasing order of subset size and involve a fraction of the total search space defined by a limiting factor. When this factor is 1, all sets of variables of the partially defined function are evaluated, until the determining sets with smallest size are found. In this case the algorithm determines the minimum number of variables on which a partially defined function depends on, as well as all sets of variables with minimum number of elements that define the function. A limiting factor that takes values between 0 and 1 generates a search that is not complete but significantly faster depending on the factor chosen. Nevertheless, the effect caused by the limiting factor on the optimality of the algorithm is remarkably small. We believe that the algorithm will be helpful for digital circuit design since it allows to determine the possible sets of variables on which a partial function depends starting from a tabular specification of the function.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 International Conference on Management of Data*, pages 207–216, Washington, D.C., 1993. ACM, New York.

[MT97] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. Technical Report C-1997-8, University of Helsinki, 1997.

[Sas08] T. Sasao. On the number of variables to represent sparse logic functions. In *17th International Workshop on Logic and Synthesis (IWLS-2008)*, pages 233–239, Lake Tahoe, California, USA, 2008. IEEE-CS.

[SD08] D. A. Simovici and C. Djeraba. *Mathematical Tools for Data Mining – Set Theory, Partial Orders, Combinatorics*. Springer-Verlag, London, 2008.

[SJ02] D. A. Simovici and S. Jaroszewicz. An axiomatization of partition entropy. *IEEE Transactions on Information Theory*, 48:2138–2142, 2002.

[SPV09] D. Simovici, D. Pletea, and R. Vetro. Mining determining sets for partially defined functions. In P. Perner, editor, *Advances in Data Mining*, LNAI 5633, pages 353–360. Springer-Verlag, 2009.

[ZH05] M. J. Zaki and C.J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17:462–478, 2005.