Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp152-157)

Describing Data Table with Best Decision

ANTS TORIM, REIN KUUSIK
Department of Informatics
Tallinn University of Technology
Raja 15, 12618 Tallinn
ESTONIA
torim@staff.ttu.ee kuusik@cc.ttu.ee http://staff.ttu.ee/~torim

Abstract: - We propose a new way for describing data tables that is inspired by decision trees. Our goal is to summarize entire data table with one "average" object called best decision. The best decision is defined here as a decision that achieves the greatest value of a weight function. In our paper we first review computationally simple weight function for defining the best decision which does not account for the dependencies between the attributes. Then we define decision as a branch in a decision tree and introduce a weight function that takes those dependencies into account. As search-space for such decision grows factorially with the number of attributes, efficient pruning techniques are necessary. We define three pruning techniques that can be applied in combination. We present some empirical data to demonstrate the effectiveness of such techniques.

Key-Words: - data mining, decision tree, best decision, optimization, algorithms

1 Introduction

Data mining approaches like rule-sets [5], decision trees [4] and clustering [6] describe data tables with structures that are quite comprehensive put not as compact as traditional descriptive statistics – mean, median, mode. Traditional descriptive statistics however are calculated independently for every attribute and don't account for correlations between values of different attributes. "Average" object composed from values calculated independently for each attribute might not exist in data table at all. In the following article we will discuss the task of representing the data table with one "average" row that we call the best decision. The concept of the best decision is inspired by decision trees [4] and monotone systems theory [2, 7, 8, 9, 10]. It was first described in [1], however it was not presented to wider international audience. In contrast to the Quinlan's ID3 algorithm [4], we are not interested in the entire decision tree but only in the one branch of it - best decision. In our paper we first describe scale of conformity approach [11] for defining the best decision and review its shortcomings. Then we give our definition for best decision. As search-space for best decision grows factorially with the number of attributes, efficient pruning techniques are necessary. We present an algorithm that uses concept of potential to prune the search-space as described in [1]. We also describe two novel enhancements that allow additional pruning. Finally, we compare time efficiency of those enhancements by empirical measurements.

2 Problem Formulation

Best decision was originally intended as a formal way to select one from possibly conflicting recommendations of experts [1]. It is however applicable to any table containing discrete data as a way to describe it with one "average" row. We present computationally simple scale of conformity approach [11] and review its shortcomings. Then we present our approach.

2.1 Best Decision as Measured by the Scale of Conformity

The problem of finding the best decision can be defined in many ways. Here we are using following argument. We can calculate a weight for every decision. If we base our weight calculation on typicality then we can measure it by the conformity [11]. Conformity as a measure is calculated by the certain transformation where instead of attribute value we use its frequency – that is, we perform the so-called frequency transformation. For every row in the data table we can then calculate the sum of all attribute-value frequencies – that is we find the weight for row which we can call the conformity of that row (decision). The greater the conformity the more typical attribute values the row contains. According to that principle the row with the greatest conformity becomes the best decision.

Though we are evaluating every decision in relation to the system as a whole (data table is an object-attribute system), while summing frequencies, we may not always get the result that we can intuitively call the best decision. That is because we are not taking into account the dependencies between the attributes. That can lead to situations where the best decision does not feel

intuitively appropriate. Let's examine the following example. Let us have a data table *T*:

Table 1. Example.

	A1	A2	A3	A4	A5	A6	A7
O1	1	1	1	1	3	3	3
O2	1	1	1	2	3	3	3
O3	1	1	1	2	2	2	2
O4	2	2	2	1	2	2	2
O5	2	2	2	1	2	2	2

In our example the number of rows | *T* | equals to five. We denote the set of attributes in the table *T* by A_T and the set of possible values for an attribute *a* by *Dom(a)*. In our example A_T = *{A1, A2,..., A7}* and *Dom(A1) = {1, 2}*. We call pair *(a, v)* where *a* is an element of A_T and *v* is an element of *Dom(a)* element of decision. For each element *(a, v)* we can calculate its frequency $\pi_T((a, v))$ in data table *T*.

Table 2. Frequencies.

Attribute's value	A1	A2	A3	A4	A5	A6	A7
1	3	3	3	3	0	0	0
2	2	2	2	2	3	3	3
3	0	0	0	0	2	2	2

Each object (row) in the data table is a set of elements:

$$O = \{ (a_1, v_1), (a_2, v_2),..., (a_n, v_n)\}$$

Objects weight according to the scale of conformity is sum of its elements frequencies:

$$W(O) = \pi_T((a_1, v_1)) + \pi_T((a_2, v_2)) + ... + \pi_T((a_n, v_n)) \quad (1)$$

Table 3. Weights of objects in the scale of conformity.

	A1	A2	A3	A4	A5	A6	A7	W(O)
O1	3	3	3	3	2	2	2	18
O2	3	3	3	2	2	2	2	17
O3	3	3	3	2	3	3	3	***20***
O4	2	2	2	3	3	3	3	18
O5	2	2	2	3	3	3	3	18

Table 4. The best decision (by the scale of conformity).

(O3)	1	1	1	2	2	2	2

We can see that the object identified as the best decision is not very typical, as there is only one instance of it. In this case the scale of conformity approach does not seem trustworthy. That kind of result was caused by not taking into account the dependencies between the attributes. How should we behave when we assume mutual dependency between the attributes? Below we describe a suitable approach. We present its result (the best decision) right now in order to show the difference.

Table 5. The best decision (by another approach)

(O4, O5)	2	2	2	1	2	2	2

2.2 The Problem of Finding the Best Decision

Concept of best decision as described here was first formulated in [1]. Decision can be described as one branch in a decision tree. It's first element (*a, v* pair) defines sub-table, its second element defines sub-table of sub-table and so on. Informally, weight of a decision is sum of rows over those recursive sub-tables. Formal definition follows.

We denote the data table after the selection of an element *(a, v)* by *T \ (a, v)* and define it as a sub-table of *T* that contains all the rows of *T* where value of an attribute *a* equals *v* and all the columns of *T* except column *a*.

For example if *T* is Table 1 then *T \ (A2, 1)* is:

Table 6. *T \ (A2, 1).*

	A1	A3	A4	A5	A6	A7
O1	1	1	1	3	3	3
O2	1	1	2	3	3	3
O3	1	1	2	2	2	2

Definition 1. Ordered set $I_T = \langle (a_1, v_1), (a_2, v_2), ..., (a_n, v_n) \rangle$ that contains *n* elements from table *T* and where no attribute a_i occurs twice is decision. If $n = |A_T|$, then I_T is complete decision, if $n < |A_T|$ then I_T is partial decision.

One complete decision for Table 1 is *⟨(A7, 3), (A2, 1), (A4, 1), (A1, 1), (A3, 1), (A6, 3), (A5, 3)⟩.*

Definition 2. For a decision $I_T = \langle (a_1, v_1), (a_2, v_2),..., (a_n, v_n) \rangle$ we define its weight $W(I_T)$ as follows:

$$W(I_T) = \pi_T((a_1, v_1)) + W(I \setminus (a_1, v_1)_{T \setminus (a1, v1)}), \text{ if } |A_T| > 1$$
$$W(I_T) = \pi_T(a_1, v_1), \text{ if } |A_T| = 1 \quad (2)$$

For Table 1:

$$W(\langle (A7, 3), (A2, 1), (A4, 1), (A1, 1), (A3, 1), (A6, 3), (A5, 3) = 2 + 2 + 1 + 1 + 1 + 1 + 1 = 9$$

Definition 3. The best decision for table *T* is decision I_T, with greatest weight. That is, for any decision I'_T in

table T holds condition $W(I_T) \geq W(I'_T)$. We denote the best decision for table T by bI_T.

The best decision and its weight for Table 1 is,
$W(\langle (A7, 2), (A6, 2), (A5, 2), (A4, 1), (A3, 2), (A2, 2), (A1, 2)\rangle) = 3 + 3 + 3 + 2 + 2 + 2 + 2 = 17$.

Our definition for the best decision has several interesting properties:

- It takes into the account dependencies between the attributes.
- Row described by the best decision is guaranteed to exist in a data table.
- The order of elements in the best decision is significant and informative. First (attribute, value) pairs represent greater part of the weight than later elements and describe frequent and correlated (attribute, value) pairs. In the scale of conformity approach that information is missing.
- Best decision represents a branch in the decision tree [4]. It can be thought of as representing properties of a typical row in the order of importance. Unlike decision tree, it is not usable for exact classification. If we are dealing with several classes of objects, we can find best decision for each class by splitting table into sub-tables - one.sub-table for every class.
- In not requiring designated class, best decision is similar to association rules approach. It gives more compact and less comprehensive description for data table than association rules.

3 Problem Solution

3.1 Algorithms

We first review a simple brute-force algorithm [3] and after that an algorithm that uses the concept of potential for pruning the search tree. Then we add additional enhancements.

3.1.1 Brute-force algorithm

Brute-force algorithm for depth-first search can be described as follows. Recursive function *BestDecision* has three parameters: table (or sub-table) T, current best decision bI and partial decision under construction I. It returns the decision with the greatest weight from two options:

- The best decision that can be constructed by extending partial decision I with elements form the table T. That is always $I + bI_T$.
- Current best decision bI.

Call for the first level of recusion has the form *BestDecision* $(T, \langle\rangle, \langle\rangle)$ and we assume that $W(\langle\rangle) = 0$.

BestDecision(T, bI, I):
(End of recursion) If $|A_T| = 1$:
  Find (a, v) with greatest $\pi_T((a, v))$
  If $W(I + \langle (a, v)\rangle) > W(bI)$, then set $bI \leftarrow I + \langle (a, v)\rangle$
(Recursion) If $(|A_T| > 1)$:
  For each (a, v) in T:
    Set $bI \leftarrow$ BestDecison$(T \setminus (a, v))$, bI, $I + \langle (a, v)\rangle$
(Return) Return the decision bI

3.1.2 Pruning by potential

Brute-force algorithm needs to search through all possible decisions. Number of possible decisions is factorially dependent to the number of attributes $|A_T|$ as we need to check all possible permutations. As S. Skiena [3] claims good pruning techniques have stronger influence to the efficiency of a backtracking algorithm than any other factor. We describe how to use the concept of potential [1] for pruning the search tree.

Definition 4. The potential V for an element (a, v) and partial decision I from table T is:

$$V((a, v), T, I) = \pi_T((a, v)) \cdot |A_T| + W(I) \quad (3)$$

Potential sets upper limit to the weight of any decision $I + \langle (a, v), ...\rangle$, that we can construct from a partial decision I and an element (a, v).

Example potential from Table 1:
$V((A1, 1), T, \langle\rangle) = 3 \cdot 7 + 0 = 21$
Example potentials from Table 6:
$V((A1, 1), T \setminus (A2, 1), \langle (A2, 1)\rangle) = 3 \cdot 6 + 3 = 21$
$V((A4, 1), T \setminus (A2, 1), \langle (A2, 1)\rangle) = 1 \cdot 6 + 3 = 9$

It is trivial to see that it is impossible to construct a decision with the greater weight than that given by the potential when adding an element (a, v) to a partial decision. Potential equals weight in the ideal case when all the rows in the table $T \setminus (a, v)$ contain identical data.

If we compare potential of a partial decision to to the weight of the current best decison then we can prune number of branches from our search tree. To increase the effectiveness of pruning we try to find the high-weight decisions as early as possible. For that we are using the heuristic of examining elements in the order of their frequencies.
(*) - Enhancement for the brute-force algorithm

BestDecision(T, bI, I):
(End of recursion) If $|A_T| = 1$:
  Find (a, v) with greatest $\pi_T((a, v))$
  If $W(I + \langle (a, v)\rangle) > W(bI)$, then set $bI \leftarrow I + \langle (a, v)\rangle$
(Recursion) If $(|A_T| > 1)$:
  (*) Order all elements (a, v) decreasingly by $\pi_T((a, v))$
  (*) For each (a, v) where $V((a, v), T, I) > W(bI)$:

Set $bI \leftarrow$ BestDecison$(T \setminus (a, v)), bI, I + \langle\langle (a, v) \rangle\rangle)$
(Return) Return the decision *bI*

Example: Let us have a data table *T* with two attributes (possible values 1 and 2) and ten rows.

Table 7. Example.

A	**B**
1	1
1	1
1	1
1	2
1	2
1	2
1	2
2	1
2	1
2	1

We have identified $\langle\langle (A, 1), (B, 2) \rangle$ as the current best decision with the weight 11. We are trying to find BestDecison(*T*, $\langle\langle (A, 1), (B, 2) \rangle$, $\langle\rangle$). As potentials for elements *(A, 2)* and *(B, 2)* are less than 11, we can eliminate those branches from our search.

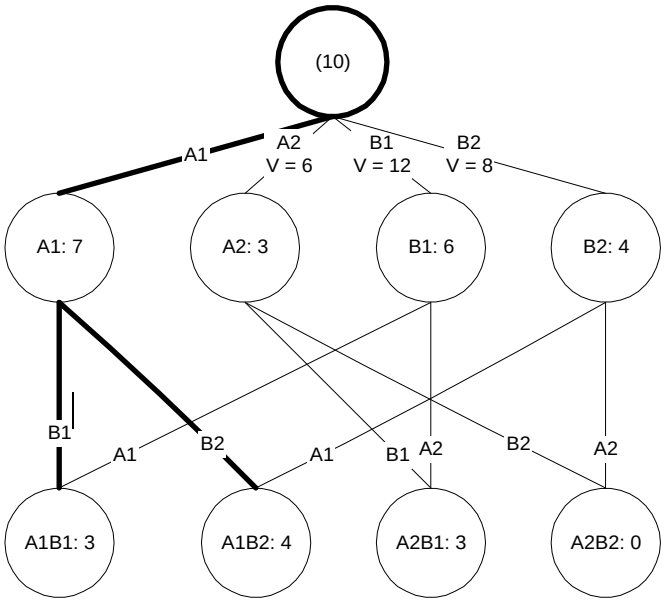

Fig. 1. Nodes are sub-tables with number of rows, lines are selections, solid lines represent examined search-space.

3.1.3 Additional enhancements

When examining our search procedure we can see that repeated visits to same sub-table are possible. We can get to the sub-table A1B1 by taking A1 as first element and B1 as second or by taking B1 as first element and A1 as second. Our new enhancements – bringing zeroes down and bringing zeroes up – reduce those repeated visits.

Bringing zeroes down allows us to detect by examining already searched elements (zeroes) from immediate upper level of backtracking, that we have already examined certain sub-table and partial decision that led to it had at least equal weight to our current partial decision.

Theorem 1
Let us have two elements *(a1, v1)* and *(a2, v2)* from table T so as $\pi_T((a1, v1)) \geq \pi_T((a2, v2))$. Then

$$W(\langle\langle (a1, v1), (a2, v2) \rangle\rangle) \geq W(\langle\langle (a2, v2), (a1, v1) \rangle\rangle \quad (4)$$

Proof: Tables $T \setminus (a1, v1) \setminus (a2, v2)$ and $T \setminus (a2, v2) \setminus (a1, v1)$ are identical (same sub-table). So frequencies $\pi_{T \setminus (a1, v1)}((a2, v2))$ and $\pi_{T \setminus (a2, v2)}((a1, v1))$ are also identical. As $\pi_T((a1, v1)) + \pi_{T \setminus (a1, v1)}((a2, v2)) \geq \pi_T((a2, v2)) + \pi_{T \setminus (a2, v2)}((a1, v1))$ it must be that $W(\langle\langle (a1, v1), (a2, v2) \rangle\rangle) \geq W(\langle\langle (a2, v2), (a1, v1) \rangle\rangle)$.
■

In other words, if $\pi_T((a1, v1)) \geq \pi_T((a2, v2))$, then partial decision $\langle\langle (a1, v1), (a2, v2) \rangle\rangle$ is at least equal in weight as partial decision $\langle\langle (a2, v2), (a1, v1) \rangle\rangle$. As we examine elements in order of their weights and use depth-first search, we can leave out from recursion level $T \setminus (a2, v2)$ every element *(a1, v1)* that was already examined in previous recursion level *T*.

Bringing zeroes up allows us to detect that sub-table corresponding to some element was already examined in immediate lower level of backtracking.

Theorem 2
Let us have two elements *(a, v)* and *(a´,v´)* from table T so as $\pi_T((a´, v´)) = \pi_{T \setminus (a, v)}((a´, v´))$. Then

$$W(bI_{T \setminus (a, v)}) \geq W(bI_{T \setminus (a´, v´)}) \quad (5)$$

Proof: If $\pi_T((a´, v´)) = \pi_{T \setminus (a, v)}((a´, v´))$ then table $T \setminus (a, v)$ must contain all the rows from table $T \setminus (a´, v´)$ – table $T \setminus (a´, v´)$ is its sub-table. Let decision $I´_{T \setminus (a, v)}$ contain same elements as decision $bI_{T \setminus (a´, v´)}$. As table $T \setminus (a, v)$ contains table $T \setminus (a´, v´)$ then $W(I´_{T \setminus (a, v)}) \geq W(bI_{T \setminus (a´, v´)})$. So table $T \setminus (a, v)$ must contain at least one decision with greater or equal weight as best decision from table $T \setminus (a´, v´)$.
■

So, if we find that frequency of element remains same in immediate lower level of backtracking then we need not examine this element again in upper level.

Enhanced algorithm:

K – Elements checked at upper level of recursion.
K' – Elements checked at current level of recursion.
(*) - Enhancement for purely potential based algorithm

BestDecision(T, bI, I, K):
(End of recursion) If $|A_T| = 1$:
  Find (a, v) with greatest $\pi_T((a, v))$
  If $W(I + \langle(a, v)\rangle) > W(bI)$, then set $bI \leftarrow I + \langle(a, v)\rangle$
(Recursion) If $(|A_T| > 1)$:
  (*) Set $K' \leftarrow \{\}$
  Order all elements (a, v) decreasingly by $\pi_T((a, v))$
  (*) For each (a, v) not in K and K' where
  $V((a, v), T, I) > W(bI)$:
    (*) Add (a, v) into K'
    Set $bI \leftarrow$ BestDecison$(T \setminus (a, v)), bI, I + \langle(a, v)\rangle, K')$
    (*) Add all elements (a', v') from $T \setminus (a, v)$ where
    $\pi_T((a', v')) = \pi_{T \setminus (a, v)}((a', v'))$ into K'
(Return) Return the decision bI

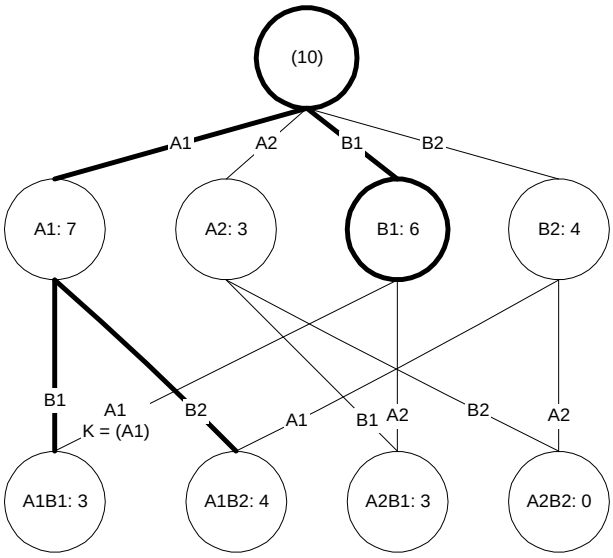

Fig 2. Example of bringing zeroes down (Table 7). As $A1$ is in the set of elements checked in level T, we can ignore it at the level $T \setminus B1$.

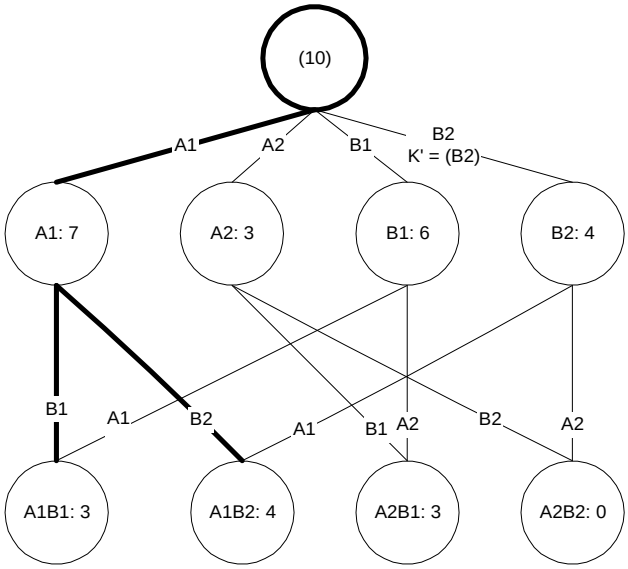

Fig. 3. Example of bringing zeroes up (Table 7). As $\pi_T((B2)) = \pi_{T \setminus A1}((B2)) = 4$ and we have examined whole level $T \setminus A1$ we can ignore $B2$ at level T.

3.2 Efficiency Comparison

We compared the efficiency of a brute-force algorithm against the efficiency of an algorithm using only potential and algorithm using potential, bringing zeroes up and bringing zeroes down for pruning. Properties of the testing enviroment were:

- processor Intel(R) Pentium(R) 4, 2.80 GHz
- 512 MB memory
- operating system Windows XP

We varied the number of columns and number of rows in our input tables and ran separate comparisons for random data tables and structured data tables. Attributes had values in range 1..9.

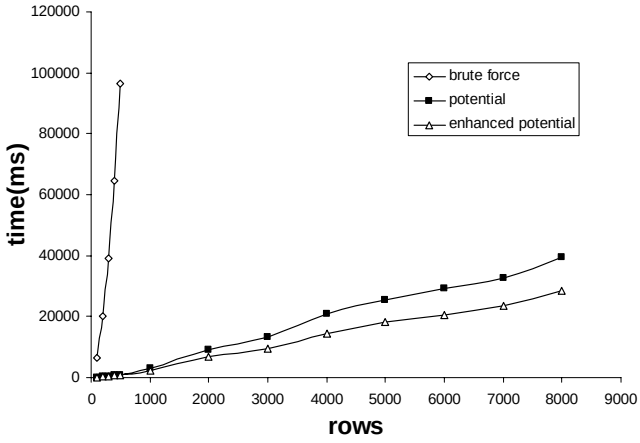

Fig 4. Random data, 5 columns.

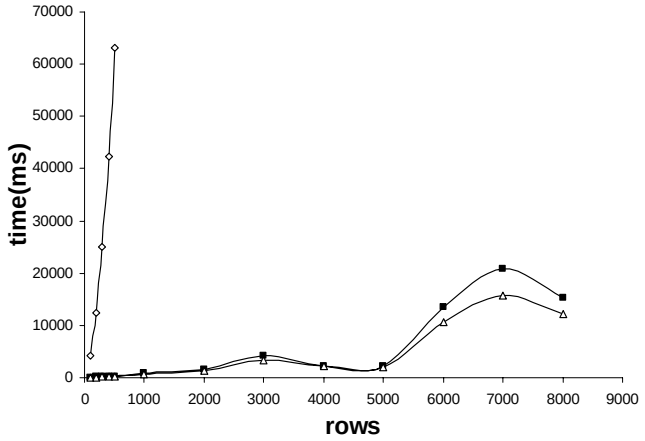

Fig 5. Structured data, 5 columns.
Uneven growth in time (potential, enhanced potential) is probably due to differences in tables internal structures.

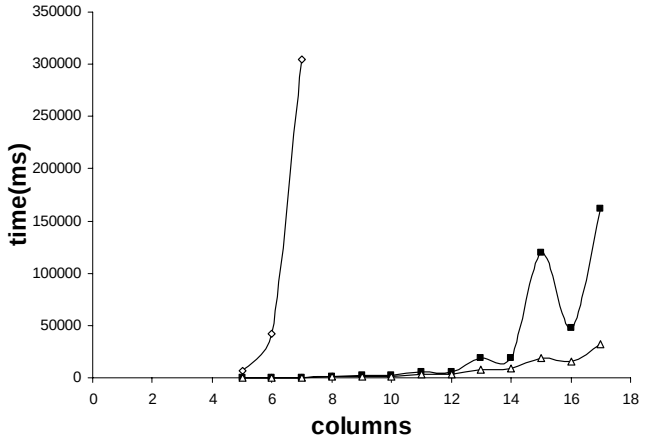

Fig. 6. Random data, 100 rows. Uneven growth in time (potential, enhanced potential) is probably due to differences in tables internal structures.

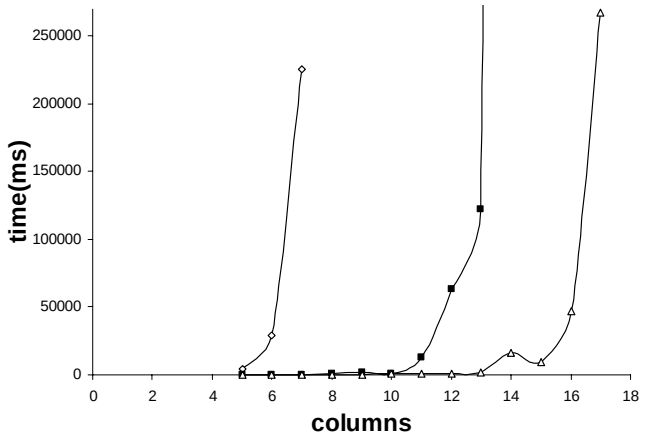

Fig. 7. Structured data, 100 rows.

As we can see, the algorithm that applies potential for pruning is significantly more efficient than the brute-force approach. Proposed additional enhancements speed it up even further.

4 Conclusion

Best decision describes data table in a compact way. Calculating it is computationally complex task. Though proposed pruning techniques offer significant speed-up compared to brute-force algorithm, for significant number of columns they are still inadequate. We have several ideas for further optimization for example through sub-table caching techniques.

References:

[1] Kuusik, R. Application of Theory of Monotonic Systems for Decison Trees Generation. *Transactions of Tallinn Technical University no. 705*

[2] Mullat, I. Extremal Monotonic Systems. *Automation and Remote Control*, No 5, 1976

[3] Skiena, S. *The Algorithm Design Manual*. Springer-Verlag, 1998

[4] Quinlan, J. R. Induction of Decision Trees, *Machine Learning*, (1), 1986, pp. 81-106

[5] Cohen, W. W. Fast effective rule induction. *Machine Learning: Proceedings of the Twelfth International Conference,* 1995. pp. 115—123

[6] Kaufman, L. and Rousseeuw, P. *Finding groups in data: An Intorduction to cluster analysis*. NewYork: Wiley, 1990.

[7] Kuusik, R. Lind, G., Võhandu, L. Pattern Mining as a Clique Extracting Task. Posters. *Tenth International Conference IPMU 2004 Information Processing and Management of Uncertainty on Knowledge-Based Systems*. July, 4-9, 2004, Perugia, Italy, ISBN 88-87242-54-2, pp. 19-20.

[8] Lind, G. Method for Data Mining - Generator of Hypotheses. Databases and Information Systems. *Proceedings of the 4th International Baltic Workshop*. Vilnius 2000. Vol. 2, pp. 304-305.

[9] Kuusik, R., Lind, G. An Approach of Data Mining Using Monotone Systems. *Proceedings of the Fifth International Conference on Enterprise Information Systems*. Angers, France 2003. Vol. 2, pp. 482-485.

[10] Kuusik, R., Lind, G. New frequency pattern algorithm for data mining. *Proceedings of the 13th Turkish Symposium on Artificial Intelligence and Neural Networks*. 10-11 June, 2004, Foca, Izmir, Turkey, ISBN 975-441-213-8, pp. 47-54.

[11] Võhandu, L. Express Methods of Data Analysis, *Transactions of Tallinn Technical University*, No. 464, 1979, pp. 21-37 (in Russian).