Research Article

A Competence-based Deletion Model for the Improvement of Case-based Maintenance in Case-based Reasoning

Nay Lin Aung* and Adtha Lawanna
Department of Information Technology, Vincent Marry School of Science and Technology, Assumption University, Bangkok, Thailand

* Corresponding author. E-mail: nl.aung8@gmail.com DOI: 10.14416/j.asep.2019.06.002
Received: 3 October 2018; Revised: 19 March 2019; Accepted: 16 May 2019; Published online: 27 June 2019

Abstract
Case-based reasoning is a methodology of reasoning used to fix new problems in previous experiences. It applies artificial intelligence, machine learning, systems based on knowledge and other associated fields due to both its extensive usage by humans and its appeal as a methodology for building intelligent systems. This paper focuses on two core problems - increasing case base size and decreasing the competency of the case base. These problems arise due to the repetition of the cycles of case-based reasoning. In order to solve these problems, case-based maintenance methods are crucial for case-based reasoning. In this paper, a useful model for case-based maintenance is proposed in order to deliver better outcomes in comparison with random deletion, utility deletion, and footprint utility deletion. In order to study the efficiency of each method, seven datasets from the machine learning repository are applied to each algorithm. The results of the proposed model show that not only is storage size lower but also competence is higher in comparison with other methods after reduction. Moreover, the reduction rate is higher, and problem solving related to performance is significantly better than when other methods are used.

Keywords: Cased-based reasoning, Case-based maintenance, Competence-based deletion, Artificial intelligence, Machine learning

1 Introduction

Case-based reasoning (CBR) is a technique of machine learning that takes advantage of old knowledge in order to understand and solve new issues. Similar issues have similar solutions to CBR's basic assumptions. A reasoner recalls a previous experience that is identical to the original problem in order to solve it by using the former case. The methodology of CBR is similar to the human method of thinking and reasoning. When humans have a problem, they try to find a solution to solve it based on their previous experiences. If there are no similar situations in the past, they will figure out a new solution in order to fix the new problem they encounter.

CBR systems are usually large - scale case bases, so maintaining the case base quality, skill, and efficacy is essential. Otherwise, CBR systems are very likely to experience a utility issue that occurs when knowledge is absorbed instead in an attempt to boost the performance degradation of a system. One of the major problems of the utility issue is the slowdown of conventional memories as the number of cases stored increases [1], [2]. Consequently, the occurrence of system performance-associated issues will increase, such as case preparation and processing times. If these issues are extended, the performance of the system will decrease. Therefore, the importance of case-based maintenance (CBM)

Please cite this article as: N. L. Aung and A. Lawanna, "A competence-based deletion model for the improvement of case-based maintenance in case-based reasoning," *Applied Science and Engineering Progress*, vol. 14, no. 1, pp. 3–12, Jan.–Mar. 2021, doi: 10.14416/j.asep.2019.06.002.

has been increasingly recognized and addressed. Different CBM strategies can be applied in undertaking the maintenance of CBR, such as deleting out-dated, redundant and/or inconsistent cases, adding important cases, and revising existing cases [3].

One of the oldest traditional deletion policies is random deletion, which can eliminate a random case from case bases once the case base surpasses certain predefined obstacles [4]. However, this deletion policy no longer provides convincing or pleasant effects with regard to optimizing the case base size, and reserving competence is difficult while removing high case utility value [4]–[6]. Another method called ironically deletion can simply decrease cases based on the recurrence of each retrieved case [7], [8] but the delivery of satisfactory outcomes and the performance of the case base may not be guaranteed. The main reason is that the critical cases that are not frequently used may be deleted. For this reason, Minton [8] proposed utility deletion (UD) based primarily on its utility metric, which selects a case item to delete by measuring its performance benefits. Nonetheless, the system's competence will still drop and will show itself to be a trade-off between the solution quality correlated with big case bases and the efficiency issue of operating with a massive case base [9]. Therefore, footprint deletion was proposed by Smyth and Keane [6] to exclude irrelevant cases leading to the optimum configuration of the case base [9]. Low-utility cases can be processed in their method even though high utility cases with some competence additions can be erased [6]. In brief, for case-based reasoning, these traditional deletion policies could have negative outcomes. Deleting important cases can significantly downgrade a CBR system's competence, making specific classes of target issues lastingly unsolvable [5].

In this regard, this paper enhances the process's ability to maintain the system by decreasing cases that have the least impact on the case base's competence and performance. The anticipated result is to develop a proposed effective case reduction model while improving CBR performance and competency. The purpose of focusing on improving performance and competence is to reduce negative impact on the four processes' performance in CBR due to the increasing case base size. Therefore, this can harm overall CBR performance in the process of retrieving, reusing, revising and retaining cases, particularly in term of increasing time. In this paper, a competence-based deletion algorithm including four processes is introduced. The performance and competence of the proposed model are evaluated by using seven datasets.

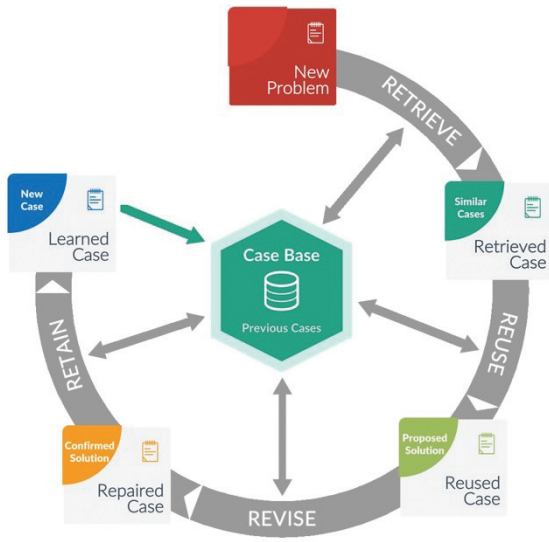

Figure 1: The CBR cycle [5], [12], [14], [15].

2 Material and Methods

2.1 *Case-based reasoning (CBR)*

Case-based reasoning [10] is a trouble-fixing paradigm that solves new issues by retrieving and reusing similar previous cases from the case base. If with previous solutions new problems cannot be solved, the retrieved solutions will be adapted (using domain knowledge) to be applicable to the new issue; this will retain new solutions (cases) and update the case base [11]–[14]. There are two main components in a case: a problem that describes the context of the environments, and the solution to the problem [7], [12].

According to Aamodt and Plaza, there are four processes in CBR (Figure 1) [5], [14], [15]:

1. RETRIEVE: the case-based reasoner explores the most similar cases that can be matched with the current problem in the database;

2. REUSE: solving the current problem by using the retrieved case;

3. REVISE: the suggested solution is revised when needed;

4. RETAIN: the new solution is retained as a new case

2.2 *Case-based maintenance (CBM)*

Because the system grows, the case base size will increase and the occurrence of system performance-associated issues will raise; for example, the time taken for case preparation and processing increases as system performance decreases. Therefore, the case base in the system must be maintained.

According to Leake and Wilson [16], [17], case-based maintenance is a case-based refinement process in the case-based reasoning system for increasing the performance of the system. Policies are implemented in CBM in order to revise the case base, aiming at reducing case research time using different strategies and ensuring better system processing time by facilitating new reasoning for specific performance purposes. Two main types of strategies used in CBM are: 1) case deletion, which decreases the case base size by dropping unnecessary cases and 2) case addition, which adds necessary cases to determine the lower bounds and upper bounds concerning the entire case [16].

2.3 *Random deletion (θ_1)*

Cases are gradually increased in the case base over a period of time, finally reaching a predefined limit to the case base size. Therefore, the cases from case base must be deleted. Random deletion is a process where a case is randomly excluded from the case base once the setup limit is surpassed by the case base size [4]. This refers to a simple deletion method that operates effectively with more principled and costly methods, and it might be suitable for the system that has an enormous size [6], [4], [18]. However, it does not deliver results that are credible and fulfilling while optimizing the case base size. The remaining cases may not ensure the case base performance for the next cycle after optimization [5]. The problem with this approach is that it is possible to delete critical cases by mistake [13].

2.4 *Utility deletion (θ_2)*

Utility deletion is introduced in order to determine the remaining issues that the prior studies such as random deletion cannot answer. Predefining the case base size using this deletion method is unnecessary. The case base size in utility deletion is very much relied on the frequency of the utilization of each case. If the case base size increases, a specific case's application frequency tends to drop, thus degrading its utility estimate [6]. Utility deletion (UD) selects a deletion case based on the utility metric of Minton. Negative-utility value cases are deleted [17], [19]. In utility deletion, the quality of the problem solution progresses with the case base size [9], [20]. However, the utility issue appears as a trade-off between the quality of the solution correlated with comprehensive case bases and the performance issue of operating with a massive case base [9].

2.5 *Footprint utility deletion (θ_3)*

There are several traditional deletion methods that can cause the system competency and the performance degradation after deletion. In order to fix this issue, Smyth and Keane [1] introduced footprint utility deletion (FUD), which is a mixed-method that combines utility deletion and footprint deletion. Cases are selected by applying the footprint utility to delete. The principle is that using case coverage and reachability according to its capacity defines each case. The case with the lowest coverage or largest reachability set is selected to delete [6], [7], [9]. This approach provides four case categories: pivotal, support, spanning, and auxiliary. Auxiliary cases are the least significant and not a primary benefaction to competence. Thus, those are removed firstly. Then, the support cases, the spanning cases, and the pivotal cases are sequentially removed [4], [6].

2.6 *The proposed model (θ_4)*

The proposed model aims to develop an effective model of CBM in order to preserve the case base's competence while optimizing its size and enhancing its performance. This model includes four processes. First, the case base is cleaned by finding complicated cases. Second, cases are treated with an algorithm in order to find redundant cases to delete. Third, cases are sorted by frequency of utilization, and the least-used cases are eliminated. Finally, cases are determined by a competence metric associated with coverage and

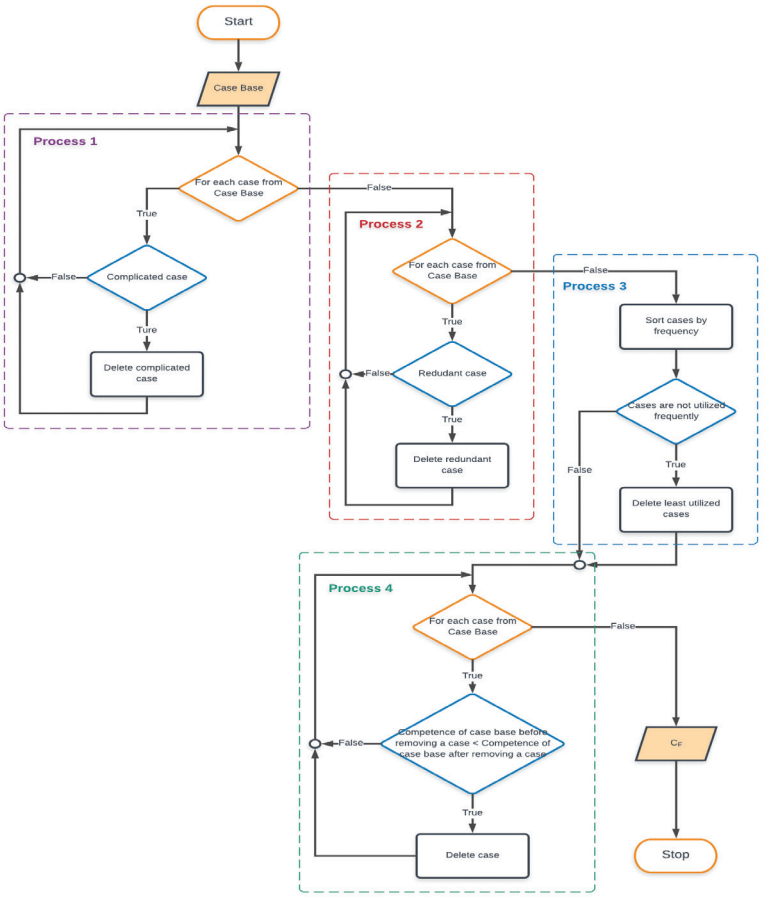

Figure 2: Proposed conceptual model diagram.

reachability in order to decide whether those cases can be deleted without affecting the competency of the case base.

The conceptual diagram of the proposed model is presented below (Figure 2).

The proposed model includes four processes and the process details are explained in the succeeding sections.

- Process 1: Determining complicated cases to delete
- Process 2: Determining redundant cases to delete
- Process 3: Determining the least utilized cases to delete
- Process 4: Determining cases that do not affect to the competence of the case base by deleting them

The deletion flow of the proposed model is presented in Figure 3. First, complicated cases are removed from the case base. Second, redundant cases are deleted from the remaining cases from process 1. Third, the least-utilized cases are determined and eliminated from the remaining cases from process 2. Last, the competencies of all remaining cases from process four are calculated in order to compare the competence value from before and after the deletion of cases. All cases not affected by the case base competence are deleted.

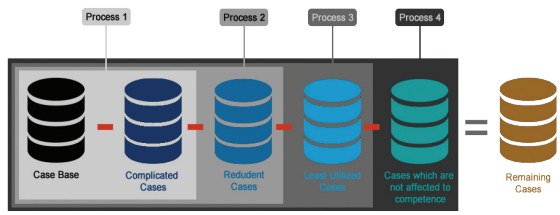

Figure 3: Deletion flow of proposed model.

The following Table 1 shows the description of notations used in this paper.

Table 1: Description of notations

Notation	Meaning
C	A Case Base
$\|C\|$	Cardinality of C
c	A case
t	A set of uncomplicated cases
t'	A set of complicated cases
A_i	An attribute of each case
n	Number
r	A set of non-redundant cases
r'	A set of redundant cases
$c.A_{mi}$	An attribute of a case
A_{mj}	An attribute from another case
u_i, u_n, u_{n+1},..	Utilization percentage range
$CM(c)$	Competence measure of a case
$Vc(c)$	Coverage value of a case
$Vr(c)$	Reachability of a case
C_F	Remaining case base
$\|C_F\|$	Cardinality of C_F
$C_{(comp)}$	Competence of initial case base
$CF_{(comp)}$	Competence of remaining case base
S	Storage size
SVP	Problem solving
P	Number of target problems
P_0	Number of problems that cannot be solved
R	Reduction rate

2.6.1 Process 1: Determining complicated cases to delete

In a case, there are requested problems that can be called attributes. The attributes can vary based on the dataset. If more than one attribute has no value, it is called a complicated case, and the problem cannot be solved as the information is not sufficient. The definition of a complicated case and an uncomplicated case are explained as below.

$C = \{c_1, c_2, c_3, ..., c_n\}$

$c_i = \{c_i.A_{m1}, ci.A_{m2}, ..., ci.A_{mi}\}$

The complicated case can be determined by using Equation (1).

$$t' = \{\{c_1.A_{m1}, null, c_1.A_{m3}, \dots, c_1.A_{mn}\}, \{null, c_2.A_{m2}, \dots, c_2.A_{mn}\}, \dots, \{c_n.A_{m1}, null, \dots, c_n.A_{mn}\}\} \quad (1)$$

where t' is a set of complicated cases, A_{mi} is an attribute of each case, null is the requested problem that may or may not be solved, and n is a number.

The uncomplicated can be defined using Equation (2).

$$t = \{\{c_1.A_{m1}, c_1.A_{m2}, \dots, c_1.A_{mn}\}, \{c_2.A_{m1}, c_2.A_{m2}, \dots, c_2.A_{mn}\}, \dots, \{c_n.A_{m1}, c_nA_{m2}, \dots, c_n.A_{mn}\}\} \quad (2)$$

where t is a set of uncomplicated case.

Algorithm: Process 1

Input: C
Output: $t = C - t'$
Receive cases from C
For each c in C
  If $Comp(c) == True$ **then**
    Remove c
  Else
    Keep c
  EndIf
EndFor
Function *Comp*(c) {
  num = len(c) // number of attribute
  i = 0
  While (count<i) {
    If c[A_i] == ∅// any attribute of c is missing
      return True
    i++
  }
  return False
}

In the algorithm, when cases are received, each case is extracted and checked for whether that case is a complicated one, such as t'. If the case is complicated case, it is excluded from the case base. Only cases such as t will be kept in the case base. This process will proceed until there are no more complicated cases in the case base. The reason for deleting those cases is because they are requested problems that do not have enough information and therefore can affect the case base performance.

2.6.2 Process 2: Determining redundant cases to delete

A redundant case is denoted as two cases having the same values, or for all intents and purposes they are equivalent. Cases can be redundant because of the

ever-evolving nature of case bases. Hence it is crucial to have a mechanism to decide if the case is subsumed by different cases within the case base. If two or more cases in a case base are very similar, cases are retrieved as the equal set of queries when the reasoner requests a case to solve a problem. It is unnecessary to preserve each case within the case base and as this will degrade the retrieval efficiency of the case base.

A redundant case is defined as in Equation (3). A redundant case is a case that is similar to the attributes of another case.

$$r' = A_{mi} \cap A_{mj} \neq \varnothing \quad (3)$$

where r' is a set of redundant cases, A_{mi} is an attribute of a case and A_{mj} is an attribute of another case.

A non-redundant case is defined as in Equation (4). The redundant case is a case that is not similar to the attributes of another case.

$$r = A_{mi} \cap A_{mj} = \varnothing \quad (4)$$

where r is a set of non-redundant cases.

Algorithm: Process 2
Input: t
Output: $r = t - r'$
Receive cases from t
For each c_t in t
  If $c_t[A_{mi}] \cap c[A_{mj}] = \varnothing$ **then**
    Remove c_t
  Else
    Keep c_t
  EndIf
EndFor

When the algorithm receives cases from t where the complicated cases have been deleted in process 1, every single case from t is extracted and compared with other cases from the case base. If the algorithm finds that a case is similar to other cases, that case will be eliminated from t.

2.6.3 Process 3: Determining least-utilized cases to delete

The utilized case is defined as a case that has been previously used to solve the problems in the case base. Because of changes in the environment, some cases in the CBR system can turn out to be out-dated or not get used frequently (or at all), and for that reason the CBR needs to do maintenance, and less utilized cases have to be deleted in order to improve the CBR performance [17]. This process measures the frequency of each case that has been utilized previous and removes the cases that have not been used frequently from the case base.

Algorithm: Process 3
Utilization range:
81–100% = u_i
61–80% = u_n
41–60% = u_{n+1}
21–40% = u_{n+2}
0–20% = un_{+3}
Input: r
Output: $u_i = r - (u_{n+3} + u_{n+2} + u_{n+1} + u_n)$
Receive cases from r
Sort cases by utilization percentage
If $c == u_{n+3} \,||\, c == u_{n+2} \,||\, c == u_{n+1} \,||\, c == u_n$ **then**
  Remove cases // $u_{n+3} + u_{n+2} + u_{n+1} + u_n$
EndIf

where u_i is a set of cases between 81–100%, u_n is a set of cases between 61–80%, u_{n+1} is a set of cases between 41–60%, u_{n+2} is a set of cases between 21–40% and u_{n+3} is a set of cases between 0–20%.

All cases received from r are sorted by the utilization percentage, which is the quantity of times that the case was used. After sorting the cases, cases that have low utility are deleted from r.

2.6.4 Process 4: Determining cases that do not affect the competence of the case base by deleting them

Competence is the capability of problem solving after reduction of the case base. On the other hand, the range of new problems can be solved successfully. In this process, the competence values of the case base are compared before and after in order to determine whether a case has to be deleted or kept. If the removal of a case does not affect the case base's competence, that case will be eliminated from the case base. Otherwise, it will not be deleted.

Case base competence can be computed by the total amount of the measure of competence (CM) of each case divided by the total number of cases in the case base [Equation (5)] [1]. Competence is based on the

two concepts of coverage and accessibility. Coverage is the size of a target problem set that a particular case can solve, and reachability is the size of the case base that can deliver solutions to the target problem [6].

$$Competence\ of\ case\ base = \frac{\sum_{i=1}^{n} CM(c_i)}{n} \quad (5)$$

$$CM(c) = \frac{V_c(c)}{V_r(c)} \quad (6)$$

where c_i is a case from the case base, n is the number cases in the case base, $CM(c)$ is the competence measure of a case, $V_c(c)$ is the coverage value of a case and $V_r(c)$ is the reachability of a case.

Algorithm: Process 4
Input: u_i
Output: $C_F = u_i - c'$
Receive cases from u_i
For each c in u_i
  Calculate competence of u_i *//Before*
  Recalculate competence of u_i after removing a case *//After*
  If *competence value from after* < *competence value from before* **then**
    Keep c
  Else
    Delete c'
  EndIf
EndFor

where c' is a case that does not affect to the competence of the case base by deleting it and C_F is a case base in which the remaining cases are left after conducting all of the processes.

In the algorithm, the competence of the whole case base is calculated first. Then, a case from the case base is deleted. After deletion, the competence of the case base is calculated again. After that, the case is decided as to whether it has to keep or delete by comparing competence value from before and after. If the competence value from after the deletion is less than competence value from before the deletion, it means that the competence is decreased and the case cannot be deleted. Otherwise, the case is deleted from the case base. This looping process continues until all of the cases from the case base are completed.

2.7 *Evaluation*

An effective case base is capable of providing as many solutions as possible efficiently and accurately. Competence and performance are critical criteria to the evaluation of a case base [6], [9].

2.7.1 Competence

Competence is the capability of problem solving after reduction of the case base. On the other hand, it is the range of target problem that can be successfully solved. A case base that has high competence can be called a good case base [19]. Competence is based on the two concepts of "coverage" and "reachability" [1]. Competence can be calculated using the covering set cardinal ratio and the reachability set cardinal ratio [1].

The competence of each case can be calculated using the competence measure (CM) in Equation (6). In order to calculate the competence of the entire case base, Equation (5) is used [Equation (7)].

$$Competence\ (\%) = \left(\frac{C_{F(comp)}}{C_{(comp)}}\right) \times 100 \quad (7)$$

where $C_{(comp)}$ is the competence of the initial case base and $C_{F(comp)}$ is the competence of the remaining case base.

2.7.2 Performance

The improvement of the storage size and accuracy of the problem solving are important criteria to judge the CBM methods' performance [5], [21].

The storage size is the extent to which methods store the cases after deletion [Equation (8)].

$$S\ (\%) = \left(\frac{|C_F|}{|C|}\right) \times 100 \quad (8)$$

The capacity of the fixing issues is the standard for assessing the CBM's overall performance. The following Equation (9) can be used to obtain the problem solving or % *SVP*.

$$SVP\ (\%) = \left(\frac{P - P_0}{P}\right) \times 100 \quad (9)$$

where P is the number of targeted problems and P_0 is the number of problems that cannot be solved.

2.7.3 Reduction rate

The percentage of reduction shows the size of the reduction rate. It calculates how much policies are able to decrease the original case base [Equation (10)].

$$R\,(\%) = \left(\frac{|C| - |C_F|}{|C|}\right) \times 100 \tag{10}$$

3 Results and Discussion

In this section, the effectiveness of some of the CBM deletion methods together with the proposed model is presented. The purpose of the deletion methods is to decrease the size of the case base while preserving the performance and competence of the system as much as possible.

Seven datasets acquired from the UCI repository were used to evaluate the performance and competence of the methods. The seven datasets were Iris with a size of 150 cases, *E.coli* with a size of 336 cases, Breast-W with a size of 699 cases, Ionosphere with a size of 351 cases, Stalog with a size of 946 cases, Yeast with a size of 148 cases and Annealing with a size of 789 cases. Details on these databases are presented in Table 2.

Table 2: Description of datasets

Dataset	Ref.	Cases	Attributes
Iris	α_1	150	4
E.coli	α_2	336	8
Breast-W	α_3	699	10
Ionosphere	α_4	351	34
Stalog	α_5	946	18
Yeast	α_6	148	8
Annealing	α_7	789	38

In this experimentation, each dataset was applied to the algorithms and the results of each algorithm were compared. The experiments showed that the competence of the proposed model (θ_4), shown in Figure 4, was significantly higher than the competence of other methods. According to Figure 4, the competency range of θ_1 was 63–78%, for θ_2 it was 81–89%, for θ_3 it was 91–94% and for θ_4 it was 97–100%. This indicates

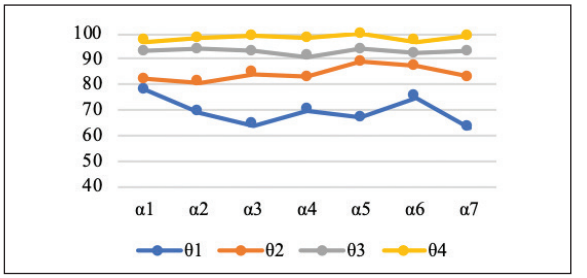

Figure 4: Comparison of four methods in term of percentage of competence.

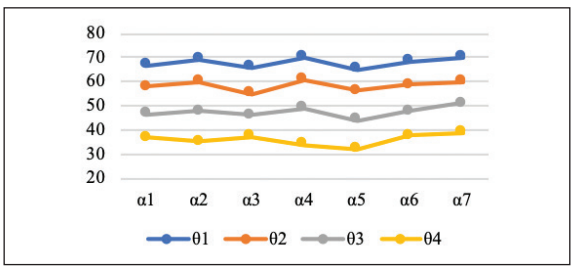

Figure 5: Comparison of four methods in term of percentage of storage size.

that the proposed model (θ_4) keeps its competence level close to 100% for all datasets. Among the seven datasets, the Stalog dataset (α_5) had the highest number of cases. It is seen that the competence of the proposed model keeps 100% in the dataset that has a large case base size. Moreover, other competence-based deletion method such as the FUD retained the level of 91% as well. However, the traditional deletion method such as random deletion does not maintain good competence level, though the UD still retains its competence over 80%. Hence the proposed model maintains the highest competency level with the larger case base size. The overall competence of the proposed model is higher than the other methods in each dataset.

In order to compare the performance of the algorithms, the storage size and the problem solving are two important criteria to be measured. According to Figure 5, it can be clearly seen that the proposed model delivers the least storage size in comparison with the others while optimizing a better competence level in the seven datasets. In the Ionosphere dataset (α_4), the result of the storage size percentage produced by the proposed method was 34% which is sensibly less than the results of the other three methods, and just a half of random deletion storage size.

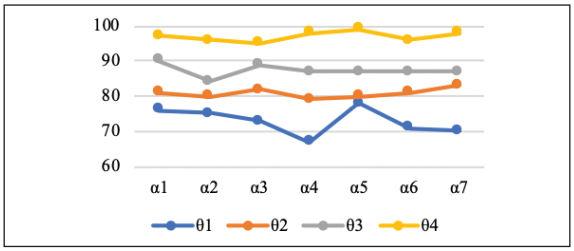

Figure 6: Comparison of four methods in term of percentage of problem solving.

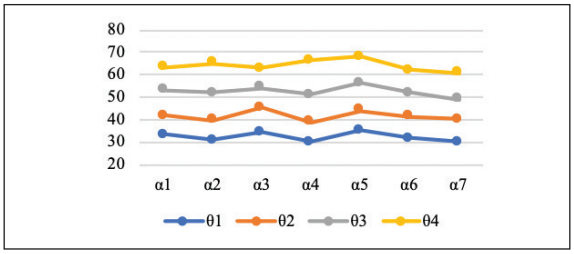

Figure 7: Comparison of four methods in term of percentage of reduction.

The results of problem-solving comparative studies are reported in Figure 6. According to that figure, the problem-solving range of θ_1 was from 67% to 78%, for θ_2 it was from 79% to 83%, for θ_3 it was from 84% to 90% and for θ_4 it was from 95% to 99%. Hence the proposed model was seen to have the greatest problem-solving rate in all datasets compared to other methods.

The interpretation of Figure 7 indicates that the reduction rate of the proposed model (θ_4) is the highest among the four algorithms. θ_1 shows a 30–35% reduction, θ_2 gives a 39–45% reduction, θ_3 indicates a 49–56% reduction and θ_4 gives a 61–68% reduction. It is observed that the reduction rate of the proposed model in the Annealing dataset (α_7) was the lowest. However, it is still 10% higher than the highest rate among the other three algorithms.

After assessing the four algorithms, we can see in each figure that the capabilities of the introduced model are the best in providing the smallest storage size, the highest reduction rate, and the highest problem-solving rate while optimizing a good competence level.

4 Conclusions

This research introduces an effective model in case-based reasoning for improving case-based maintenance. It aims to solve the problem of competence and the percentage of a decrease in problem solving while reducing cases from the case base. In order to research the performance of the proposed model, three other algorithms such as random deletion, utility deletion, and footprint utility deletion were used to perform the comparative studies using seven databases acquired from the UCI repository.

An experimental evaluation for proposed model outperformed comparable to other models in all aspects. While the proposed model made a huge reduction of the case base compared to other models, its competence and problem solving were still the highest in the outcomes. Hence, its performance was better than that of the other models.

References

[1] A. Smiti and Z. Elouedi, "WCOID-DG: An approach for case base maintenance based on weighting, clustering, outliers, internal detection and dbsan-gmeans," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 27–38, Feb. 2014.

[2] M. Salamó and M. López-Sánchez, "Rough set based approaches to feature selection for case-based reasoning classifiers," *Pattern Recognition Letters*, vol. 32, no. 2, pp. 280–292, Jan. 2011.

[3] S. Ferrandiz and M. Boullé, "Bayesian instance selection for the nearest neighbor rule," *Machine Learning*, vol. 81, no. 3, pp. 229–256, May. 2010.

[4] S. Markovitch and P. D. Scott, "The role of forgetting in learning," in *Proceedings of the Fifth International Conference on Machine Learning*, 1988, pp. 459–465.

[5] A. Smiti and Z. Elouedi, "Overview of maintenance for case-based reasoning systems," *International Journal of Computer Applications*, vol. 32, no. 2, pp. 49–56, Oct. 2011.

[6] B. Smyth and M. T. Keane, "Remembering to forget a competence-preserving case deletion policy for case-based reasoning systems," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, Aug. 1995, pp. 377–383.

[7] B. Morello, M. K. Haouchine, and N. Zerhouni, "Case-based maintenance: Structuring and

incrementing the case," *Knowledge-Based System*, vol. 88, pp. 165–183, Nov. 2015.

[8] S. N. Minton, "Quantitative results concerning the utility of explanation-based learning," *Artificial Intelligence*, vol. 42, no. 2–3, pp. 363–391, Mar. 1990.

[9] M. Haouchine, B. Chebel-morello, and N. Zerhouni, "Competence-preserving case-deletion strategy for case-base maintenance," in *Proceedings Martin Schaaf. ECCBR'08*, Trier, Germany, Sep. 2008, pp.171–184.

[10] J. L. Kolodner, "An introduction to case-based reasoning," *Artificial Intelligence Review*, vol. 6, no. 1, pp. 3–34, Mar. 1992.

[11] D. C. Wilson and D. B. Leake, "Maintaining case-based reasoners: Dimensions and directions," *International Computational Intelligence Journal*, vol. 17, no. 2, pp. 196–213, Dec. 2002.

[12] R. Qu, "Case-based reasoning for course timetabling problems," Ph.D. dissertation, Department of Computer Science, University of Nottingham, Nottingham, UK, 2002.

[13] J. Zhu, "Similarity metrics and case base maintenance," M.S. thesis, Department of Computing Science, Simon Fraser University, British Columbia, Canada, 1998.

[14] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994.

[15] A. Lawanna, "An effective model for case-based maintenance in case-based reasoning systems," in *Proceedings 2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, Mar. 2016, pp. 129–134.

[16] D. B. Leake and D. C. Wilson, "Categorizing case-base maintenance: Dimensions and directions," in *Proceedings of the 14th European Workshop on Advance in Case-Based Reasoning (EWCBR'98)*, vol. 1488, pp. 196–207, 1998.

[17] B. Smyth and P. Cunningham, "The utility problem analyzed a case-based reasoning perspective," in *Proceedings Third European Workshop EWCBR-96 Lausanne*, *Switzerland*, 1996, pp. 392–399.

[18] B. Smyth and M. T. Keane, "Experiments on adaptation-guided retrieval in case-based design," in *Proceedings International Conference on Case-Based Reasoning*, Oct. 1995, vol. 1010, pp. 313–324.

[19] M. K. Haouchine, B. Chebel-morello, and N. Zerhouni, "Case base maintenance approach," in *Proceedings International Conference on Industrial Engineering and Systems Management (IESM'2007)*, 2007, pp. 10.

[20] B. Smyth and E. McKenna, "Modelling the competence of case-bases," in *Proceedings 4th European Workshop (EWCBR-98)*, 1998, pp. 208–220.

[21] A. Smiti and Z. Elouedi, "Maintaining case based reasoning systems based on soft competence model," in *Proceedings International Conference on Hybrid Artificial Intelligence Systems*, Jun. 2014, pp. 666–677.