Exploration of Knowledge Bases Inspired by Rough Set Theory

Agnieszka Nowak-Brzezinska and Alicja Wakulicz-Deja

Department of Computer Science, Institute of Computer Science, Silesian University, Poland
http://ii.us.edu.pl

Abstract. In this paper, we discuss some issues related to exploration of knowledge bases inspired by the rough set theory, which emerged 30 years ago, and is nowadays a rapidly developing branch of artificial intelligence. The partition of rules approach allows us to divide a large set of rules into smaller subsets that are easier to manage. Optimisation relies on reducing the number of rules searched in each run of inference. The paper presents the definition of the knowledge base model based on partition of rules and a modification of the forward inference algorithm for groups of rules generated by the partition strategy. It also contains a simple case study with an example of the partition of rules and the inference process for a simple knowledge base as well as experimental results.

Key words: knowledge base, inference process, goal driven, data driven, decision support system

1 Introduction

For the last twenty years, there has been an enormous interest in integrating database and knowledge-based system technologies to create an infrastructure for modern advanced applications. The result of it is a knowledge base (KB) system which consists of database systems extended with some kind of knowledge, usually expressed in the form of *rules*[1] - logical statements (implications) of the type „*if* condition$_1$ & ... & condition$_n$ *then* conclusion". The knowledge of experts, expressed in such a natural way, makes rules easily understood by people not involved in the expert system building. KBs are constantly increasing in volume, thus the knowledge stored as a set of rules is getting progressively more complex and searching such sets is an important data-mining task. When rules are not organized into any structure, the system is inefficient. There is a growing research interest in searching for methods that manage large sets of rules. Most of them use the clustering approach as well as joining and reducing rules [1, 2, 10, 11]. This paper presents a different idea, in which rules are divided into a number of groups based on similar conditions and/or conclusion. The process is called *partition of rules*[2] and is conducted by so-called *partition strategies*. In the

[1] Rules have been extensively used in knowledge representation and reasoning. It is very space efficient: only a relatively small number of facts needs to be stored in the KB and the rest can be derived by the inference rules.

[2] The idea is new but it is based on the authors' previous research, where the idea of *clustering rules* as well as creating so-called *decision units* was introduced[9, 5].

authors' opinion, *partition of rules* leads to the improvement of the inference process efficiency [5]. If we assume that the user wants to get an answer from the system as soon as possible, the proposed modification of the KB structure reduces the time of inference. Instead of searching within the whole set of rules (as in case of traditional inference processes), only representatives of groups are compared with the set of facts and/or hypothesis to be proven. The most relevant group of rules is selected and the exhaustive searching is done only within a given group. We show how exploration of complex KBs can be applied based on the partition of rules. Our motivation is two-fold. First, we are concerned with how to reason KBs effectively with a large number of rules. Second, we are concerned with improving the efficiency of reasoning over a set of rules by partitioning the set with respect to some strategy. To this end, we provide algorithms for partitioning and reasoning with such partition of rules.

The idea of *partition of rules* is implemented in the `kbExplorer` system[3], which is not limited to the inference optimization. The practical goal of the project is to create an expert system shell that allows for flexible switching between different inference methods based on knowledge engineer preferences[4].

The rest of the paper is organized as follows. In Section 2, the research background is introduced. It also contains a very general description of the proposed concept. Section 3 presents the basic definitions of the partition of rules and partition strategies, as well as the specification of partition's representative. It also includes the inspiration of the rough set theory. Section 4 describes the forward inference algorithm for the partition of rules which needs to modify the classical algorithm. The results of experiments are given in Section 5. A simple case study is also given. Finally, Section 6 concludes the paper.

2 Related Work and the Proposed Idea

There is a number of studies related to knowledge matching and rules modularization. Interesting and effective approaches to the problem with managing a large set of rules and necessity of its partitioning can be found in [1–3, 10, 11], where known systems like `CHIRA`, `XTT2` or `D3RJ` are described. In [2] the authors present `CHIRA` - an algorithm performing decision rules aggregation. Rules are joined if their conditional parts are built from the same conditional attributes or if the conditional attributes set of one rule is a subset of the conditional attributes set of the second one. Another joining algorithm, which operates on rules determined by the dominance-based rough set model, was proposed in [3]. Rules that lie close to each other are joined if their joint causes no deterioration of accuracy of the obtained rule. `XTT2` provides a modularized rule base, where rules working together are placed in one context corresponding to a single

[3] http://kbexplorer.ii.us.edu.pl/

[4] The user has the possibility of creating KBs using a special creator or by importing a KB from a given data source. The format of the KBs enables working with a rule set generated automatically based on the RST theory as well as with rules given apriori by the domain expert. The KB can have one of the following file formats: XML, RSES, TXT. It is possible to define attributes of any type: nominal, discrete or continuous. There are no limits for the number of rules, attributes, facts or the length of the rule.

decision table[10]. D3RJ is a method which produces more general rules, where each descriptor can be endorse subset of values [11]. In most of these tools, a global set of rules is partitioned by the system designer into several parts in an arbitrary way.

The idea of partition of rules, presented in this paper, is based on the previous authors' research concerning rules clusters and decision units [8, 9, 5]. The proposed algorithm is based on the same idea as the classical inference algorithm, but it uses groups' representatives (playing a role of general rules) instead of each single rule. If a representative matches the searched data, it means that its group probably contains a rule or rules that we are looking for. In this context it is necessary to rewrite the pattern matching algorithm in one point. Instead of finding rules that match exactly a given set of facts, we compare facts with rules' representatives and the most similar group of rules is selected. Further research is performed only on this group. The advantage is as follows: having n rules divided into k groups, only k groups' representatives are searched, while in the classical version of the inference process all n rules have to be analysed.

3 Partition of Rules Idea

The proposed concept assumes division of a KB into coherent subgroups of rules. Therefore, this section presents the basic concepts and notations of KB partitioning. We assume that a KB is a single set $\mathcal{R} = \{r_1, \ldots, r_i, \ldots, r_n\}$ without any order of n rules. Each rule $r_i \in \mathcal{R}$ is stored as a Horn's clause defined as: $r_i : p_1 \wedge p_2 \wedge \ldots \wedge p_m \rightarrow c$, where p_s is s-th literal (a pair of an attribute and its value) (a, v_i^a) $(s = 1, 2, \ldots, m)$. Attribute $a \in A$ may be a conclusion of rule r_i as well as a part of the premises. For every KB with n rules, the number of possible subsets is 2^n. Any arbitrarily created subset of rules $R \in 2^{\mathcal{R}}$ is called *partition of rules* (PR) and it can be generated by one of the many possible *partition strategies* (PS).

3.1 Definition of Partition of Rules

The partition of rules $PR = \{R_1, R_2, \ldots, R_k\}$, where k is the number of groups of rules in partition PR, R_j is j-th group of rules for $j = 1, \ldots, k$ and $PR \subseteq 2^R$ is generated by one of many possible partition strategies. We need a function $mc : R \times PR \rightarrow [0..1]$ to decide whether rule r_i belongs to group R_j or not. It is defined individually for every PS. If there is no doubt that rule r_i belongs to group R_j, $mc(r_i, R_j) = 1$. Otherwise, $mc(r_i, R_j) = 0$. It means that rule r_i definitely does not belong to group R_j. Values from the range 0 to 1 mean partial membership. In mathematical meaning, a *partition of rules* is a collection of subsets $\{R_j\}_{j \in J}$ of $\mathcal{R}$ such that: $\mathcal{R} = \bigcup_{j \in J} R_j$ and if $j, l \in J$ and $j \neq l$ then $R_j \cap R_l = \emptyset$. The subsets of rules are non-empty and every rule is included in one and only one of the subsets[5].

[5] However, we also consider the case, in which a given rule belongs to more than one group. In our future work we are going to extend the definition of the partition in this context.

3.2 Rough Set Theory and Indiscernibility Relation as Inspiration for *Partition of Rules* Idea

One of the tools for exploration of large rules data sets is analysing the similarities between rules. It leads naturally to their grouping and integration. The similarities of objects are examined by the *indiscernibility* relations used in the *rough set theory* (RST)[12, 13]. If $\mathcal{R}$ is the set of all rules in the KB, and r_i, r_f are arbitrary rules, they are said to be *indiscernible* by PR, denoted $r_i \widetilde{PR} r_f$, if and only if r_i and r_f have the same value on all elements in PR:

$$IND(PR) = \{(r_i, r_f) \in \mathcal{R} \times \mathcal{R} : \forall_{a \in PR}\, a(r_i) = a(r_f)\}.$$

As such, it induces a partition of $\mathcal{R}$ generated by PR, denoted PR^*[6]. Let us assume that the KB contains the following rules:
$r_1 : (a,1) \rightarrow (b,1)$ $r_2 : (a,1) \rightarrow (c,1)$ $r_3 : (d,1) \rightarrow (e,1)$
$r_4 : (d,1) \rightarrow (f,1)$ $r_5 : (b,1) \rightarrow (g,1)$ $r_6 : (c,1) \rightarrow (g,1)$
$r_7 : (e,1) \rightarrow (h,1)$ $r_8 : (f,1) \rightarrow (h,1)$ $r_9 : (g,1) \rightarrow (i,1)$
$r_{10} : (i,1) \rightarrow (k,1)$ $r_{11} : (i,1) \rightarrow (k,1)$ $r_{12} : (h,1) \rightarrow (j,1)$
$r_{13} : (j,1) \rightarrow (l,1)$ $r_{14} : (a,1) \wedge (j,1) \rightarrow (b,1)$
$r_{15} : (a,1) \wedge (j,1) \wedge (c,2) \rightarrow (b,1)$
$r_{16} : (a,1) \wedge (j,1) \wedge (c,2) \wedge (e,2) \rightarrow (b,1)$
$r_{17} : (a,1) \wedge (j,1) \wedge (c,2) \wedge (e,2) \rightarrow (b,1)$
For this KB it is possible to create different partitions in terms of different criteria. An *equivalence* relation on PR, defined by premises of rules in $\mathcal{R}$, is as follows: $\{PR\}^* = \{r_{17}, r_{16}, r_{15}, r_{14}, r_{13}, r_2, r_1\}, \{r_5\}, \{r_6\}, \{r_7\}, \{r_8\}, \{r_9\}, \{r_{12}\}, \{r_4, r_3\}, \{r_{11}, r_{10}\}$ while the *equivalence* relation on PR being conclusions of rules produces the partition: $\{PR\}^* = \{r_{17}, r_{16}, r_{15}, r_{14}, r_1\}, \{r_2\}, \{r_3\}, \{r_4\}, \{r_5, r_6\}, \{r_7, r_8\}, \{r_9\}, \{r_{12}\}, \{r_4, r_3\}, \{r_{11}, r_{10}\}, \{r_{13}\}$ whereas the *equivalence* relation on PR for $PR = (b,1)$ in conclusions is as follows: $\{PR\}^* = \{r_{17}, r_{16}, r_{15}, r_{14}, r_1\}$, $\{r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{12}, r_4, r_3, r_{11}, r_{10}, r_{13}\}$. This last example partition needs additional comments. Usually, when the partition strategy divides rules into the groups that match a given condition (in this case the literal $(b,1)$), the final partition is given by the so-called *selection* in which the first part of the partition is the group that matches a given condition, and the second group does not meet this condition [7].

3.3 Partition Strategies

Partition strategy is the general concept. It is possible to point out a number of different approaches for creating groups of rules. The most general division of partition strategies talks about two types of strategies: *simple* and *complex*. *Simple strategies*[8] allocate

[6] It does not necessarily contains a subset of attributes. It may also include the criterion of creating groups of rules, i.e. groups of rules with at least m number of premises or groups of rules with a premise containing a specific attribute or particular pair (attribute,value).

[7] $\{r_{17}, r_{16}, r_{15}, r_{14}, r_1\}$ contains literal $(b,1)$ in conclusion part, and $\{r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{12}, r_4, r_3, r_{11}, r_{10}, r_{13}\}$ does not contain this literal.

[8] It allows for partitioning the rules using the algorithm with time complexity not higher than $O(nk)$, where $n = |R|$ and $k = |PR|$. Simple strategies create final partition PR by a

every rule r_i to the proper group R_j, according to the value of the function $mc(r_i, R_j)$. The example is the strategy of finding a pair of rules the most similar in a given context, i.e. premises of rules. *Complex strategies* usually do not generate the final partition in a single step. It is defined by *a sequence of simple strategies* or *a combination of them*, or by *iteration* of a single simple strategy. An example is the strategy of creating *similarity based partition*, which stems from the method of *cluster analysis* used for rules clustering. It uses a simple strategy which finds pairs of the most similar rules many times. The process terminates if the similarity is no longer at least T [9]. In effect, we get k groups of rules $R_1, R_2, \ldots, R_l, \ldots, R_k$ such that $\bigwedge_{r_i, r_f \in R_l} sim(r_i, r_f) \geq T$. Each group R_l contains rules for which mutual similarity is at least equal to T. It is possible to obtain many different rules partitions. In this strategy, rules that form the same group are said to have the same or similar premises. It uses the similarity function $sim(r_i, r_f)$: $\mathcal{R} \times \mathcal{R} \rightarrow [0..1]$ which can be defined in a variety of ways, i.e. based on the conditional part of the rules as

$$sim(r_i, r_f) = \frac{|cond(r_i) \cap cond(r_f)|}{|cond(r_i) \cup cond(r_f)|},$$

where $cond(r_i)$ denotes the conditional part of rule r_i and $concl(r_i)$ its conclusion - respectively. The value of $sim(r_i, r_f)$ is equal to 1 if rules are formed by the same literals and 0 when they do not have any common literals. The more similar the rules are the closer to 1 is the value of $sim(r_i, r_f)$.

3.4 Similarity-Based Partition of Rules

The similarity-based partition strategy, described shortly above, allows for improving the efficiency of the inference process (see alg. Alg01). In the first step of the algorithm, a specific partition ($createSingletonGroups$), in which every rule forms a separate group, is created: $\forall_{R \in PR} |R| = 1$ ($\forall_{R \in PR} \mid \cup \{R : R \in PR\}| = n$ and $\forall_{R_i, R_j \in PR, i \neq j} R_i \cap R_j = \emptyset$). Further, it finds a pair of the most similar rules iteratively and join them into one group. Similarity is determined by using function *sim* based on the similarity of the conditional part of rules (or groups). This strategy is used to build the partition of 17 rules (presented in section 3.2). At the begining, each rule creates a single group: $R_1, \ldots, R_{17}$. Next, the following groups are created: $R_{18} = \{R_{17}, R_{16}\}$, $R_{19} = \{R_{18}, R_{15}\}$, $R_{20} = \{R_{19}, R_{14}\}$, $R_{21} = \{R_1, R_2\}$, $R_{22} = \{R_3, R_4\}$, $R_{23} = \{R_{10}, R_{11}\}$, $R_{24} = \{R_{20}, R_{13}\}$, $R_{25} = \{R_{24}, R_{21}\}$. This algorithm is based on the classical AHC algorithm discussed in [7, 9, 5]. The partition of rules achieved in this way has got the hierarchical structure thus it is necessary to search the tree of groups of rules [10].

Alg01: Similarity-based partition - an algorithm

Require: $\mathcal{R}$, sim, T;

---

single search of rules set R according to the value of $mc(r_i, R_j)$ function described above. For complex strategies time complexity is rather higher than any simple partition strategy.

[9] Let us assume that threshold value $0 \leq T \leq 1$ exists.

[10] The time complexity of this operation is $O(\log n)$ where n is the number of elements in the tree.

Ensure: $PR = \{R_1, R_2, \ldots, R_k\}$;
procedure ***createPartitions***($\mathcal{R}$, **var** PR, sim, T)
var R_i, R_j;
begin
PR = ***createSingletonGroups***($\mathcal{R}$);
$R_i = R_j = \{\}$;
while ***findTwoMostSimilarGroups***(sim, R_i, R_j, PR) $\geqslant T$ **do**
PR = ***rearrangeGroups***(R_i, R_j, PR);
end while
end procedure

3.5 Partition's Representatives - *Profiles*

The efficiency of the partition of rules and its usage in the inference process depends on the quality of the created partition. It is based on both: the cohesion and separation of the created groups of rules as well as on the representatives of the groups (*Profiles*). There can be found many different approaches to determine representatives for groups of rules. There are many possible forms of $Profile(R)$ [11]. We propose an approach inspired by the RST with *lower* and *upper* approximations[12]. It allows for defining two representatives: $\underline{Profile(R_j)}$ and $\overline{Profile(R_j)}$ for every group R_j. The lower approximation set of a profile of group R_j, is the union of all these literals from premises of rules, which certainly appear in R_j, while the upper approximation is the union of the literals that have a non-empty intersection with the subset of interest. As it was mentioned above, rule $r_i : p_1 \wedge p_2 \wedge \ldots \wedge p_m \rightarrow c$ has a conjunction of m literals in the conditional part. Each literal $p_c \in cond(r_i)$ is a premise of rule r_i. Thus,

$$\underline{Profile(R_j)} = \{\bigcap_i cond(r_i)\}$$

denotes the set of all literals p_c that are an intersection of the conditional part of each rule r_i in group R_j, whereas

$$\overline{Profile(R_j)} = \{\bigcup_i cond(r_i) : cond(r_i) \cap \bigcap_f cond(r_f) \neq \emptyset\}$$

for $f \neq i, r_i, r_f \in R_j$ contains the set of all premises of rules creating group R_j which have a non-empty intersection with the premises of other rules from the same group. For example, if in the KB presented above, group R_{22} contains two rules r_3 and r_4

[11] It may be (i) the set of all premises of rules that form group R, (ii) the conjunction of the selected premises of all rules included in a given group R as well as (iii) the conjunction of the premises of all rules included in a given group R.

[12] If X denotes a subset of universe element U ($X \subset U$) then the lower approximation of X in B ($B \subseteq A$) denoted as $\underline{BX}$, is defined as the union of all these elementary sets which are contained in X: $\underline{BX} = \{x_i \in U | [x_i]_{IND(B)} \subset X\}$. Upper approximation $\overline{BX}$ is the union of these elementary sets, which have a non-empty intersection with X:$\overline{BX} = \{x_i \in U | [x_i]_{IND(B)} \cap X \neq \emptyset\}$.

(groups R_3, R_4), $\underline{Profile(R_{22})} = \{(d,1)\}$ whereas $\overline{Profile(R_{22})} = \{(d,1),(b,2)\}$. The $BN(Profile(R_{22})) = \overline{Profile(R_{22})} - \underline{Profile(R_{22})} = \{(b,2)\}$. It means that literal $(d,1)$ appears in every rule in this group while $(b,2)$ makes rule r_3 distinct from r_4. In other words, $(d,1)$ certainly describes every rule in group R_{22} while $(b,2)$ possibly describes this group (there is at least one rule which contains this literal in the conditional part). Lower and upper approximations of the profile of a given group are very convenient in further searching. When we want to find a rules which certainly contains some literal, we have to match it to the lower approximation of the profile for every group. If we look for a rule that possibly contains some literal, we only have to match it to the upper approximation of the profiles of groups.

4 Inference Algorithm

In authors' opinion, both the proposed idea of the partition of rules and high quality of representatives' representation lead to improve the efficiency of the inference process. There are two inference algorithms: *forward* and *backward*[13]. Due to the limited size of the paper only the forward inference algorithm is discussed. The goal of the *forward inference* process is to find a set of rules with given data included in the premise part of a particular rule. The shorter the time needed to perform such a process, the better. During the inference process, premises of each and every rule are analysed to match them with the current set of facts (F). Then the subset of the conflict set is chosen consisting of the rules that are actually activated. Finally, previously selected rules are analysed and the set of facts is updated. The most time consuming procedure searches rules and finds those relevant to the given data (goal/facts) [4]. The aim is to reduce this process in the best way. Modification of the classical inference algorithm, proposed by the authors, is based on changes in the structure of the rules set. It is no longer a list of all rules. Now it forms a partition of rules with representatives. Thanks to this, the searching procedure is optimized. It needs less time to find the group and then the rule that match a given set of facts.

4.1 Forward Inference Algorithm Based on Partition of Rules

The forward inference algorithm based on the partition of rules (see Alg02) reduces search space by choosing only rules from a particular group of rules, which matches the current facts set. Thanks to the *similarity-based partition* strategy as the result we get groups of similar rules. In every step, the inference engine matches facts to groups' representatives and finds a group with the greater similarity value. To find a relevant group of rules, during the inference process, the maximal value of similarity between the set of facts F (and/or hypothesis to be proven) and the representatives of each group

[13] In case of *backward inference* we look for rules with a given hypothesis as a conclusion [6]. Examples of applying the inference in rule KBs are described in [8], where the modification of the backward inference algorithm for rule KBs is presented.

of rules, $Profile(R_j)$ is searched[14]. The more facts with hypothesis are included in the profile of group R_j, the greater the value of similarity. Exhaustive searching of rules is done only within the selected group. At the end, rules from the most promising group are activated. The inputs are: PR - groups of rules with the representatives and F-the set of facts. The output is F the set of facts, including possible new facts obtained through the inference. The algorithm uses temporary variable R, which is the set of rules that is the result of the previous selection.

Alg02: Modified forward inference algorithm

Require: $\mathcal{PR}$, $\mathcal{F}$;
Ensure: $\mathcal{F}$;
procedure ***forwardInference***($\mathcal{PR}$, **var** $\mathcal{F}$)
var R,r;
begin
R := ***selectBestFactMachingGroup***($\mathcal{PR}$, $\mathcal{F}$);
while $R \neq \emptyset$ **do**
r:=***selectRule***(R, strategy);
activeRule(r);
R := ***selectBestFactMachingGroup***($\mathcal{PR}$, $\mathcal{F}$);
end while
end procedure

selectBestFactMachingGroup is the procedure responsible for finding the most relevant group of rules. If the result of the selection is not empty ($R \neq \emptyset$), the *selectRule* procedure is commenced. It finds a rule, in which the premises part is fully covered by facts. If there is more than one rule, the strategy of the conflict set problem plays a significant role in the final selection of one rule. The *activeRule* procedure adds a new fact (conclusion of the activated rule) to the KB. Of course, it is necessary to remove the activated rule from further searching. Afterwards, the *selectBestFactMachingGroup* procedure is called again. The whole presented process is repeated in a while loop until the selection becomes empty. Among the advantages of the proposed modification of the classical forward inference process are: reducing the time necessary to search within the whole KB in order to find rules to activate and achieving additional information about fraction of rules that match the input knowledge (the set of facts). It is worth to mention that such a modification led to firing not only certain rules but also the approximate rules for which the similarity with the given set of facts is at least equal to T or is greater than 0.

5 Experiments

The main goal of the research is to show that the inference process for a KB with the partition of rules is carried out in a shorter time than the classical inference process for a KB with a list of rules. To do this, it is necessary to compare the following parameters:

[14] $sim(F, R_j) = \frac{|F \cap Profile(R_j)|}{|F \cup Profile(R_j)|}$. The value of $sim(F, Profile(R_j))$ is equal to 0 if there is no such fact f_i (or hypothesis to prove) which is included in the representative of any group R_j. It is equal to 1 if all facts (and/or hypothesis) are included in $Profile(R_j)$ of group R_j.

inference time for both inference algorithms , the size of the KB (the number of rules or groups of rules) and the $\%$ of the KB that is really analyzed. The results of the inference process on two different structures of the KB (rules and partition of rules) should not differ. Facts extracted from rules should be the same as facts extracted from the partition of rules. Checking this became an additional goal of the experiments. For a better understanding of the inference process for the partition of rules below we show a simple case study, and then the results of the experiments with some interpretation.

5.1 A Simple Case Study

To illustrate an idea of knowledge exploration from the partition of rules, let us consider an example of a KB presented in section 3.2[15]. The classical inference algorithm requires searching each of 17 rules in this set. Using the strategy, presented in section 3.1, based on similar premises only representatives of 8 created groups of rules are analysed. The more rules the greater the profit from using this approach. Usually $k << n$. The rules partitions, intended for the forward inference are represent by profiles, as follows:
$R_5 = \{r_5\}$, Profile: $\{(b,1)\}$, $R_6 = \{r_6\}$, $Profile : \{(c,1)\}$
$R_7 = \{r_7\}$, Profile: $\{(e,1)\}$, $R_8 = \{r_8\}$, $Profile : \{(f,1)\}$
$R_9 = \{r_9\}$, Profile: $\{(g,1)\}$, $R_{12} = \{r_{12}\}$, $Profile : \{(h,1)\}$
$R_{22} = \{r_4, r_3\}$, Profile: $\{(d,1)\}$, $R_{23} = \{r_{11}, r_{10}\}$, $Profile : \{(i,1)\}$
$R_{25} = \{r_{17}, r_{16}, r_{15}, r_{14}, r_{13}, r_2, r_1\}$, Profile: $\{(a,1),(j,1),(c,2),(e,2)\}$
Assuming that there is no given goal of the inference and $F = \{(a,1),(j,1),(c,2),(e,2)\}$ we need to find a group of rules that matches set F.
$sim(F, Profile(R_5)) = 0, sim(F, Profile(R_6)) = 0, sim(F, Profile(R_7)) = 0,$
$sim(F, Profile(R_8)) = 0, sim(F, Profile(R_9)) = 0, sim(F, Profile(R_{12})) = 0.$
$sim(F, Profile(R_{22})) = 0, sim(F, Profile(R_{23})) = 0, sim(F, Profile(R_{25}))$ = 1.
If there is more than one relevant group, it is necessary to choose one for further analysis. In our case, there is only one such group. It is group $R_{25} = \{r_{17}, r_{16}, r_{15}, r_{14}, r_{13}, r_2, r_1\}$.
We need to search for rules to execute within this group: $sim(F, Profile(R_{17})) = 1, sim(F, Profile(R_{16})) = 1, sim(F, Profile(R_{15})) = 1,$
$sim(F, Profile(R_{14})) = 1, sim(F, Profile(R_{13})) = 1, sim(F, Profile(R_2)) = 1.$
$sim(F, Profile(R_1)) = 1.$
Every rule can be executed because it matches the whole set of facts. It is necessary to use the so-called *conflict set strategy*[16], which allows for selecting one rule. Using the LAST_RULE strategy, rule r_{17} is chosen to execute. In effect, conclusion of rule $r_{17} : (b,1)$ is added as a new fact to $F (F = F \cup \{(b,1)\})$. Next, according to user preferences, the algorithms runs again until it executes all relevant rules, or a given number of iterations has been done or a given goal of the inference was included in the set of facts.

[15] Due to the limitation of size, the example is very simple and it is directed at presenting a conception described above, useful for improving the classical inference algorithm.

[16] There are many possible conflict set strategies: LAST_RULE, FIRST_RULE, LONGEST_RULE, SHORTEST_RULE

5.2 Results of Experiments

We are aware that experiments should be done on real KBs. The results presented in the article are derived from both real and artificial rules sets. Unfortunately, access to real large rules sets is not easy. It is worth to mention that currently we are obtaining two real KBs with over one and over four thousand of rules. Table 1 [17] presents the results of the experiments carried out on 12 different cases of KBs named $1A \ldots 4C$. Even if the number of rules is the same for some sets, they differ in sets of facts given as the input knowledge. Thus KBs are considered as different sets. We expect that during

Table 1. The results oft he experiments

KB	rule-based inference					partition of rules-based inference							
KB	1	2	3	3A	4	5	6	7	7A	8	9	10	10A
1A	10	10	10		2827	11	11	10		95,8	2939	100%	
1B	10	50	9,97	9,98	2691	11	11	16,6	12,8	50,3	2544	100%	100%
1C	10	10	9,97		2713	11	11	11,8		67	2593	100%	
2A	17	17	10		3260	25	14	10,2		236	3111	56%	
2B	17	62	20	13,33	3190	25	19	10	10,07	149	3076	76%	56%
2C	17	17	10		3147	25	9	10		98,8	3116	36%	
3A	199	199	30,4		16065	389	11	9,93		63026	15610	2, 8%	
3B	199	199	42,9	39,47	16006	389	75	61,4	31,84	59865	23024	19, 3%	8,1%
3C	199	199	45,1		15694	389	9	24,2		62078	16228	2, 3%	
4A	416	416	51,9		29431	819	13	21,7		410940	31044	1, 6%	
4B	416	416	51,8	56,73	30230	819	13	19,8	20,5	421159	44445	1, 6%	12,5%
4C	416	416	66,5		29410	819	280	20		448260	404121	34, 2%	

the next two months it will be possible to present the results of these experiments. As it can be seen in the table, in case of the classical forward inference process (rule-based) the more rules in the KB, the longer the time of inference. The reason is the necessity of matching every rule with a given set of facts. In contrast, the more rules, the faster partition-based inference. The explanation is simple. The more rules, the more advantages of using the modification of the classical forward inference algorithm in comparison to the classical deduction. The differences in times of inference on two structures of the KB: rules and the partition of rules are getting bigger when the number of rules grows up.

[17] The following information is presented in the table: the number of rules (1), the number of rules analyzed during the inference (2)[18], the time of the inference process for rules [ms] (3) and the average time of the inference [ms] (3A), the time of loading the KB for rules (4), the number of groups (5) and the number of groups analyzed during the inference (6), the time of inference process for groups of rules [ms] (7) and the average time of the inference for partition of rules [ms] (7A), the time to build the partition of rules [ms] (8), the time of loading the KB [ms] (9) as well as % of the KB searched in contrast to the classic forward inference process (10) with average % of the KB analyzed in contrast to the classic version (10A).

We have observed that the time of rules partitioning is significantly long in comparison to the time of inference, and the difference increases as the number of rules gets higher. That is why we suggest having the partition of rules stored as a static structure (assuming that it is rebuilt whenever the KB is modified). Being aware of it, we are interested only in reducing the time of the inference maximally.

6 Conclusion

The article presents an idea of a rule-based KB decomposed into groups of rules, called *partition of rules*. The KB in such a form allows for improving the efficiency of inference algorithms. In this article, only the strategy based on rules clustering is presented. However, there are many possible strategies for partitioning rules. Searching representatives of the partition of rules, instead of every single rule enables reducing the time of inference significantly. The algorithm for the forward inference on the partition of rules (proposed by the authors) has been compared to the classical inference algorithm (for rules). The results presented in Table 1 are promising. We can assume that for large rules sets (consisted of hundreds of rules or more) it is better to partition them into groups of similar rules and to carry out the inference process on representatives of such groups. The partition of rules is a very useful property which makes management of large rules sets easier and speeds up the inference process. The concept of rules' partitions is a new proposition, based on the previous research concerning rules clusters [9] and decision units [8]. Proposition of the modified inference algorithm is the continuation of the previous research on optimisation of the inference in knowledge-based systems [7]. During next works the authors plan to perform more detailed comparative experiments on real world KBs, consisted of over 4000 rules.

Mining knowledge bases is becoming an important task for many data mining applications. Thus in the future work, to reduce the time complexity of the algorithm for rules clustering and the inference on created groups of rules, the authors are going to present modifications of clustering algorithms based on incomplete knowledge.

Acknowledgements

This work is a part of the project „Exploration of rule knowledge bases" founded by the Polish National Science Centre (NCN: 2011/03/D/ST6/03027).

References

1. Toivonen H., Klemettinen M., Ronkainen P., Hatonen K., Mannila H.: Pruning and Grouping Discovered Association Rules, 1995
2. Sikora M., Gudyś A.: CHIRA—Convex Hull Based Iterative Algorithm of Rules Aggregation, Fundam. Inf.,Vol. 123 (2), p. 143–170, IOS Press, 2013.
3. Pindur R., Susmaga R., Stefanowski J.: Hyperplane aggregation of dominance decision rules, Fundam. Inf.,Vol. 61 (2), p. 117–137, IOS Press, 2004.
4. Pedrycz W., Cholewa W.: Expert Systems [in polish], Silesian University of Technology, Poland, Section of Scientific Publications, (1987)

5. Nowak-Brzezińska A., Simiński R.: Knowledge mining approach for optimization of inference processes in rule knowledge bases, Lecture Notes in Computer Science, vol. 7567, Springer Berlin Heidelberg, (534-537), (2012)
6. Akerkar R., Sajja P.: Knowledge-Based Systems. Jones & Bartlett Learning, 2010.
7. Nowak-Brzezińska A., Simiński R., Xieski T., Jach T.:Towards a practical approach to discover internal dependencies in rule-based knowledge bases, LNCS, 6954, p. 232-237, Springer Verlag, 2011.
8. Simiński R.: Extraction of Rules Dependencies for Optimization of Backward Inference Algorithm, Beyond Databases, Architectures, and Structures,Communications in Computer and Information Science, Springer International Publishing, vol. 424, p. 191-200, Springer Verlag, 2014.
9. Nowak-Brzezińska A., Wakulicz-Deja A.:The way of rules representation in composited knowledge bases, Advanced In Intelligent and Soft Computing, Man - Machine Interactions, p. 175-182, Springer-Verlag, 2009.
10. Nalepa G., Ligeza A., Kaczor K.: Overview of Knowledge Formalization with XTT2 Rules, Rule-Based Reasoning, Programming, and Applications, LNCS 6826, p. 329-336, Springer Verlag, 2011.
11. Latkowski R., Mikołajczyk M.: Data decomposition and decision rule joining for classification of data with missing values, Lecture Notes in Artificial Intelligence, LNAI 3066, p. 254-263, Springer Verlag, 2004.
12. Pawlak Z.: On Learning - A Rough Set Approach, In: Lecture Notes in Computer Science (A. Skowron, Ed.). Berlin: Springer, Vol.28, pp.197-227, 1985.
13. Skowron A., Pawlak Z., Komorowski J., Polkowski L.: Rough sets perspective on data and knowledge, in: W. Kloesgen, J. Zytkow (Eds.), Handbook of Data Mining and Knowledge Discovery, Oxford University Press, Oxford, pp. 134-149, 2002.