Constraint Based Action Rule Discovery with Single Classification Rules

Angelina Tzacheva[1] and Zbigniew W. Raś[2,3]

[1] University of South Carolina Upstate, Department of Informatics, Spartanburg, SC 29303
[2] University of North Carolina at Charlotte, Department of Computer Science, Charlotte, N.C. 28223
[3] Polish-Japanese Institute of Information Technology, 02-008 Warsaw, Poland

Abstract. Action rules can be seen as an answer to the question: what one can do with results of data mining and knowledge discovery? Some applications include: medical field, e-commerce, market basket analysis, customer satisfaction, and risk analysis. Action rules are logical terms describing knowledge about possible actions associated with objects, which is hidden in a decision system. Classical strategy for discovering them from a database requires prior extraction of classification rules which next are evaluated pair by pair with a goal to suggest an action, based on condition features in order to get a desired effect on a decision feature. An actionable strategy is represented as a term $r = [(\omega) \wedge (\alpha \rightarrow \beta)] \Rightarrow [\phi \rightarrow \psi]$, where ω, α, β, ϕ, and ψ are descriptions of objects or events. The term r states that when the fixed condition ω is satisfied and the changeable behavior $(\alpha \rightarrow \beta)$ occurs in objects represented as tuples from a database so does the expectation $(\phi \rightarrow \psi)$. With each object a number of actionable strategies can be associated and each one of them may lead to different expectations and the same to different reclassifications of objects. In this paper we will focus on a new strategy of constructing action rules directly from single classification rules instead of pairs of classification rules. It presents a gain on the simplicity of the method of action rules construction, as well as on its time complexity. We present A*-type heuristic strategy for discovering only interesting action rules, which satisfy user-defined constraints such as: feasibility, maximal cost, and minimal confidence. We, therefore, propose a new method for fast discovery of interesting action rules.

1 Introduction

There are two aspects of interestingness of rules that have been studied in data mining literature, objective and subjective measures [1], [5]. Objective measures are data-driven and domain-independent. Generally, they evaluate the rules based on their quality and similarity between them. Subjective measures, including unexpectedness, novelty and actionability, are user-driven and domain-dependent.

A. An et al. (Eds.): RSFDGrC 2007, LNAI 4482, pp. 322–329, 2007.

Action rules, introduced in [6] and investigated further in [10], [11], [8], are constructed from certain pairs of association rules. Interventions, defined in [3], are conceptually very similar to action rules.

The notion of a cost of an action rule, which is a subjective measure, was introduced in [11]. It is associated with changes of values of classification attributes in a rule. The strategy for replacing the initially extracted action rule by a composition of new action rules, dynamically built and leading to the same reclassification goal, was proposed in [11]. This composition of rules uniquely defines a new action rule. Objects supporting the new action rule also support the initial action rule but the cost of reclassifying them is lower or even much lower for the new rule. In [8] authors propose a new simplified strategy for constructing action rules. This paper presents a heuristic strategy for discovering interesting action rules which satisfy user-defined constraints such as: feasibility, maximal cost, and minimal confidence. There is a similarity between the rules generated by Tree-Based Strategy [10] and rules constructed by this new method.

2 Action Rules

In the paper by [6], the notion of an action rule was introduced. The main idea was to generate, from a database, special type of rules which basically form a hint to users showing a way to re-classify objects with respect to some distinguished attribute (called a decision attribute). Values of some attributes, used to describe objects stored in a database, can be changed and this change can be influenced and controlled by user. However, some of these changes (for instance *profit* can not be done directly to a decision attribute. In such a case, definitions of this decision attribute in terms of other attributes (called classification attributes) have to be learned. These new definitions are used to construct action rules showing what changes in values of some attributes, for a given class of objects, are needed to re-classify these objects the way users want. But, users may still be either unable or unwilling to proceed with actions leading to such changes. In all such cases, we may search for definitions of a value of any classification attribute listed in an action rule. By replacing this value of attribute by its definition extracted either locally or at remote sites (if system is distributed), we construct new action rules which might be of more interest to users than the initial rule [11]. We start with a definition of an information system given in [4].

By an information system we mean a pair $S = (U, A)$, where:

1. U is a nonempty, finite set of objects (object identifiers),
2. A is a nonempty, finite set of attributes i.e. $a : U \rightarrow V_a$ for $a \in A$, where V_a is called the domain of a.

Information systems can be seen as decision tables. In any decision table together with the set of attributes a partition of that set into conditions and decisions is given. Additionally, we assume that the set of conditions is partitioned into stable and flexible [6].

Attribute $a \in A$ is called stable for the set U if its values assigned to objects from U can not be changed in time. Otherwise, it is called flexible. *Place of birth*

is an example of a stable attribute. *Interest rate* on any customer account is an example of a flexible attribute. For simplicity reason, we consider decision tables with only one decision. We adopt the following definition of a decision table:

By a decision table we mean an information system $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$, where $d \notin A_{St} \cup A_{Fl}$ is a distinguished attribute called the decision. The elements of A_{St} are called stable conditions, whereas the elements of $A_{Fl} \cup \{d\}$ are called flexible. Our goal is to change values of attributes in A_{Fl} for some objects in U so the values of attribute d for these objects may change as well. Certain relationships between attributes from $A_{St} \cup A_{Fl}$ and the attribute d will have to be discovered first.

By $Dom(r)$ we mean all attributes listed in the *IF* part of a rule r extracted from S. For example, if $r = [(a_1, 3) \wedge (a_2, 4) \rightarrow (d, 3)]$ is a rule, then $Dom(r) = \{a_1, a_2\}$. By $d(r)$ we denote the decision value of rule r. In our example $d(r) = 3$.

If r_1, r_2 are rules and $B \subseteq A_{Fl} \cup A_{St}$ is a set of attributes, then $r_1/B = r_2/B$ means that the conditional parts of rules r_1, r_2 restricted to attributes B are the same. For example if $r1 = [(a1, 3) \rightarrow (d, 3)]$, then $r1/\{a1\} = r/\{a1\}$.

We assume that $(a, v \rightarrow w)$ denotes the fact that the value of attribute a has been changed from v to w. Similarly, the term $(a, v \rightarrow w)(x)$ means that the property (a, v) of an object x has been changed to property (a, w).

Assume now that rules r_1, r_2 are extracted from S and

$r_1/[Dom(r_1) \cap Dom(r_2) \cap A_{St}] = r_2/[Dom(r_1) \cap Dom(r_2) \cap A_{St}]$, $d(r_1) = k_1$, $d(r_2) = k_2$. Also, assume that $(b_1, b_2, \ldots, b_p)$ is a list of all attributes in $Dom(r_1) \cap Dom(r_2) \cap A_{Fl}$ on which r_1, r_2 differ and $r_1(b_1) = v_1$, $r_1(b_2) = v_2, \ldots$, $r_1(b_p) = v_p$, $r_2(b_1) = w_1$, $r_2(b_2) = w_2, \ldots$, $r_2(b_p) = w_p$.

By (r_1, r_2)-action rule we mean statement r:

$[r_2/A_{St} \wedge (b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \ldots \wedge (b_p, v_p \rightarrow w_p)] \Rightarrow [(d, k_1 \rightarrow k_2)]$.

Object $x \in U$ supports action rule r, if x supports the description $[r_2/A_{St} \wedge (b_1, v_1) \wedge (b_2, v_2) \wedge \ldots \wedge (b_p, v_p) \wedge (d, k_1)]$. The set of all objects in U supporting r is denoted by $U^{<r>}$. The term r_2/A_{St} is called the header of action rule.

Extended action rules, introduced in [10], form a special subclass of action rules. We construct them by extending headers of action rules in a way that their confidence is getting increased. The support of extended action rules is usually lower than the support of the corresponding action rules.

3 Action Rule Discovery from Single Classification Rule

Let us assume that $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$ is a decision system, where $d \notin A_{St} \cup A_{Fl}$ is a distinguished attribute called the decision. Assume also that $d_1 \in V_d$ and $x \in U$. We say that x is a d_1-object if $d(x) = d_1$. Finally, we assume that $\{a_1, a_2, ..., a_p\} \subseteq A_{Fl}$, $\{b_1, b_2, ..., b_q\} \subseteq A_{St}$, $a_{[i,j]}$ denotes a value of attribute a_i, $b_{[i,j]}$ denotes a value of attribute b_i, for any i, j and that

$r = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge [b_{[1,1]} \wedge b_{[2,1]} \wedge ..... \wedge b_{[q,1]}] \rightarrow d_1]$

is a classification rule extracted from S supporting some d_1-objects in S. By $sup(r)$ and $conf(r)$ we mean *support* and *confidence* of r, respectively. Class d_1 is a preferable class and our goal is to reclassify d_2-objects into d_1 class, where $d_2 \in V_d$.

By an action rule $r_{[d2\rightarrow d1]}$ associated with r and the reclassification task $(d, d_2 \rightarrow d_1)$ we mean the following expression [8]:

$r_{[d2\rightarrow d1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge$

$[(b_1, \rightarrow b_{[1,1]}) \wedge (b_2, \rightarrow b_{[2,1]}) \wedge ..... \wedge (b_q, \rightarrow b_{[q,1]})] \Rightarrow (d, d_2 \rightarrow d_1)]$.

In a similar way, by an action rule $r = [\rightarrow d_1]$ associated with r and the reclassification task $(d, \rightarrow d_1)$ we mean the following expression:

$r_{[\rightarrow d1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge$

$[(b_1, \rightarrow b_{[1,1]}) \wedge (b_2, \rightarrow b_{[2,1]}) \wedge ..... \wedge (b_q, \rightarrow b_{[q,1]})] \Rightarrow (d, \rightarrow d_1)]$.

The term $[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}]$, built from values of stable attributes, is called the header of action rule and its values can not be changed.

The support set of the action rule $r_{[d_2\rightarrow d_1]}$ is defined as:

$Sup(r_{[d_2\rightarrow d_1]}) = \{x \in U : (a_1(x) = a_{[1,1]}) \wedge (a_2(x) = a_{[2,1]}) \wedge ... \wedge (a_p(x) = a_{[p,1]}) \wedge (d(x) = d_2)\}$.

In the following paragraph we show how to calculate the confidence of action rules. Let $r_{[d_2\rightarrow d_1]}, r'_{[d_2\rightarrow d_3]}$ are two action rules extracted from S. We say that these rules are p-equivalent ($\approx$), if the condition given below holds for every $b_i \in A_{Fl} \cup A_{St}$:

if r/b_i, r'/b_i are both defined, then $r/b_i = r'/b_i$.

Let us take d_2-object $x \in Sup(r_{[d_2\rightarrow d_1]})$. We say that x positively supports $r_{[d_2\rightarrow d_1]}$ if there is no classification rule r' extracted from S and describing $d_3 \in V_d$, $d_3 \neq d_1$, which is p-equivalent to r, such that $x \in Sup(r'_{[d_2\rightarrow d_3]})$. The corresponding subset of $Sup(r_{[d_2\rightarrow d_1]})$ is denoted by $Sup^+(r_{[d_2\rightarrow d_1]})$. Otherwise, we say that x negatively supports $r_{[d_2\rightarrow d_1]}$. The corresponding subset of $Sup(r_{[d_2\rightarrow d_1]})$ is denoted by $Sup^-(r_{[d_2\rightarrow d_1]})$. By the confidence of $r_{[d_2\rightarrow d_1]}$ in S we mean:

$Conf(r_{[d2\rightarrow d1]}) = [card[Sup^+(r_{[d2\rightarrow d1]})]/card[Sup(r_{[d2\rightarrow d1]})]] \cdot conf(r)$.

4 Cost and Feasibility of Action Rules

Depending on the cost of actions associated with the classification part of action rules, business user may be unable or unwilling to proceed with them.

Assume that $S = (X, A, V)$ is an information system. Let $Y \subseteq X$, $b \in A$ is a flexible attribute in S and $b_1, b_2 \in V_b$ are its two values. By $\wp_S(b_1, b_2)$ we mean a number from $(0, +\infty]$ which describes the average cost of changing the attribute value b_1 to b_2 for any of the qualifying objects in Y. These numbers are provided by experts. Object $x \in Y$ qualifies for the change from b_1 to b_2, if $b(x) = b_1$. If the above change is not feasible, then we write $\wp_S(b_1, b_2) = +\infty$. Also, if $\wp_S(b_1, b_2) < \wp_S(b_3, b_4)$, then we say that the change of values from b_1 to b_2 is more feasible than the change from b_3 to b_4.

Let us assume that

$r = [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \ldots \wedge (b_p, v_p \rightarrow w_p)] \Rightarrow (d, k_1 \rightarrow k_2)$

is an action rule.

By the *cost* of r in S, denoted by $cost(r)$, we mean the value $\sum\{\wp_S(v_k, w_k) : 1 \leq k \leq p\}$. We say that r is *feasible*, if $cost(r) < \wp_S(k_1, k_2)$.

Now, let us assume that $R_S[(d, k_1 \rightarrow k_2)]$ denotes the set of action rules in S having the term $(d, k_1 \rightarrow k_2)$ on their decision side. Sometimes, for simplicity reason, attribute d will be omitted. An action rule in $R_S[(d, k_1 \rightarrow k_2)]$ which has the lowest cost value may still be too expensive to be of any help. Let us notice that the cost of an action rule $r = [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \ldots \wedge (b_p, v_p \rightarrow w_p)] \Rightarrow (d, k_1 \rightarrow k_2)$ might be high because of the high cost value of one of its sub-terms in the conditional part of the rule. Let us assume that $(b_j, v_j \rightarrow w_j)$ is that term. In such a case, we may look for an action rule in R_S $[(b_j, v_j \rightarrow w_j)]$, which has the smallest cost value. Assume that

$r_1 = [[(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge \ldots \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})] \Rightarrow$
$(b_j, v_j \rightarrow w_j)]$ is such a rule which is also feasible in S.

Now, we can compose r with r_1 getting a new feasible action rule:

$[(b_1, v_1 \rightarrow w_1) \wedge \ldots \wedge [(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge \ldots \wedge$
$(b_{jq}, v_{jq} \rightarrow w_{jq})] \wedge \ldots \wedge (b_p, v_p \rightarrow w_p)] \Rightarrow (d, k_1 \rightarrow k_2)$.

Clearly, the cost of this new rule is lower than the cost of r. However, if its support in S gets too low, then such a rule has no value to the user. Otherwise, we may recursively follow this strategy trying to lower the cost of re-classifying objects from the group k_1 into the group k_2. Each successful step will produce a new action rule which cost is lower than the cost of the current rule. Obviously, this heuristic strategy always ends.

5 A*-Type Algorithm for Action Rules Construction

Let us assume that we wish to reclassify objects in S from the class described by value k_1 of the attribute d to the class k_2.

The term $k_1 \rightarrow k_2$ jointly with its cost $\wp_S(k_1, k_2)$ is stored in the initial node n_0 of the search graph G built from nodes generated recursively by feasible action rules taken initially from R_S [(d, $k_1 \rightarrow k_2$)] .

For instance, the rule

$r = [[(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \ldots \wedge (b_p, v_p \rightarrow w_p)] \Rightarrow (d, k_1 \rightarrow k_2)]$
applied to the node $n_0 = \{[k_1 \rightarrow k_2, \wp_S(k_1, k_2)]\}$ generates the node
$n_1 = \{[v_1 \rightarrow w_1, \wp_S(v_1, w_1)], [v_2 \rightarrow w_2, \wp_S(v_2, w_2)], \ldots, [v_p \rightarrow w_p, \wp_S(v_p, w_p)]\}$
and from n_1 we can generate the node $n_2 = \{[v_1 \rightarrow w_1, \wp_S(v_1, w_1)],$
$[v_2 \rightarrow w_2, \wp_S(v_2, w_2)], \ldots, [v_{j1} \rightarrow w_{j1}, \wp_S(v_{j1}, w_{j1})], [v_{j2} \rightarrow w_{j2}, \wp_S(v_{j2}, w_{j2})],$
$\ldots, [v_{jq} \rightarrow w_{jq}, \wp_S(v_{jq}, w_{jq})], \ldots, [v_p \rightarrow w_p, \wp_S(v_p, w_p)]\}$
assuming that the action rule
$r_1 = [[(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge \ldots \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})] \Rightarrow$
$(b_j, v_j \rightarrow w_j)]$ from $R_S[(b_j, v_j \rightarrow w_j)]$ is applied to n_1.
This information can be written equivalently as:
$r(n_0) = n_1$, $r_1(n_1) = n_2$, $[r_1 \circ r](n_0) = r_1(r(n_0)) = n_2$.

By $Dom_S(r)$ we mean the set of objects in S supporting r.

Search graph G is dynamically built by applying action rules to its nodes. Its initial node n_0 contains information given by the user. Any other node n in G shows an alternative way to achieve the same reclassification with a cost that is lower than the cost assigned to all nodes which are preceding n in G. Clearly, the

confidence of action rules labelling the path from the initial node to the node n is as much important as the information about reclassification and its cost stored in node n.

The $A^{\star}$-type strategy for identifying a node in G, built for a desired reclassification of objects in S, with a cost possibly the lowest among all the nodes reachable from the node n, was given in [11]. This strategy was controlled by three threshold values: λ_1 - minimum confidence of action rules, λ_2 - maximum cost of action rules, and λ_3 - feasibility of action rules. The last threshold was introduced to control the minimal acceptable decrease in the cost of action rule to be constructed. If the search is stopped by threshold λ_1, then we do not continue the search along that path. If the search is stopped by threshold λ_2, then we can either stop or continue the search till it is stopped by threshold λ_1.

Assume that N is the set of nodes in graph G for S and n_0 is its initial node.

For any node $n \in N$, by $F(n) = (Y_n, \{[v_{n,j} \rightarrow w_{n,j}, \wp_S(v_{n,j}, w_{n,j}, Y_n)]\}_{j \in In})$ we mean its domain (set of objects in S supporting r, the reclassification steps for objects in Y_n and their cost, all assigned by reclassification function F to the node n, where $Y_n \subseteq X$.

The cost of node n is defined as: $cost(n) = \Sigma\{\wp_S(v_{n,j}, w_{n,j}, Y_n) : j \in In\}$.

We say that action rule r is applicable to a node n if:

$[Y_n \cap Dom_S(r) \neq \emptyset]$ and $[(\exists k \in In)[r \in R_S[v_{n,kj} \rightarrow w_{n,k}]]]$.

If node n_1 is a successor of node n in G obtained by applying the action rule r to n, then $Y_{n_1} = Y_n \cap Dom_S(r)$.

We assume here that the cost function $h(n_i) = \lceil [cost(n, Y_i) - \lambda_2]/\lambda_3 \rceil$ is associated with any node n_i in G. It shows the maximal number of steps that might be needed to reach the goal from the node n_i.

By $conf(n)$, we mean the confidence of action rule associated with node n.

A search node in a graph G associated with node m is a pair

$p(m) = ([conf(m), f(m)], [m, n_1, n_2, n_o])$, where $f(m) = g(m) + h(m)$ and $g(m)$ is the cost function defined as the length of the path $[m, n_1, n_2, n_o]$ in G (without loops) from the initial state n_o to the state m.

The search node associated with the initial node n_o of G is defined as $([conf(n_o), f(n_o)], [n_o])$. It is easy to show that $f(m)$ is admissible and never overestimates the cost of a solution through the node m.

6 New A*-Type Algorithm for Action Rules Construction

In this section we propose a modified version of A*-type heuristic strategy discussed in Section 5 which is based on the method of constructing action rules directly from single classification rules instead of their pairs [8]. It presents a gain on the simplicity of the method of action rules construction, as well as on its time complexity.

First, we introduce the notion of a cost linked with the attribute value itself as $\wp_S(b_1)$, where $b_1 \in V_b$, which again is a number from *(0, +∞]* describing the average cost associated with changing any value of attribute b to value b_1.

Next, assume that

$R = [[(a, a_1) \wedge (b, b_1) \wedge (c, c_1) \wedge (e, e_1) \wedge (m, m_1) \wedge (k, k_1) \wedge (n, n_1) \wedge (r, r_1)] \rightarrow (d, d_1)]$ is a classification rule extracted from S:

Assume that attributes in $St(R) = \{a, b, c, e\}$ are stable and in $Fl(R) = \{m, k, n, r\}$ flexible. Also, assume that class $d_1 \in V_d$ is of highest preference. The rule R defines the concept d_1. Assume that $V_d = \{d_1, d_2, d_3, d_4\}$.

Clearly, there may be other classification rules that define concept d_1. We pick the rule which has the lowest total cost on the flexible part, i.e. the sum of cost of all flexible attributes $\sum\{\wp_S(Fl(R)_i) : i = m, k, \ldots, r\}$ is minimal.

Next, we are picking objects from X which have property, let's say, d_2 i.e. objects of class d_2, which satisfy the header of stable attribute values in R:

$Y = \{x : a(x) = a_1, b(x) = b_1, c(x) = c_1, e(x) = e_1, d(x) = d_2\}$

In order to 'grab' these objects into d_1, we construct action rule:

$[(a_1 \wedge b_1 \wedge c_1 \wedge e_1] \wedge [(m, \rightarrow m_1) \wedge (k, \rightarrow k_1) \wedge (r, \rightarrow r_1)] \Rightarrow (d, d_2 \rightarrow d_1)$

In other words, if we make the specified changes to the attributes in $Fl(R)$, the expectation is that the objects in Y will move to the desired class d_1. Looking at the changes needed, the user may notice that the change $(k, \rightarrow k_1)$ is the worst, i.e. it has the highest cost, and it contributes most to the cost of the sum (total cost) of all changes. Therefore, we may search for new classification rules, which define the concept k_1, and compose the feasible action rule $R_1 = [St(R_1)] \wedge [Fl(R_1)]$ which suggests the reclassification to k_1 at the lowest cost, where $St(R) \subseteq St(R_1)$. As defined earlier, such action rule will be feasible if the sum (total cost) of all changes on the left hand side of the rule is lower, than the right side. Therefore, the action rule R_1 will specify an alternative way to achieve the reclassification to k_1 at a cost lower than the currently known cost to the user. Next, we concatenate the two action rules R and R_1 by replacing $(k, \rightarrow k_1)$ in R, with $[Fl(R_1)]$, and modifying the header to include $St(R) \cup St(R_1)$.

$[(a_1 \wedge b_1 \wedge c_1 \wedge e_1) \wedge St(R_1)] \wedge [(m, \rightarrow m_1) \wedge Fl(R_1) \wedge (r, \rightarrow r_1)] \Rightarrow (D, d_2 \rightarrow d_1)$.

Clearly, there may be many classification rules that we can choose from. We only consider the ones which stable part does not contradict with *St(R)*. Among them, we choose rules with a minimal number of new stable attributes, as each time we add a new stable attribute to the current rule we may decrease the total number of objects in Y which can be moved to the desired class d_1. In relation to flexible attributes, they have to be the same on the overlapping part of a new classification rule and the rule R. This may further decrease the number of potential objects in Y which can be moved to the desired class d_1.

Therefore, we need a heuristic strategy, similar to the one presented in the previous section for classical action rules, to look for classification rules to be concatenated with R and which have the minimal number of new stable attributes in relation to R and minimal number of new flexible attributes jointly with flexible attributes related to the overlapping part with R.

We propose a modified version of $A^\star$-algorithm we saw in the previous section. Again, we assume that user will provide the following thresholds related to action rules: λ_1 - minimum confidence, λ_2 - maximum cost, and λ_3 - feasibility.

Clearly, it is expensive to build the complete graph G and next search for a node of the lowest cost satisfying both thresholds λ_1, λ_2. The heuristic value associated with a node n in G is defined as $h(n) = \lceil [cost(n) - \lambda_2]/\lambda_3 \rceil$. It shows the maximal number of steps that might be needed to reach the goal. The cost function $g(m)$ is defined as the length of the path in G (without loops) from the initial state n_o to the state m. It is easy to show that $f(m) = g(m) + h(m)$ is admissible and never overestimates the cost of a solution through the node m.

7 Conclusion and Acknowledgements

The new algorithm for constructing action rules of the lowest cost is a significant improvement of the algorithm presented in [11] because of its simplicity in constructing headers of action rules and because the concatenation of action rules is replaced by concatenation of classification rules.

This research was partially supported by the National Science Foundation under grant IIS-0414815.

References

1. Adomavicius, G., Tuzhilin, A. (1997) Discovery of actionable patterns in databases: the action hierarchy approach, in **Proceedings of KDD′97 Conference**, Newport Beach, CA, AAAI Press
2. Hilderman, R.J., Hamilton, H.J. (2001) **Knowledge Discovery and Measures of Interest**, Kluwer
3. Greco, S., Matarazzo, B., Pappalardo, N., Slowiński, R. (2005) Measuring expected effects of interventions based on decision rules, in **Journal of Experimental and Theoretical Artificial Intelligence**, Taylor Francis, Vol. 17, No. 1-2
4. Pawlak, Z., (1991) Information systems - theoretical foundations, in **Information Systems Journal**, Vol. 6, 205-218
5. Silberschatz, A., Tuzhilin, A., (1995) On subjective measures of interestingness in knowledge discovery, in **Proceedings of KDD′95 Conference**, AAAI Press
6. Raś, Z., Wieczorkowska, A. (2000) Action Rules: how to increase profit of a company, in **Principles of Data Mining and Knowledge Discovery**, LNAI, No. 1910, Springer, 587-592
7. Raś, Z.W., Tzacheva, A., Tsay, L.-S. (2005) Action rules, in **Encyclopedia of Data Warehousing and Mining**, (Ed. J. Wang), Idea Group Inc., 1-5
8. Raś, Z.W., Dardzińska, A. (2006) Action rules discovery, a new simplified strategy, in **Foundations of Intelligent Systems**, F. Esposito et al. (Eds.), LNAI, No. 4203, Springer, 445-453
9. Tsay, L.-S., Raś, Z.W. (2005) Action rules discovery system DEAR, method and experiments, in **Journal of Experimental and Theoretical Artificial Intelligence**, Taylor & Francis, Vol. 17, No. 1-2, 119-128
10. Tsay, L.-S., Raś, Z.W. (2006) Action rules discovery system DEAR3, in **Foundations of Intelligent Systems**, LNAI, No. 4203, Springer, 483-492
11. Tzacheva, A., Raś, Z.W. (2005) Action rules mining, in **International Journal of Intelligent Systems**, Wiley, Vol. 20, No. 7, 719-736