Investigation of the genetic algorithm possibilities for retrieving relevant cases from big data in the decision support systems

K. Serdyukov[1], T. Avdeenko[1]

[1] *Novosibirsk State Technical University, Prospekt K. Marksa 20, 630073, Novosibirsk, Russia*

Abstract

In present paper we consider the advantages and disadvantages of case-based reasoning (CBR) approach for knowledge representation of the application domain. One of the CBR shortcomings is the insufficient speed of real-time retrieval of cases, as well as the insufficient relevance of the retrieved cases to the current situation. To solve these problems, we offer the use of genetic algorithm. We propose formal statement of the genetic algorithm to the CBR retrieving stage. The results of the investigation are presented. The major advantage of the genetic algorithm is that it gives a more compact set of retrieved cases extracted which possesses, however, characteristic features of the current situation. This property can be very useful when extracting such cases from big data In conclusion, the perspectives of applying the method for adaptation of cases have been given.

Keywords: decision support systems; case-based reasoning; genetic algorithm; data mining; big data

1. Introduction

The emergence of Decision Support Systems (DSS) in the mid-1960s was associated with a model-oriented approach, popular at the time. The first DSS were limited to the types of models implemented in them, mostly deterministic, that practically did not use operative information about the circumstances in which decisions had to be made. In addition, solutions produced as a result of the operation of such systems were based on deterministic optimization models and were not always understandable and explainable from the point of view of the decision-maker, which significantly hampered their practical application.

The situation has dramatically changed since the 1990s, when DSS began to integrate first with operational databases, and then with specialized warehouses built using OLAP (On-line Analytical Processing) technology. There appeared an opportunity of operative decision-making on the basis of the objective information saved up in data warehouses. In the modern era of the Internet, when the situation in which decisions have to be made changes literally before our eyes, the DSS concept undergoes drastic changes. Information becomes so much (even the stable expression "big data" has appeared) that the data itself has ceased to be of great value. What is really valuable is knowledge, which can be extracted from the data to solve an actual problem in a particular problem domain. The task of obtaining such (informationally saturated) qualitative knowledge, and organizing them in a specially designed knowledge base, is now, in the era of large data, more relevant than ever before. It is knowledge in the form of human-understandable (cognitive) constructions, cleared of information garbage, that make it possible to organize decision support at a completely different qualitative level, when the decision-maker not only receives recommendations from the DSS, but also understands why in a particular situation it is necessary to make such a decision.

Methods of representation and organization of knowledge are traditionally developed within the Artificial Intelligence (AI) scientific discipline. In the process of development of this scientific field, two basic approaches to the declarative representation of knowledge were formed: rule-cased and case-based. The emergence, in the 1970s, of expert systems based on knowledge, is associated with the approach to the representation of knowledge in the form of rules. It is with these systems that the first real commercial successes in the field of artificial intelligence are connected. By 1992 about two thousand expert systems based on rules were implemented [1].

However, despite the success, even at the very beginning of development of systems based on rules, their shortcomings became obvious. The main problem was the problem of acquisition of knowledge from sources of information and presenting them in the form of rules. Most often, experts intuitively make decisions based on their extensive experience, without thinking, what kind of rule they apply in this or that case. Splitting the expert's specific behavior into separate blocks, called rules, is the key problem (bottleneck) in the development of rule-based systems. Another problem is the discrepancy between the real complexity of the problem domain and the very simple rule structure in the early expert systems. At present, this problem is partially solved by introducing an object-oriented description of the rule parts.

On the other hand, since the 1980s, the alternative paradigm for presenting knowledge and reasoning has attracted more and more adherents. Case based reasoning (CBR) allows solving new problems by adapting the experience of solving similar problems in the past, just as a person does in real conditions. In the paper [2] the foundations of this method are given, it is suggested to generalize knowledge about past cases and save them in the form of scenarios that can be used to develop solutions in similar situations. Later, Schank [3] continued to investigate the role played by the memory of previous situations (precedents) presented in the form of a certain knowledge container, in decision making and in the learning process.

At present, in the studies on AI, CBR is one of the key directions that is rapidly developing. The following generally accepted definition of a case could be given: "a case is a description of the problem or situation in conjunction with a detailed description of actions taken in a given situation for solving current problem". Thus, the case as a unit of knowledge includes the following:

- description of the situation;

- the decision that was made in this situation;
- the result of applying the solution.

There are various ways of presenting cases - from simple (linear representation) to more complex hierarchical representations. The case generally includes a description of the problem, as well as a solution to the problem. In case the cases from the knowledge base were used to solve specific practical problems, an additional component in the description of the case may be the result (or forecast) of the case use (positive or negative). It is interesting to note that in [4], which is often referred to as a philosophical basis of the precedent approach, it is noted that natural concepts of the application domain can often not be described by a simple linear set of properties (features), but require more complex structures for their description.

CBR-approach to the knowledge representation allowed to overcome a number of limitations inherent to the systems based on rules [5]. It does not require an explicit model for representing knowledge of the application domain, so the complex problem of knowledge acquisition is transformed into the task of accumulating cases of decision making. The implementation of the system is reduced to identifying the significant features of the case and the subsequent description of the decision-making cases in accordance with these features, which, of course, is much simpler task than building an explicit knowledge model of the application domain.

By now, the following advantages of CBR have become apparent:

• The ability to use the accumulated experience directly, without the direct involvement of a specialist who proposed a solution to a similar problem, from the case base;

• Reduction of the time for the development and making a new decision due to the available experience in solving similar problems;

• For very similar problems, the probability of making an erroneous decision is reduced;

• You can use well-developed database technology to store large volumes of cases;

• It seems promising to apply machine learning methods to the CBR-systems to the extracting knowledge in an explicit form, as well as to expanding the case base.

At the same time, there are fundamental limitations to the traditional CBR. First, when describing cases, specialists are often limited only to general knowledge or description of the problem, without deepening into the process of deriving a decision and confining themselves only to the results. Thus, the structure of the decision-making cases in this problem area does not correspond to its complexity. Secondly, as the knowledge base accumulates, the number of cases grows, which negatively affects the performance of the DSS and, accordingly, the quality of the decision made. Based on these shortcomings, it is possible to highlight the most actual requirements for the CBR-system design:

• The need for clear indexing and organization of systems for cases comparison ;

• Requirement for the selection of relevant precedents, and not just similar ones based on the closeness concept;

• Interpretability of the retrieved cases in relation to the specific problem to be solved;

• Formulation of the a solution even if there are no similar cases in the knowledge base.

Reasoning by the analogy based on CBR consists in solving the current problem in accordance with the following four steps forming the so-called CBR-cycle, or the 4R-cycle (Retrieve, Reuse, Revise, Retain) shown in fig. 1. The main stages of the CBR-cycle are:

Retrieve the most appropriate or similar case (a subset of cases) from the case base (knowledge base);

Reuse the retrieved cases to solve the current problem;

Revise (or adapt) cases, if necessary, to obtain a more specific and accurate solution;

Retain the solution in the knowledge base as a new case for its further use.

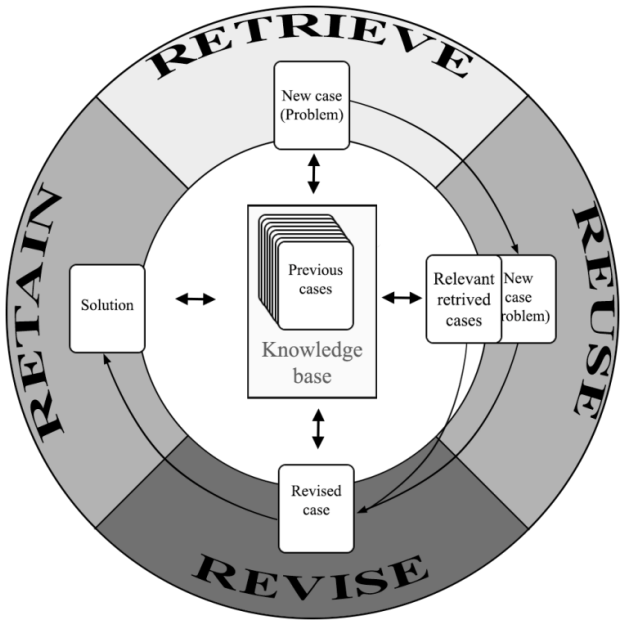

Fig. 1. CBR-cycle.

The first and most investigated stage of the CBR-cycle is to retrieve cases. The main problem in retrieving cases is a choice of the method by which the similarity measure is calculated. Often, the closest neighbor method is used for this purpose, which is based on measuring the degree of coincidence of the feature values determining the case. In papers [6-9], measures based on the introduction of the weight function, taking into account the significance of each of the features forming the case, have been proposed. However, despite numerous studies in this field, there remain problems of insufficient relevance of the retrieved cases, as well as insufficient speed of their extraction. In addition, the problem of adapting the retrieved cases to the real conditions in which decisions are made is still very far from any acceptable solution. It seems to us promising to use evolutionary approaches, in particular, the genetic algorithm, both from the point of increasing the speed of retrieving cases and from the point of view of relevance of the extracted cases to the current problem. An interesting area of research is seems the adaptation of the retrieved case to the current situation through evolutionary development.

In this paper, we consider an approach to solving the problem of retrieving and adapting cases based on the genetic algorithm. Section 2 provides a formal problem statement and scheme of the genetic algorithm for solving the problem of retrieving cases. Section 3 shows the results of research of the implemented algorithms for the two data samples. In section 4 we formulate conclusions on the work and propose perspectives for further research.

2. Problem statement in terms of the genetic algorithm

Suppose that in the DSS we have the knowledge base for decision support consisting of n cases $Case_i$, $i = \overline{1,n}$. Let we set the task of retrieving a subset of cases $Retreived = \left\{Case_{i_1}, Case_{i_2}, \dots, Case_{i_m}\right\}$, $i_k \in \{1, \dots, n\}$, that best fit the current problem $Target$, determined by a set of features $Target^j, \overline{j = 1,m}$. Note that each case $Case_i$ is determined by a set of features some of which $Case_i^j$, $\overline{j = 1,m}$, exactly corresponds to the characteristics of the target problem.

The genetic algorithm solves the problems of searching in complex decision spaces on the basis of evolutionary principles [10]. We formulate the task of retrieving cases in terms of a genetic algorithm as follows. Suppose that the population of individuals contains P chromosomes $X_p, p = \overline{1,P}$, each of which is a binary vector of the dimension n, consisting of genes encoding the presence or absence of an appropriate case $Case_{i_k}$ in a subset of the retrieved cases $Retreived$:

$$X_p = \left[X_p^1, X_p^2, \dots, X_p^n\right]^T,$$

where $X_p^i = \begin{cases} 1, \text{ if chromosome } X_p \text{ corresponds to retrieving the case, } Case_i \in Retreived \\ 0, \text{if chromosome } X_p \text{ corresponds not to retrieving the case, } Case_i \notin Retreived \end{cases}$

Thus, each chromosome corresponds to a certain subset of the retrieved cases and is characterized by a definite value of the generalized unfitness function $UF(X_p)$, that has the more value the less similar are the target problem and the subset of the retrieved cases in general:

$$UF(X_p) = \sum_{i=1}^{n} gap(Target, Case_i), \quad (1)$$

where $gap(Target, Case_i)$ is the discrepancy between the target problem and the case $Case_i$, computed as a weighted sum of discrepancies by all features

$$gap(Target, Case_i) = \sum_{j=1}^{m} w_j * \delta(Target^j, Case_i^j), \quad (2)$$

where the weights w_j define the significance of the considered features.

The values discrepancies $\delta(Target^j, Case_i^j)$ between separate features are calculated in various ways for categorical and quantitative characteristics. For categorical variables we have

$$\delta(Target^j, Case_i^j) = \begin{cases} 0, \text{if the value of the } j-\text{th feature coincides for the target problem and the case} \\ 1, \text{if the values of the } j-\text{th feature for the target problem and the case differ} \end{cases}.$$

For the numerical features we have $\delta(Target^j, Case_i^j) = \frac{\left|Target^j - Case_i^j\right|}{\max_i\left|Case_i^j\right| - \min_i\left|Case_i^j\right|}$.

In fig. 2 we present composition of one population. One population consists of a set of chromosomes, which, in turn, consist of genes. Each gene is given a value of 0 if it is unfitted or 1 if it is fitted. Accordingly, the color of the cell will also depend on the value of the unfitness function. Thus, the more retrieving cases are there in the chromosome the better it is suitable for crossing.

Fig. 2. Composition of one population.

Based on this formalization, the method sequentially implements three operations of the genetic algorithm: selection, crossover and mutation, as presented in fig. 3. The initial population is randomly generated. Further selection of individuals is performed on the basis of the unfitness function $UF(X_p)$,. The lower is the chromosome unfitness function, the higher is its

reproductive capacity. We use a single-point crossover to cross chromosomes, and we also assume a mutation probability of 0.05.

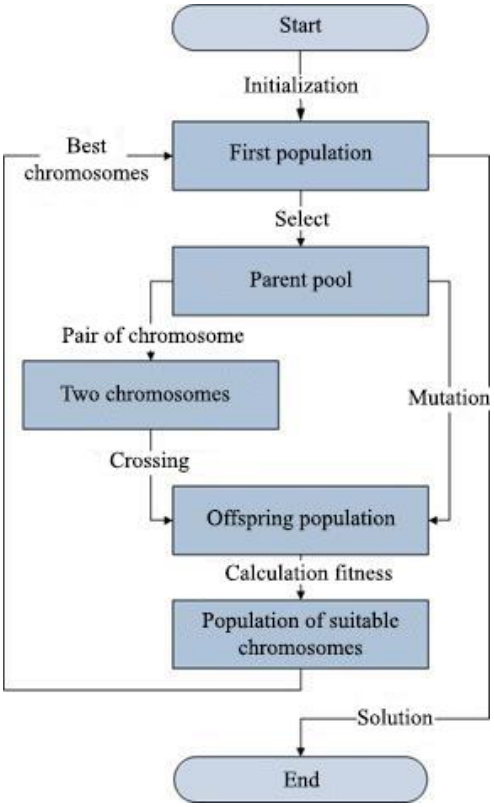

Fig. 3. Scheme of the genetic algorithm.

The next section presents the results of computations of the implemented algorithm on the test data.

3. Results of the genetic algorithm performance

3.1. Investigation of the genetic algorithm for the numerical features

In this subsection we investigate the proposed approach with the data set Iris (available at http://archive.ics.uci.edu/ml/) with 150 cases, characterized by 4 numerical features ($n = 4$), which are classified into 3 classes (Iris Setosa, Iris Versicolour or Iris Virginica) through the classifying attribute. The features are as follows:

- Length of sepals;
- Width of sepals;
- Length of petals;
- Width of petals.

In our test we are interested in the retrieving cases similar to the request: length of the sepals 6.3 cm, width of the sepals 2.3 cm, length of the petals 4.4 cm, width of petals 1.3 cm. The flower with these parameters refers to the species Iris Versicolour. The genetic algorithm settings are as follows: population is 100 chromosomes, 10 generations, 50% crossover position and 5% mutation chance.

In table 1 we give the results obtained for three variants that differ from one another in the flexibility of the query. In the first test, we are interested in the exact match of the features of the request (current situation) and the retrieved cases (0% discrepancy). For numeric attributes, the probability of the exact coincidence of all four features is very small, so we get a small number of retrieved cases, but also a short computation time. In the second test, we allow 10% deviation of the features of the retrieved cases from the query, and in the second test, we allow 20% such deviation. In this case, we obtain a consistent increase in the number of retrieved cases, and accordingly increase of the computation time.

As for the quality of the retrieving, we can judge it by the quality of the classification of the retrieved cases. As you can see, the correct classification takes place in 83% of cases for 20% and 10% feature deviations, i.e. allowing a flexible query, we support the representativeness of the retrieved sample. If we set 0% feature deviation, the accuracy of the classification of the retrieved cases is reduced.

3.2. Investigation of the genetic algorithm for big data

In this test we use Adult database from UCI [15]. The case base contains 32561 cases. Database contains the following features:

- Age (numeric feature);
- Workclass (categorical feature);
- Fnlwgt – final weight (numeric feature);
- Education – last education (categorical feature);
- Education-num – number of different education types (numeric feature);
- Marital-status (categorical feature);
- Occupation (categorical feature);
- Relationship (categorical feature);
- Race – ethnic group (categorical feature);
- Sex (categorical feature);
- Capital-gain – incomings (numeric feature);
- Capital-loss – expenses (numeric feature);
- Hours-per-week (numeric feature);
- Native-country (categorical feature);
- Annual income (numeric feature).

Table 1. Investigation of accuracy and speed of the genetic algorithm for Iris data.

No	Results of deviation								
	20% feature deviations			10% feature deviations			0% without deviation (exact)		
	Time (s)	The number	Class	Time (s)	The number	Class	Time (s)	The number	Class
1	0.56	3	Iris Versicolour	0.35	1	Iris Versicolour	0.05	1	Iris Versicolour
2	0.31	2	uncertainty	0.15	2	Uncertainty	0.09	1	Iris Versicolour
3	0.48	3	Iris Versicolour	0.23	1	Iris Versicolour	0.12	1	Iris Setosa
4	0.54	3	Iris Versicolour	0.1	1	Iris Versicolour	0.1	2	Iris Versicolour
5	0.12	1	Iris Versicolour	0.12	1	Iris Versicolour	0.12	2	uncertainty
6	0.22	1	Iris Versicolour	0.09	1	Iris Versicolour	0.09	1	Iris Versicolour
7	0.37	1	Iris Versicolour	0.17	2	Iris Versicolour	0.05	2	H/H
8	0.38	1	Iris Versicolour	0.26	1	Iris Versicolour	0.07	1	Iris Versicolour
9	0.43	1	Iris Versicolour	0.12	1	Iris Virginica	0.11	1	Iris Versicolour
10	0.31	3	uncertainty	0.13	1	Iris Versicolour	0.09	1	Iris Setosa
Av	0.372	1.9	Iris Versicolour (~83%)	0.172	1.2	Iris Versicolour (~83%)	0.089	1.3	Iris Versicolour (~70%)

We use only two of 14 features in the request (current situation) - numerical feature Age (equal to 30 years) and categorical feature Education (bachelor). As the classifying attribute we use annual income with two classes of more than and less than $50 000 (<=50K or >50K).

In the first test we carried out the research similar to those made with Iris dataset. The genetic algorithm settings are as follows: population is 100 chromosomes, 10 generations, 50% crossover position and 5% mutation chance. The results are given in table 2 and are similar to those obtained in section 3.1.

Table 2. Investigation of accuracy and speed of the genetic algorithm for UCI data.

No	Results of deviation								
	20% feature deviations			10% feature deviations			0% without deviation (exact)		
	Time (s)	The number	Class	Time (s)	The number	Result	Time (s)	The number	Class
1	136	1849	<=50K	127	1253	<=50K	101	89	<=50K
2	121	1807	<=50K	114	1224	<=50K	95	85	<=50K
3	116	1858	<=50K	120	1226	<=50K	94	89	<=50K
4	115	1814	<=50K	138	1237	<=50K	102	90	<=50K
5	116	1803	<=50K	112	1212	<=50K	94	82	<=50K
6	137	1780	<=50K	134	1250	<=50K	89	86	<=50K
7	134	1857	<=50K	126	1261	<=50K	100	74	<=50K
8	135	1805	<=50K	125	1224	<=50K	89	100	<=50K
9	151	1804	<=50K	115	1227	<=50K	89	90	<=50K
10	146	1820	<=50K	108	1241	<=50K	97	85	<=50K
	130.7	1819.7	<=50K (~65%)	121.9	1235.5	<=50K (~70%)	95	87	<=50K(~63%)

In the second test we compare genetic algorithm with conventional CBR. In the table 3 we give computation time, the number of retrieved cases and accuracy of classification for conventional CBR, and the genetic algorithm with 1,2 and 5 generations. The genetic algorithm settings are as follows: population is 10 chromosomes, 50% crossover position, 5% mutation chance and 5% numerical feature (age) deviation. As a result one can see that for the genetic algorithm we have obtained less amount of the retrieved cases with the same classification accuracy and insignificant increase in the computation time. This indicates a greater representativeness of a set of the retrieved cases when using the evolutionary approach for retrieving.

Table 3. Comparison of genetic algorithm with conventional CBR.

No	Conventional CBR			GA with 1 generation			GA with 2 generations			GA with 5 generations		
	Comp. time, s	The numb.	Accu-racy	Comp. time, s	The numb.	Accu-racy	Comp. time,s	The numb.	Accu-racy	Comp. time,s	The numb.	Accu-racy
1	11	3534	66%	14	1767	66%	12	1768	66%	29	1726	66%
2	11	3534	66%	13	1767	66%	12	1727	66%	26	1769	65%
3	11	3534	66%	8	1757	68%	16	1793	67%	22	1792	66%
4	11	3534	66%	10	1747	65%	17	1755	65%	23	1778	65%
5	11	3534	66%	9	1761	66%	17	1816	66%	20	1760	67%
	11	3534	66%	10.8	1759.8	66%	14.8	1771.8	66%	24	1770.4	66%

4. Conclusion

Thus, we considered the basic stages of reasoning by analogy, and also the peculiarities of the CBR-cycle. We proposed formal statement of the genetic algorithm for the problem of retrieving a subset of cases relevant to the problem solved. The results of the conducted research on the test data were presented that testify to good prospects for using the genetic algorithm not only in the retrieving stage but also in the adaptation stage of the CBR-cycle. It seems promising to integrate the genetic algorithm with the generation of fuzzy rules [12] to implement the adaptation of retrieved cases to the problem being solved.

Acknowledgements

The reported study was funded by Russian Ministry of Education and Science, according to the research project No. 2.2327.2017/PCh.

References

[1] DTI. Knowledge-based systems survey of UK applications. Department of Trade & Industry UK, 1992.
[2] Schank RC, Abelson RP. Scripts, Plans, Goals and Understanding. Erlbau, 1977.
[3] Schank RC. Dynamic Memory: A theory of reminding and learning in computers and people. Cambridge University Press, 1982.
[4] Wittgenstein L. Philosophical Investigations. Blackwell, 1953.
[5] Watson I, Marir F. Case-based reasoning: A review. The Knowledge Engineering Review 1994; 9(4): 327–354.
[6] Bonzano P, Cunningham P, Smith B. Using introspective learning to improve retrieval in CBR: A case study in air traffic control. Proc. 2nd Int. Conf. Case-based Reasoning, 1997; 291–302.
[7] Cercone N, An A, Chan C. Rule-induction and case-based reasoning: Hybrid architectures appear advantageous. IEEE Trans. Knowledge and Data Engineering 1999; 11: 166–174.
[8] Coyle L, Cunningham P. Improving recommendation ranking by learning personal feature weights. Proc. 7th European Conference on Case-Based Reasoning, 2004; 560–572.
[9] Jarmulak J, Craw S, Rowe R. Genetic algorithms to optimize CBR retrieval. Proc. European Workshop on Case-Based Reasoning (EWCBR 2000), 2000; 136–147.
[10] Yang HL, Wang CS. Two stages of case-based reasoning - Integrating genetic algorithm with data mining mechanism. Expert Systems with Applications 2008; 35: 262–272.
[11] Adult Data Set. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml/datasets/Adult.
[12] Avdeenko TV, Makarova ES. Integration of case-based and rule-based reasoning through fuzzy inference in decision support systems. Procedia Computer Science 2017; 103: 447–453.