Robust and Cost-Effective Approach for Discovering Action Rules

Nasrin Kalanat, Pirooz Shamsinejad, and Mohamad Saraee

***Abstract*—The main goal of Knowledge Discovery in Databases is to find interesting and usable patterns, meaningful in their domain. Actionable Knowledge Discovery came to existence as a direct respond to the need of finding more usable patterns called actionable patterns. Traditional data mining and algorithms are often confined to deliver frequent patterns and come short for suggesting how to make these patterns actionable. In this scenario the users are expected to act. However, the users are not advised about what to do with delivered patterns in order to make them usable. In this paper, we present an automated approach to focus on not only creating rules but also making the discovered rules actionable.**

Up to now few works have been reported in this field which lacking incomprehensibility to the user, overlooking the cost and not providing rule generality. Here we attempt to present a method to resolving these issues. In this paper CEARDM method is proposed to discover cost-effective action rules from data. These rules offer some cost-effective changes to transferring low profitable instances to higher profitable ones. We also propose an idea for improving in CEARDM method.

***Index Terms*—actionable knowledge discovery, cost-effective action rules, profit mining.**

I. Introduction

Data mining has focused on studying how to build statistical models such as classification rules, association rules . . . etc of large database. These models often satisfy expected technical interestingness and provide passive knowledge. Using these models in real world business, users can only discover object (customer) models or profiles. But real world businesses often are interested to be delivered active knowledge such as marketing strategies. For example a company may want to produce marketing strategies for stopping their valuable customers from leaving. Since data mining algorithms often limited to deliver frequent patterns usually don't take any step for suggesting active knowledge and users will be responsible for it. So users will be faced with many patterns that they are confused about how and what to do with them. It is because of the gap between the academia and business aims. A general requirement in bridging the gap between academia and business is to consider domain and business factors and constraints in data mining process. It has been done by domain driven data mining [1,2,3]. Actionable Knowledge Discovery is a paradigm moving toward Domain-Driven Data Mining that is aimed at discovering active knowledge called actionable patterns.

Let illustrate this concept by an example in CRM. Consider a bank loan system. We can define two types of useful knowledge in this system. First, “How much is the probability of a customer pay back his loan?” and second, “How we can increase the probability of a customer pay back his loan?” The first question is passive knowledge that is more informative and less actionable and it is the concern of traditional data mining, but the second one is active knowledge that is more actionable and AKD aiming for answering it.

We can divide the works that have been done in AKD from the types of mined patterns point of view into two categories: those who try to define some actionability measures for filtering mined patterns[5,6] and those who try to extract new actionable patterns from mined patterns[7,8]. In other words methods in first category don’t create new patterns but those in second one extract new type of pattern. One type of these created patterns called “action rules”. This presented work is an action rule creating work.

Action rule generally means a rule that suggest an action to user to gain a profit in his/her domain. For example in our bank loan system an action rule could be like this : “If we can change marital status of male customers from single to married in some way then the probability of they pay back their loan will be more ”. It is worth noting that action rules are not talking about causality but about probability.

Up to now a few works have been done on mining action rules. For example in [7,8] a method for extracting cost-effective action rules for each customer from decision tree is proposed. In [9] some pruning strategies devised for filtering most actionable patterns. In that work the actions supposed to be present. Constructing action rules from certain pairs of previously mined classification rules is presented in [10,11,12,13,14,15,16] but in these works the cost of actions is neglected.

There are two important factors in constructing an action rule. First is the generality of action rule that means to how many instances it can apply. Second is the cost of action rule in its domain. In current paper a new method is proposed for mining cost-effective action rules. Our contribution is to combine the generalization power of E-action rules [12] and cost-effectiveness. For doing this a new algorithm, namely CEARDM is devised for extracting cost effective E-action rules from an information system. We also purpose an idea for improving this algorithm. The rest of paper is as follows: In Section II action rules are defined and the method of constructing them is described. The CEARDM is explained in Section III. Correctness of purpose algorithm is proved in section IV. An idea for improving CEARDM algorithm is presented in Section V. Finally we conclude the paper in

Manuscript received September 8, 2011, revised September 13, 2011.

Nasrin Kalanat and Pirooz Shamsinejadbabaki are with Electrical and Computer Engineering Department of Isfahan University of Technology, Isfahan, Iran (e-mail: n.kalanat@ec.iut.ac.ir; p_shamsinejad@ec.iut.ac.ir).

Mohammad Saraee is the founder and Director of the Intelligent Databases, Data Mining and Bioinformatics Research Centre.

Section VI.

II. Action Rules

Action rules would be constructed from certain pairs of previously mined classification rules [13,14]. These rules suggest changes in the value of some attributes of an instance to make it probably more profitable. For example let assume after mining purchasing data of a company two classification rules have been found as follows:

$$r1 : (sex, male) \wedge (service, H) \rightarrow (loyality, high)$$

$$r1 : (sex, male) \wedge (service, L) \rightarrow (loyality, low)$$

By combining these two rules that are only informative we can make this action rule that suggests some changes to improve the loyalty of a group of customers:

$$r1 - r2\ action\ rule\ : (sex, male) \wedge (service, L \rightarrow H) \rightarrow$$

$$(loyality, low \rightarrow high)$$

This rule suggests if we change the service status of male customers from low to high then it will be probable that their loyalty goes from low to high. But the problem is how to construct these rules from data and how to find the most valuable of them. Both of these will be discussed in this paper.

We assume data are placed in a table named decision table. Columns of table are attributes and rows are instances. Attributes are of two types: condition attributes and decision attributes. Decision attributes are attributes that profit in domain is related to them directly. In above example “loyalty” is a decision attribute and “sex” and “service” are condition attributes. Table 1 is a decision table for a fictitious problem. In this table {A,B,C,D,E,F} are condition attributes, Z is a decision attribute and {X1,X2,…,X13} are instances. For simplicity it’s assumed that values of condition attributes are numbers or are mapped to numbers, and there is only one decision attribute shown by *Z*.

Domain Driven Data Mining [21, 22] suggests that for making the results of DM process more applicable in real domains more characteristics of domain must be integrated to the process. According to this rule we took a more realistic look at condition attributes and divided them to three types: 1. Stable attributes whose values can’t change at sensible cost like “sex”. 2. Flexible attributes which their values can change at reasonable cost i.e. “service level”. 3. Asymmetric attributes which changing some of their values is sensible and others is not, i.e. “experience level” that it can be changed from low to high by spending some money and time but it can’t be changed from high to low at a sensible cost.

For integrating characteristics of all types of attributes in mining cost-effective action rules it is defined a cost matrix CM_A for each attribute A. Rows and columns of CM_A are both values of attribute A and $CM_A[i][j]$ shows the cost of changing i^th value to j^th value; changes with unreasonable cost show by infinity value in their corresponding cells. Cost matrixes for attributes of above example are depicted in Fig. 1. In addition to cost, we must consider the profit that gained from an action. In this work we assume the decision attribute has two values low and high and the profit of changing decision attribute value from low to high for an instance that is measurable and is shown by $P(L \rightarrow H)$. Changes whose cost is more than $P(L \rightarrow H)$ are worthless. The minimum cost of a change in value of an attribute is called flexibility factor of that attribute and it can be simply computed from cost matrix of each attribute.

TABLE I: A SAMPLE DECISION TABLE

	A	*B*	*C*	*D*	*E*	*F*	*Z*
X1	1	1	2	1	1	2	L
X2	1	1	2	1	2	2	L
X3	2	2	2	2	1	1	L
X4	2	2	2	1	1	2	L
X5	1	1	2	2	1	2	L
X6	1	1	1	2	1	2	L
X7	1	2	2	2	2	1	H
X8	2	3	2	2	2	1	H
X9	1	1	1	2	2	1	H
X10	2	1	1	1	1	1	H
X11	1	1	2	2	1	2	L
X12	1	1	1	1	1	2	L
X13	1	1	2	2	1	1	L

C	*1*	*2*
1	0	15
2	8	0

B	*1*	*2*	*3*
1	0	30	50
2	100	0	40
3	20	50	0

A	*1*	*2*
1	0	100
2	100	0

F	*1*	*2*
1	0	8
2	3	0

E	*1*	*2*
1	0	1
2	1	0

D	*1*	*2*
1	0	100
2	18	0

Figure. 1. Cost Matrixes for attributes in decision table shown in table I

Based on the above discussion the attributes are divided into two main types: Attributes that their flexibility factor is more than $P(L \rightarrow H)$, namely invaluable attributes and those with flexibility factor less than $P(L \rightarrow H)$, namely valuable attributes.

Valuable attributes contain at least one possible change at a reasonable cost regarding the most profit that may be gained. If *V* stands for set of valuable attributes of decision table *T* and *IV* for its invaluable attributes then table schema can be shown by $T(V \cup IV \cup \{Z\})$. The new concept of valuable and invaluable attributes makes it possible to define flexibility or stability of attributes in a dynamic way which means their flexibility in a particular domain will depend on the profit that may be obtained by their changes in that domain.

Classification rules are the raw material of action rules, so that for discovery of action rules the first step is mining classification rules from data. There are many algorithms for finding classification rules like C4.5, ID3, CART [23]. Table 2 shows some classification rules mined from decision table shown in Table 1. In this table each row represents a classification rule and first column of each row shows the instances satisfied that rule. For example the first row of the table represents the following rule *R* and also informs instances X_1 and X_2 satisfy this rule.

$$R : (A = 1) \wedge (B = 1) \wedge (C = 2) \wedge (F = 2) \rightarrow (Z = low)$$

For describing the process of constructing action rules some notations must be defined. Let L_R to be the set of the left attributes of rule *R*, $Val_{R,A}$ to be the value of attribute *A* in rule *R* and D_R to be the value of decision attribute of *R*. So that for above rule, L_R would be the set *{A,B,C,F}*, $Val_{R,A}$ equals 1 and D_R would be “*low*”. Also the following notations are presumed:

- (A, v) means attribute *A* must have the value of *v*.
- $(A, \rightarrow v)$ means the value of attribute *A* must be changed to the value of *v*.
- $(A, v \rightarrow w)$ means attributes *A* must be changed from the value of *v* to the value of *w*.

TABLE II: MINED CLASSIFICATION RULES WITH THEIR SUPPORTING OBJECTS

	A	B	C	D	E	F	Z
x1,x2	1	1	2	1		2	L
x3,x4	2	2	2		1		L
x1,x5,x6,x11,x12,x13	1	1			1		L
x7,x8			2	2	2	1	H
x9,x10		1	1			1	H

There are two following preconditions for each pair R_1 and R_2 of classification rules to be able to construct an action rule:

- $Val_{R_1, attr} = Val_{R_2, attr} \quad attr \in \left\{ IV \cap L_{R_1} \cap L_{R_2} \right\}$
- $Val_{R_1, Z} = low \quad \wedge \quad Val_{R_2, Z} = high$

where L_R is the set of attributes in left side of rule *R*. If the preconditions hold then R_1 and R_2 can construct an action rule by the algorithm described in Algorithm 1.

But the problems are how to find the consistence pair of classification rules and how to extract the most cost-effective action rules. In the next section the CEAT algorithm is presented as a solution to these problems.

III. EXTRACTING COST-EFFECTIVE ACTION RULES

A. Net Profit of Action Rule

For computing the net profit of an action rule it is necessary to compute the number of instances that support it in the population of instances. Let assume *R* be an action rule that has been constructed from R_1 and R_2, (b_1, b_2,.., b_p) be set of all valuable attributes of *R* which have different values in R_1 and R_2. If v_i and w_i stands for values of attribute b_i in R_1 and R_2 respectively, then instance *X* is said to support *R* if there is another instance *Y* as the following conditions hold:

- $Val_{X,Z} = low \wedge Val_{Y,Z} = high$
- $\forall i \le p, \quad Val_{X, b_i} = v_i$
- $\forall i \le p, \quad Val_{Y, b_i} = w_i$
- $\forall attr \in \{IV \cap L_R\}, \quad Val_{X, attr} = Val_{Y, attr}$
- X support R_1
- Y support R_2

The above conditions simply say that an object *X* supports an action rule *R* if there is another instance *Y* that it is possible to apply *R* on *X* and convert it to *Y*. This definition is close to that described in [12] but with a change in defining new concept of valuable and invaluable attributes instead of flexible and stable attributes.

```
ARCM ($R_1$, $R_2$, $IV$, $V$) {
    $R$= {};
    for each $A \in [IV \cap (L_{R_2} - L_{R_1})]$ do
        $R = R \cup [A, ?Val_{R_2, A}]$;
    for each $A \in [V \cap (L_{R_2} \cap L_{R_1})]$ do
        If ( $Val_{R_1, A} = vi \ \&\& Val_{R_2, A} = wi$ ) then
            $R = R \cup [A, ?Val_{R_1, A} \rightarrow Val_{R_2, A}]$;
    for each $A \in [V \cap (L_{R_2} - L_{R_1})]$ do
        $R = R \cup [A, ? \rightarrow Val_{R_2, A}]$;
    return $R$;
}
```

Algorithm 1: The algorithm of constructing an action rule from a pair of classification rules.

The net profit of an action rule *R* defines as follow:

$$PNet(r) = \sum \forall x \in set\ of\ objects\ that\ \text{support}\ ?\ PNet(x, r) \qquad (1)$$

where PNet(x,r) is the net profit that gained from applying action rule *R* on the instance *X* and can be computed using Eq. (2):

$$PNet(x, r) = P(L \rightarrow H) - \sum_i Cost_i(x) \qquad (2)$$

where $Cost_i(x)$ is the cost of changing ith attribute of instance *X* based on action rule *R*. For example, consider rule r as below that is extracted from two rules which exist in first row and last row of Table II.

$r : (B,1) \wedge (C, 2 \rightarrow 1) \wedge (F, 2 \rightarrow 1) \rightarrow (Z, L \rightarrow H)$

Based on above formulas its net profit can be computed as follow:

$PNet(r) = PNet(x1, r) + PNet(x2, r) = 2(10 - (8 + 3)) = -2$

We assume $P(L \rightarrow H)$ = 10; the negative value for net profit shows this rule is not cost-effective.

B. Discovering Cost-Effective Action Rule Algorithm

In this section we present a new algorithm for discovering Cost-Effective action rules called CEARDM.

The algorithm works in two phases: 1- constructing a cost-effective action tree from previously mined classification rules, 2- Extracting cost-effective action rules from the action tree. Action tree partitions rules based on invaluable attributes. Each leaf of action tree will be containing set of rules that the values of their invaluable attributes have no conflict with each other. Two rules have no conflict in their invaluable attributes if the values of their invaluable attributes are the same or not important in at least one of them. After constructing the action tree, algorithm extracts action rules from the rules placed in leaves of the action tree using algorithm 1 and finally select the most cost-effective of them using Eq. (1). The complete algorithm is shown in Algorithm 2.

Let us take Table I as an example of a decision table T that cost matrixes of its attributes are presented in Fig. 1. Assume now that our goal is to re-classify some objects from the class H into the class L.

We represent the set R of classification rules extracted from T as a table (see Table 2). First, CEARDM method finds set of invaluable attributes and sorts them descending based on theirs FF values. Since in our example FF_A=100, FF_B=20, FF_C=8, FF_D=18, FF_E=1, FF_F=3, this set will be like IV= {A, B, D}. Then the algorithm calls CEAT method. In this step the construction of a cost-effective action tree starts with all extracted classification rules as the root of the tree (T_1 in Fig. 2). Then CEAT select an unmarked attribute from AIV which number of its different values in current node be more than one, then marks it and creates a branch for each of its values. Then it places rules on corresponding branches based on their value of selected attribute. Rules with "don't care" value for selected attribute will be placed in all branches.

```
Define EndTables as a global empty set of tables;
CEARDM (T, Attributes, CostMatrixes) {
Input:
 T: Table of instances
 Attributes: Set of all attributes
 CostMatrixes: Set of all Cost Matrixes of attributes
Output:
 print all cost-effective action rules

  IV = an empty list of attributes ;
  for each A   Attributes do
      $FF_A$ = minimum value of $CM_A$ ;
      if ( $p(L \rightarrow H) < FF_A$ ) then
          Add A to IV
  Sort IV descending based on the FF values ;
  UnMarked = IV;
  Marked = an empty list of attributes ;
  CEAT ( T , UnMarked , Marked ) ;
  CEAR ( CostMatrixes ) ;
}

CEAT ( T , UnMarked , Marked ) {
  if ( $\exists$ $r_i$, $r_j \in$ rows of T that decision value of $r_i \neq$ decision value of $r_j$ )
  then
     A= first attribute of UnMarked;
    NV= number of different values of attribute A in T
    while ( A ! = null && NV < 1) do
       Move A from UnMarked to Marked;
       A =  next attribute of UnMarked;
    if ( A ! = null ) then
      Move A from UnMarked to Marked;
      for each vi $\in$ set of different values of attribute A in T do
         t = empty table;
         Add to t each ri $\in$ rows of T that have vi or null value for attribute A;
        CEAT ( t , UnMarked , Marked );
    else
      Add T to EndTables ;
}

CEAR ( CostMatrixes ) {
 for each table t in EndTables do
    for each row $r_i$ of t with decision value of L do
      for each row $r_j$ of t with decision value of H do
        AR = Construct an action rule by Algorithm1;
        NP = Calculate the net profit of AR by Eq. (1);
        if  (NP > 0)
           Print AR as a cost-effective action rule;
}
```

Algorithm2: Cost-Effective Action Rule Discovery Method

In our example the root node selection is attribute A, so the table is divided into two sub-tables: one table contains rules with value "1" or "don't care" for attribute A and the other contains rules with value "2" or "don't care" for A. Then the process is repeated recursively for each child node. If at any time all instances at one node have the same decision value, then the algorithm stops growing the tree through that node (T_3 in Fig. 2). When all invaluable attributes are selected, tables in leaf nodes which contain at least two rules with different decision values will be added to *EndTables* (T_4, T_6, and T_7 in Fig. 2). In second phase cost-effective action rules will be extracted from each table in *EndTables*.

The following action rule is extracted from T_4 and it is considered as cost-effective action rule because its net profit is positive:

$$r_1 : (D,2) \wedge (C,2) \wedge (E,1 \rightarrow 2) \wedge (F, \rightarrow 1) \rightarrow (Z, L \rightarrow H)$$

$$PNet(r_1) = PNet(x_3, r_1) = 9$$

T6 results the following action rule that is not considered as cost-effective action rule because its net profit is not positive.

$$r_2 : (B,1) \wedge (C,2 \rightarrow 1) \wedge (F,2 \rightarrow 1) \rightarrow (Z, L \rightarrow H)$$

$$PNet(r_2) = PNet(x_1, r_2) + PNet(x_2, r_2) = -2$$

Also r3 that is a cost-effective action rule is gained from T7:

$$r_3 : (D,2) \wedge (C, \rightarrow 2) \wedge (E,1 \rightarrow 2) \wedge (F, \rightarrow 1) \rightarrow (Z, L \rightarrow H)$$

$$PNet(r_3) = PNet(x_5, r_3) + PNet(x_6, r_3) + PNet(x_{11}, r_3) +$$

$$PNet(x_{13}, r_3) = 12$$

Presented algorithm returns all cost-effective action rules without any unnecessary comparison. It selects an attribute with maximum FF value in each level of tree and dividing rules based on it. If the cost of changing selected attribute is more than resulting profit of changing decision attribute from low to high, algorithm continues. Selecting attributes and dividing table will be continued until FF value of the selected attribute becomes less than $P(L \rightarrow H)$. In this step more dividing the table may cause losing some cost-effective action rules. After stopping the branching process algorithm will extract all cost-effective action rules from resulted tables in leaves of action tree. Then it is possible to sort them based on their net profit and select the most cost-effective ones.

IV. Proof

In this section we show the correctness of above algorithm. As it mentioned before our aim is finding all of profitable action rules. We show that above algorithm returns all of profitable action rules without any unnecessary comparison.

The algorithm selects an attribute with maximum FF value in each level of tree for dividing rules based on it, in other words it selects the attribute that the minimum cost of changing its values is maximum in comparison with other attributes. Now if this cost of selected attribute is more than resulting profit of changing decision attribute from low position to high position, the algorithm will continue for dividing rules. This is because of value changing for this attribute is not cost-effective. Selecting attributes and dividing rules will be continued until FF value of the selected attribute becomes less than $P(L \rightarrow H)$. In this stage more dividing rules may cause losing some profitable action rules such as an action rule which include the change with the minimum cost for changing the values of selected attribute. So that the algorithm stops dividing the table at this step and goes to the phase of extracting action rules because of not losing profitable action rules. We follow an example for further understanding. Suppose Table 3 as a decision table, and cost matrix of all its attributes that is presented in Fig. 3. We start with set R (Table 4) of classification rules extracted from Table 3 in root of tree and continue according to CEARM algorithm (Fig. 4). Since IV= {A}, this table is divided into two sub-tables base on A's different values. Then because of non existence of more IV attribute, dividing sub-tables is stopped in second level of tree. It results following candidate action rules.

$$r1:(B,1 \rightarrow 3) \wedge (C, \rightarrow 2) \rightarrow (Z, L \rightarrow H), PNet(r_1) = -2$$

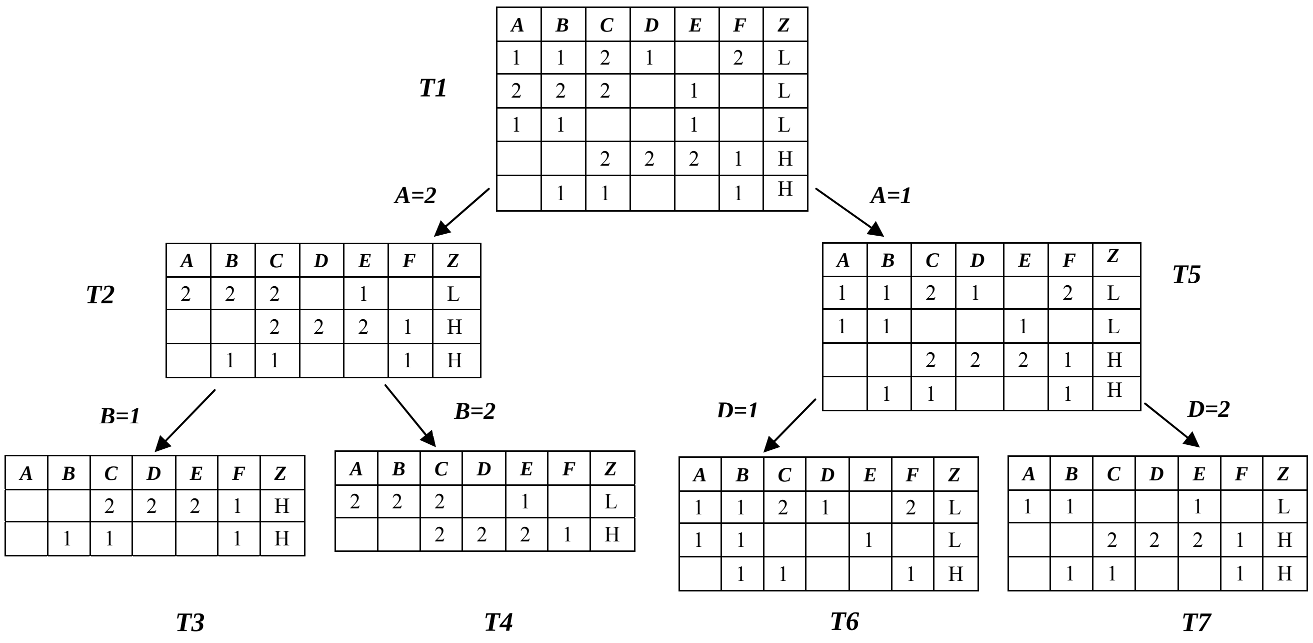
$$r2:(B, \rightarrow 1) \wedge (C,1 \rightarrow 0) \rightarrow (Z, L \rightarrow H), PNet(r_2) = 3$$

$$r3:(B, \rightarrow 3) \wedge (C,1 \rightarrow 2) \rightarrow (Z, L \rightarrow H), PNet(r_1) = -16$$

Now consider more dividing and stopping in next levels (Fig. 5). It causes losing some candidate action rules that may be profitable. For example consider we select attribute with maximum FF among unselected attributes ({B, C}) and continue dividing sub-tables based on its different values then stop in third level. It results only one following candidate action rule.

$$r1:(B,1 \rightarrow 3) \wedge (C, \rightarrow 2) \rightarrow (Z, L \rightarrow H), PNet(r_1) = -2$$

Also stopping in previous level of second level not only doesn't lead to be found any more cost-effective action rules than DCEAR, but also need extra unnecessary comparisons. So DCEAR algorithm finds all of profitable action rules without any unnecessary comparison.



T1

A	B	C	D	E	F	Z
1	1	2	1		2	L
2	2	2		1		L
1	1			1		L
		2	2	2	1	H
	1	1			1	H

T2

A	B	C	D	E	F	Z
2	2	2		1		L
		2	2	2	1	H
	1	1			1	H

T5

A	B	C	D	E	F	Z
1	1	2	1		2	L
1	1			1		L
		2	2	2	1	H
	1	1			1	H

T3

A	B	C	D	E	F	Z
		2	2	2	1	H
	1	1			1	H

T4

A	B	C	D	E	F	Z
2	2	2		1		L
		2	2	2	1	H

T6

A	B	C	D	E	F	Z
1	1	2	1		2	L
1	1			1		L
	1	1			1	H

T7

A	B	C	D	E	F	Z
1	1			1		L
		2	2	2	1	H
	1	1			1	H

Figure. 2. Cost-effective action tree for discussed example

TABLE 3: A SAMPLE DECISION TABLE

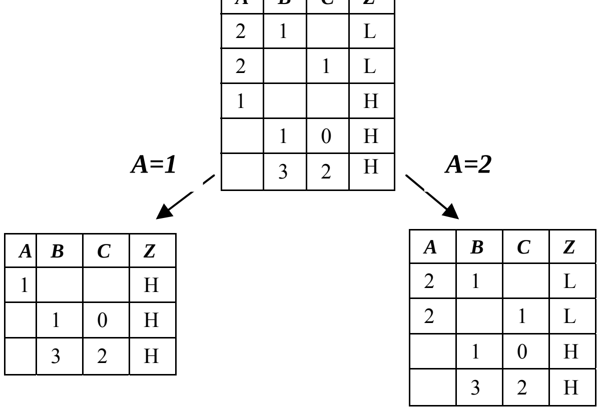
	A	B	C	Z
X1	2	1	2	L
X2	2	1	2	L
X3	1	1	0	H
X4	1	1	0	H
X5	2	3	2	H
X6	2	3	2	H
X7	2	1	1	L
X8	2	1	1	L
X9	2	2	1	L
X10	2	3	0	L
X11	1	1	2	H
X12	1	1	1	H

C	0	1	2
0	0	20	10
1	8	0	11
2	50	40	0

B	1	2	3
1	0	3	5
2	3	0	3
3	3	20	0

A	1	2
1	0	100
2	100	0

Figure. 3. Cost matrix



A	B	C	Z
2	1		L
2		1	L
1			H
	1	0	H
	3	2	H

A	B	C	Z
1			H
	1	0	H
	3	2	H

A	B	C	Z
2	1		L
2		1	L
	1	0	H
	3	2	H

Figure. 4. Cost-effective action tree

TABLE 4: SET OF RULES R WITH SUPPORTING OBJECTS

	A	*B*	*C*	*Z*
x3,x4,x11,x12	1			L
x1,x2,x7,x8	2	1		L
x7,x8,x9	2		1	H
x3,x4		1	0	H
x5,x6		3	2	H

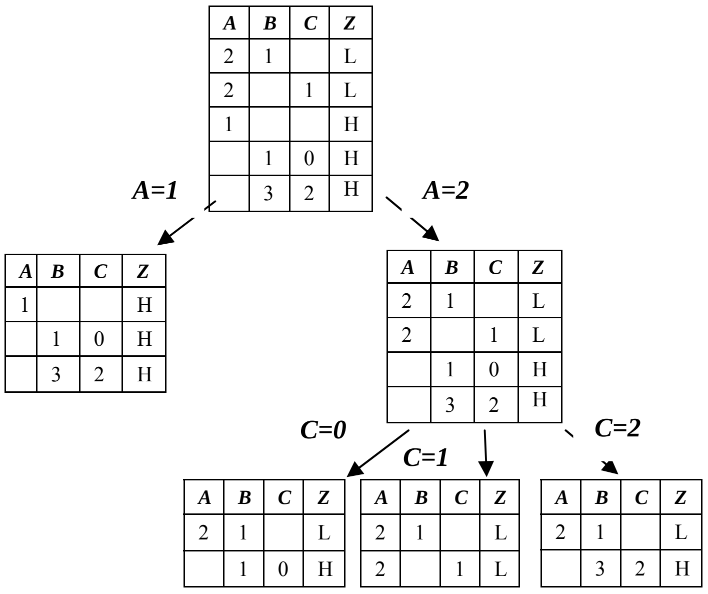


Figure. 5. Action tree

V. IMPROVEMENT

In this section we purpose an idea for improving CEAT algorithm. By this idea we can decrease number of iterations for above algorithm.

We mentioned before that CEAT marks next unmarked attribute in AIV set while finds an attribute that the number of its different values in current table be more than one, then marks it and for each of these values a branch will be created. In this step of algorithm we can make a change and create a branch only for useful values of selected attribute in current table. Useful values of an attribute in a table are values that for each of them in the table either exist at least two rules with different decision values that both of them have a value for the attribute, Or exist at least two rules with different decision values one with a value for the attribute and other with a don't care value for it. It causes to identify values that lead to create a table with same decision value earlier and prevent for creating table and recalling the algorithm for them. For example for T_2 in Fig. 2 we selected attribute B for dividing table and created a branch for each of its values. As you see T_3 has two entries with same decision values. But by mentioned change in algorithm we can check it in previous level and prevent from creating the table and recalling algorithm for it.

VI. CONCLUSION

Discovering useful knowledge from large database and acting on that knowledge is becoming increasingly important in today's competitive world. This knowledge is an active knowledge and can be provided by actionable patterns by actionable knowledge discovery methods. Action rules are one of the most effective actionable patterns that have attracted a lot of attentions recently.

In this work cost-effective action rules and a new method for discovering them is presented.. Not like traditional action rules, cost-effective action rules consider cost of an action in addition to its profit. To implement the approach a cost matrix for each attribute is created and integrated it into action rule mining process.

The method reported in this paper can integrate more background knowledge into mining process and therefore can find more useable actions. The detailed algorithm along with step by step examples has been presented in to show the functionality and effectiveness of the new method.

REFERENCES

[1] Zhengxiang Zhu, Jifa Gu, Wenxin Yang, Xingsen Li, "Toward Domain-Driven Data Mining", *Intelligent Information Technology Application Workshops, International Symposium, IEEE*, pp. 44-48, 2008.

[2] Longbing Cao, "Domain Driven Data Mining", *International Conference on Data Mining Workshops, IEEE*, pp. 74-76, 2008.

[3] L. B. Cao, C. Q. Zhang, "Domain-driven, actionable knowledge discovery," *IEEE Intelligent Systems*, vol.22, pp.78–88, July 2007.

[4] I. Witten, E. Frank. *Data Mining, Practical Machine Learning Tools and Techniques*, Morgan Kaufman, 2005.

[5] K. McGarry, "A Survey of Interestingness Measures for Knowledge Discovery". *The Knowledge Engineering Review*, pp. 39-41, 2005.

[6] B. Liu, W. Hsu, and Y. Ma. "Identifying non-actionable association rules". *ACM New York*, NY, USA, 2001.

[7] Q. Yang, J. Yin, C. Ling, and Rong Pan, "Extracting Actionable Knowledge from Decision Trees", *IEEE Transactions on Knowledge and Data Engineering*, VOL. 19, NO. 1, pp. 43-56, January 2007.

[8] Q. Yang, J Y. Ling, T. Chen, "Postprocessing Decision Trees to Extract Actionable Knowledge", *Proceedings of the Third IEEE International Conference on Data Mining, IEEE*, pp. 685-688, 2003.

[9] K. Wang, Y. Jiang, A. Tuzhilin, "mining actionable patterns by role models", *Proceedings of the 22nd International Conference on Data Engineering, IEEE*, 2006.

[10] L-S. Tsay, Z W. Ra´s, " E-Action Rules ", Post-Proceeding of FDM'04 *Workshop Advances in soft Computing, Springer*, Berlin Heidelberg New York, pp. 277-288, 2006.

[11] L-S Tsay, Z W. Ras, "Action rules discovery: system DEAR2, method and experiments", *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 17, No. 1–2, pp. 119–128, 2005.

[12] Z W. Ra´s, L-S Tsay, "Mining E-Action Rules, System DEAR", *Studies in Computational*, pp. 289-298, 2008.

[13] Z W. Ra´s, A. Wieczorkowska, "Action rules: how to increase profit of a company", *Principles of Data Mining and Knowledge Discovery, Proceedings of PKDD'00 (Eds: DA. Zighed, J. Komorowski, J. Zytkow)*. Lyon, France, LNCS/LNAI, No. 1910, *Springer*, Berlin Heidelberg New York, pp. 587–592, 2000.

[14] L-S Tsay, Z W. Ras, A. Wieczorkowska (2004) "Tree-based algorithm for discovering extended action-rules (System DEAR2) ". In: *Proceedings of the IIS'2004 Symposium, Zakopane, Poland, Springer*, 459-464

[15] L-S Tsay, Z W. Ras, "Action rules discovery system DEAR3". In: *Esposito*, F., Ra´s, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 483–492. *Springer*, Heidelberg (2006)

[16] Zbigniew W. Ra and Agnieszka Dardzinska, "Action Rules Discovery Based on Tree Classifiers and Meta actions" J. Rauch et al. (Eds.): ISMIS 2009, LNAI 5722, pp. 66–75, 2009.

[17] H. Geffner, J. Wainer, " Modeling action, knowledge and control. In: ECAI 98", *Proceedings of the 13th European Conference on AI, (Ed: Prade H)*. Wiley, New York, 532–536, 1998

[18] Z. Pawlak, "Rough sets-theoretical aspects of reasoning about data Algorithms for Packet Classification", Kluwer, Dordrecht, 1991.

[19] B. Liu, W. Hsu, S. Chen, "Using general impressions to analyze discovered classification rules", of *KDD97 Conference*. AAAI, Newport Beach, CA288 L.-S. Tsay and Z.W. Ra´s, 1997.

[20] Z. Pawlak, "Information systems – theoretical foundations", InformationSystems Journal, Vol. 6, pp. 205–218, 1981.

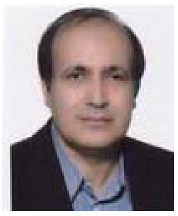
[21] Z. Zhu, J. Gu, W. Yang, X. Li, "Toward Domain-Driven Data Mining", *Intelligent Information Technology Application Workshops, International Symposium, IEEE*, pp. 44-48, 2008.

[22] L. Cao, "Domain Driven Data Mining", *International Conference on Data Mining Workshops, IEEE*, pp. 74-76, 2008.

[23] J. Han, M. Kamber, *data mining : concepts and techniques*, 2nd ed., Morgan Kaufman, 2006.

Nasrin kalanat born in Isfahan, Iran, 1986.
She is a M.Sc. student in Computer Software Engineering at Electrical and Computer Engineering Department of Isfahan University of Technology, Isfahan, Iran, from 2009.
Her research interests include machine learning, data base systems, data mining and knowledge discovery.

Pirooz Shamsinejadbabaki born in Tehran, Iran, 1984. He has a B.Sc. in Computer Software Engineering from Isfahan University of Technology, 2004. His M.Sc. is in Computer Architecture from IUT, 2006. He is a Ph.D. candidate in Computer Science at IUT from 2007.
His research interest is actionable knowledge discovery from large databases, domain driven data mining, text mining and genetic algorithms.



Dr Mohamad Saraee received his PhD from University of Manchester in Computation, MSc from University of Wyoming, USA in Software Engineering and BSc from ShahidBeheshti University, Iran.
His main areas of research are Intelligent databases, Mining advanced and complex data including medical and Bio, Text Mining and E-Commerce. He has published extensively in each of these areas and served on scientific and organizing committee on number of journals and conferences.