Improving Efficiency of Self-Configurable Autonomic Systems Using Clustered CBR Approach

Malik Jahan KHAN†(a), Member, Mian Muhammad AWAIS†(b), and Shafay SHAMAIL†(c), Nonmembers

SUMMARY  Inspired from natural self-managing behavior of the human body, autonomic systems promise to inject self-managing behavior in software systems. Such behavior enables self-configuration, self-healing, self-optimization and self-protection capabilities in software systems. Self-configuration is required in systems where efficiency is the key issue, such as real time execution environments. To solve self-configuration problems in autonomic systems, the use of various problem-solving techniques has been reported in the literature including case-based reasoning. The case-based reasoning approach exploits past experience that can be helpful in achieving autonomic capabilities. The learning process improves as more experience is added in the case-base in the form of cases. This results in a larger case-base. A larger case-base reduces the efficiency in terms of computational cost. To overcome this efficiency problem, this paper suggests to cluster the case-base, subsequent to find the solution of the reported problem. This approach reduces the search complexity by confining a new case to a relevant cluster in the case-base. Clustering the case-base is a one-time process and does not need to be repeated regularly. The proposed approach presented in this paper has been outlined in the form of a new clustered CBR framework. The proposed framework has been evaluated on a simulation of Autonomic Forest Fire Application (AFFA). This paper presents an outline of the simulated AFFA and results on three different clustering algorithms for clustering the case-base in the proposed framework. The comparison of performance of the conventional CBR approach and clustered CBR approach has been presented in terms of their Accuracy, Recall and Precision (ARP) and computational efficiency.

key words:  Autonomic computing, self-management, case-based reasoning, clustering

1. Introduction

Inspired from the self-managing behavior of the human body, autonomic computing [16], [19], [26], [40] is a promising concept to incorporate autonomic behavior in software systems with the intention of minimizing human intervention in managing software systems, thus reducing the cost and time of system management. Autonomic systems possess many self-management capabilities including self-configuration, self-optimization, self-protection and self-healing [19], [26], [40]. Self-configuration capability enables seamless adaptation of new execution environment without significant human intervention [5], [24]. Self-optimization capability optimizes various parameters at run time according to environmental conditions [35]. Self-protection capability protects the system from malicious attacks [9]. Self-healing capability is responsible for recovering the system from failures [41]. Self-configuration in autonomic systems is a continuous process and the system has to keep on configuring itself with the passage of time and varying environmental conditions. Various artificial intelligence techniques like rule-based systems [24], control theory [2], [12], [15], [25], [44], AI planning [5], case-based reasoning (CBR) [27], [28], [38] etc. have been used to devise new configuration settings in autonomic systems.

CBR [1], [6], [11], [34], [47] is a reasoning methodology which uses past experience to find solution of a new problem. Past experience is maintained in a repository called a case-base. When a new problem comes in, its description is compared with the cases in the case-base and the similarity between the current case and all cases in the case-base is calculated using certain similarity measures [18], [30]. A set of similar cases is developed that is used to devise the solution to the given problem. The solution obtained is revised within certain limits, and the case-base is updated for future use, if required. This revision and retention property of CBR is a feature distinguishing it from other artificial intelligence techniques and adds to its problem-solving power.

Due to added problem-solving capability, CBR promises many potential benefits as compared to other conventional methods used for self-management in autonomic systems. These include:

1. Flexibility in case representation due to non-structural approach as opposed to AI planning, rule-based systems and control theory
2. Simple and natural self-managing behavior due to nature inspired solution strategy
3. Comparatively lesser input pre-processing overhead
4. The capability of exploiting past and the most recent experiences to suggest a solution of the current problem
5. Flexible revision mechanism for updating and upgrading of a suggested solution
6. Minimum off-line learning requirement as compared to most of the probabilistic and statistical learning techniques

Along with many advantages, conventional CBR also experiences efficiency bottleneck due to continuous updating of the case-base. As new cases are retained after revision, the size of the case-base continuously increases re-
sulting in a large number of comparisons required before a solution is presented. This paper proposes a clustered case-based reasoning (CBR) approach to overcome this efficiency bottleneck in achieving self-configuration mechanism for autonomic systems. The proposed approach promises to exploit the recent experience equally and at the same time reduces the computational cost of finding a solution to a given problem without compromising on the accuracy of the system.

In their previous work [27], [28], Khan et al. have suggested to use conventional CBR approach to achieve self-configuration in autonomic systems. Existing application of CBR in autonomic systems is limited in terms of efficiency of solution finding algorithm and explosive case-base size. As self-configuration is a continuous process, case-base keeps on increasing in size and incurs high computational cost in finding similarity of current case with each existing case. In real time application, it harms the efficiency constraints. This paper suggests to cluster the case-base into reasonable number of clusters using an appropriate clustering algorithm. Once a new problem comes in, it is classified among one of these clusters, and the cases within the relevant cluster are used to devise the solution of the new problem. Now, solution devising step is limited to a single cluster instead of the whole case-base, which will significantly reduce the number of comparisons needed to find similar cases. This paper presents the proposed approach in the context of self-configuration capability of autonomic systems. Two algorithms are discussed here in this regards. The first algorithm presents the method for clustering the case-base and the second algorithm outlines the clustered CBR-based solution finding methodology. As part of the first algorithm, implementation and comparison of three different clustering algorithms is presented, i.e. k-Means, FarthestFirst and density-based clustering algorithms. A performance comparison of the conventional CBR approach with the clustered CBR based approach is presented in terms of Accuracy, Recall and Precision (ARP) and computational cost.

The performance of the proposed approach has been tested on a simulation of standard Autonomic Forest Fire Application (AFFA) presented in [32], [33].

The rest of the paper is organized as: Sect. 2 discusses related background work on autonomic computing and CBR. Section 3 explains the problem domain and proposed approach. Section 4 explains the results, implementation and performance analysis to support the suggested approach in the context of AFFA. Finally, the conclusion and future directions are discussed in Sect. 5.

2. Background and Related Work

In this section, an overview of autonomic computing and its various properties, focusing on self-configuration and case-based reasoning is presented. A brief literature review of various implementations of autonomic systems is also presented.

2.1 Autonomic Computing

With passage of time, the reliance and dependence on computers have increased tremendously. People have been more curious to make everything automated, speedy and accurate, but the additional layer of complexity has not been taken care of, which is added to the existing systems in order to enhance the level of automation. With the increased functionality, software systems grew in size as well as complexity. Today, it has become convenient to solve a lot of problems using such large complex systems, but within these systems, complexity itself is a huge problem. Whenever, something happens unexpectedly or accidentally, it becomes very difficult to find solution within limited time. Sometimes, solution is trivial to find and sometimes it isn’t. In large and complex systems, diagnosis of problem and then solving it manually is not an easy job. Because of the heavy dependence on such systems, delays in problem solving cannot be afforded. Immediate solutions are needed to minimize the downtime or time of abnormal behavior of the systems.

Inspired from the human nervous system, in which most of the activities inside human body are performed without permission of human itself, autonomic computing is a concept which recommends to inject some self-managing capabilities inside the large systems so that human intervention for managing those systems can be minimized. This additional layer of complexity performs a lot of automatic and autonomous problem-solving functionalities. There are four basic properties of autonomic systems which are collectively known as self-* properties. These properties introduce the autonomic behavior within a system. To implement or enable any of these properties, autonomic manager continuously monitors the managed element, analyzes the monitored data, plans to take some intelligent actions and then executes those actions onto the managed element. These properties include [26]:

- **Self-configuration**: It enables the system to automatically and autonomously configure the system to adapt a new component or new execution environment [5], [24].
- **Self-optimization**: It enables monitoring, experimenting and tuning various parameters to make best use of the existing resources [26].
- **Self-healing**: It detects various system failures and recovers from failures to enable minimum downtime of the system [41].
- **Self-protection**: It protects the system from various malicious attacks [9].

2.2 Case-Based Reasoning

CBR is a problem-solving methodology which utilizes the past experience on similar kind of problems to solve the current problem in an elegant fashion. Past experience is maintained in a repository in the form of problem-solution
pairs. Each pair is known as a case and the whole repository is referred to as a case-base. When a new problem arises, it is compared with the existing cases in the case-base. Similarity is calculated between the current case and every case in the case-base. Those cases are taken into account which have a reasonable amount of similarity with the problem. Various similarity metrics are used for this purpose, and their selection depends upon the level of accuracy needed and the domain of the problem. CBR systems work in four major phases. [11], [34]:

- **Retrieve**: Current case is compared with the existing cases in the case base and most matching cases are retrieved. To accomplish this task, various similarity metrics are used. Some of the commonly used similarity metrics for this purpose include City block distance [18], Euclidean distance [18], Mahalanobis distance [30], geometric similarity metrics [17], probabilistic similarity measures [39] and similarity measure based on fuzzy inner and outer products [27]. This gives us a set of nearest neighbors of the current case.

- **Reuse**: Solution of the nearest neighbors is used to devise solution of the current case. There exist various solution algorithms which can be applied on solutions of the nearest neighbors to find out the solution of the case at hand. These algorithms include arithmetic average, weighted average etc [18]. Selection of a solution algorithm depends on the priorities of neighbors.

- **Revise**: For better adaptation, we may need to revise the solution of the current problem. However to make CBR more sensible, adaptation phase should be kept minimal. Substantial case adaptation or revision may harm the knowledge engineering advantages to be obtained from CBR [11].

- **Retain**: Once we have completed the job of devising solution of the current case, we retain this new problem-solution pair in the case-base. This is an attractive characteristic of the CBR approach as compared to other machine learning strategies that a retaining solution algorithms which can be applied on solutions of the current case. However, to make CBR more sensible, adaptation phase should be kept minimal. Substantial case adaptation or revision may harm the knowledge engineering advantages to be obtained from CBR [11].

2.3 Literature Review of Existing Applications

This section presents a literature review of the existing applications of autonomic computing. Various techniques and tools adapted to enable self-managing behavior within the autonomic managers have been summarized in Table 1.

The literature survey reveals that most of the techniques do not support continuous learning with the changing execution environment. However, CBR is a lazy learning model which continuously improves the learning curve of the learner through its retain phase. The tradeoff observed from this option is continuously increasing size of the case-base due to retain phase. It incurs high computational costs when the case-base has significantly grown. To overcome this efficiency bottleneck, this paper proposes a clustered CBR approach for self-configurable autonomic systems.

In literature, clustering has been applied with CBR in other problem domains for case indexing purposes. Kim and Han [31] proposed an indexing method using competitive ANN’s like self-organizing maps (SOM) and leaning vector quantization (LVQ). Both of their methods are used to find centroid values of the clusters representing all cases. Number of clusters are manually determined by initializing the weight vectors which converge as centroid values of the clusters. Once the centroid values have been calculated then a decision tree scheme is applied to decide the membership of each case to a particular cluster based on the distance from each of the centroid values. Solution of the new case is computed by indexing the case in either of the clusters and confining solution space within the cluster. Though this approach results in the improvement of retrieval efficiency, it is static in nature. Number of clusters is decided in prior and remains constant in future. It does not guarantee to yield the accurate solution at user defined number of clusters. As the size of the case-base increases due to retention process, efficiency bottleneck arises again and this scheme may fail to find the remedy. Similarly, Chiu and Tsai [10] proposed a weighted feature C-Means clustering algorithm to define k clusters with the objective of minimizing the distances among all objects and their corresponding centers. It is explicitly stated in [10] that user has to decide the value of k. The purpose of applying clustering procedure is to determine the weights of features. The features which play vital role in clustering process are assigned higher weights for solution finding process. Determining the cluster for a new case is same as in [31]. This approach also exhibits efficiency bottleneck as the case-base size grows and no criteria has been proposed to determine the number of clusters. Also, the retrieval strategy is dependent on the dissimilarity threshold which may lead to expand the solution space to many clusters. This flexibility goes against the objective of the study which claims to enhance the retrieval efficiency. Hong and Liou [23] also used clustering with case-based reasoning but their objective was to determine the significance of features using clustering so that the overall feature space should be reduced which may lead to retrieval effi-
### Table 1  Summary of existing techniques to enable autonomic behavior.

<table>
<thead>
<tr>
<th>Title</th>
<th>Brief Description</th>
<th>Supported Autonomic Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control theoretic approaches [2], [12], [15], [25], [44]</td>
<td>Controller with feedback loop acts as autonomic manager, and it monitors and fine-tunes various attributes in the execution environment</td>
<td>Self-optimization, Self-configuration</td>
</tr>
<tr>
<td>Probabilistic approach: active probing [42]</td>
<td>Appropriate probes are selected and sent in a distributed environment to diagnose a problem</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Probabilistic approach: Bayesian estimation [45]</td>
<td>PeerPressure is a statistical solution to diagnose problems resulted from various misconfigurations of the system</td>
<td>Self-healing, Self-configuration</td>
</tr>
<tr>
<td>Mathematical modeling [35]</td>
<td>An autonomic query optimizer known as LEO validates mathematical model of query execution and is capable to correct wrong estimates</td>
<td>Self-optimization</td>
</tr>
<tr>
<td>Bounded resource usage [36]</td>
<td>An autonomic failure detection algorithms suggests a self-regulating mechanism based on bounds on resource usage and failure detection latency</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Cheap recovery [7]</td>
<td>Micro-reboot is a cheap recovery technique which starts recovery process by restarting components from a granular level and incrementally grows</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Rollback strategy [41]</td>
<td>Rx is a failure recovery technique which rolls back the system to a previously known safe state and re-executes the modified environment</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Reinforcement learning [3]</td>
<td>This technique picks an appropriate action from a finite pool of actions using reinforcement learning</td>
<td>Self-configuration</td>
</tr>
<tr>
<td>Decision tree learning [8]</td>
<td>Various parameters representing the system state are ranked based on the information gain value and used to determine problem in a distributed environment</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Conversational CBR approach [4], [37], [38]</td>
<td>A case is prepared by asking various questions about the system state and simple matching with previous cases is used to diagnose the problem</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Hybrid approach [20]</td>
<td>A hybrid approach using conversational CBR and rule-based technique is used to diagnose the problem and find solution</td>
<td>Self-healing</td>
</tr>
<tr>
<td>Conventional CBR approach [27]–[29]</td>
<td>This approach exploits various features of CBR like various similarity options, revise and retain strategies with different measures</td>
<td>Self-configuration</td>
</tr>
</tbody>
</table>

ciency. These approaches have following common limitations:

1. The number of clusters is not guaranteed to be optimal. If it has not been selected appropriately, it may also lead to poor performance in terms of accuracy, recall and precision as evident from our empirical study.
2. The efficiency bottleneck may reappear, once sizes of individual clusters grow explosively and get closer to the original size of the case-base.
3. Selection of the clustering algorithm to be applied is biased and none of the conventional algorithms have been exploited.

This paper addresses the above mentioned limitations in the following ways:

1. To find the optimal number of clusters for the given case-base, an accuracy-based decision criteria has been devised and presented
2. Re-clustering decision criteria has devised in order to overcome the common drawback of the existing approaches.
3. The proposed approach offers the flexibility to select any clustering algorithm and exploit it.

An additional contribution of this paper is that clustered CBR based approach has been proposed and applied in the domain of autonomic computing to handle the efficiency bottleneck in computing systems.

### 3. Proposed Approach: Self-Configuration Using Clustered CBR

To enable self-configuration capability in autonomic systems, Khan et al [27], [28] suggested to use conventional
CBR cycle. According to [27], [28], whenever a new configuration problem is observed by the sensor of autonomic manager, its description is compared with all the cases of the case-base using an appropriate similarity measure and a set of nearest neighbors is extracted from the complete case-base. Finding the similarity values is very costly in terms of computational complexity, especially when there are large number of cases present in the case-base. Once the set of nearest neighbors is found, the solution strategy is applied to devise the solution for the new configuration problem. Based on the above discussion, following two major limitations are identified:

1. New case was compared with all the cases present in the case-base irrespective of their similarity with the new case. Inclusion of less similar cases in the solution finding process caused reduction in accuracy, recall and precision.
2. Size of the complete case-base increased due to retention phase of CBR cycle, thus resulting in increased computational complexity of the solution finding process of new problem.

The current work proposes remedy to the above mentioned drawbacks by modifying the conventional CBR cycle. The modifications are highlighted as follows:

1. M1: Convert the standard case-base into a clustered case-base containing \( k \) clusters
2. M2: Before applying the retrieve phase of the conventional CBR cycle, classify the new case into one of the identified clusters, i.e. predictedCluster
3. M3: Find the solution for the new case within predictedCluster using CBR cycle

By incorporating the above mentioned modifications in the CBR cycle, the architecture of autonomic systems changes to clustered CBR-based framework as shown in Fig. 2. Based on the proposed architecture, Algorithms 1 and 2 are presented in this paper to accomplish the above-mentioned modifications.

Given a large case-base, a clustering algorithm is selected to be applied on the case-base and the cases are divided among reasonable number of clusters. Their performance is compared with the non-clustered CBR approach using Accuracy, Recall and Precision (ARP) analysis [43], [48] and efficiency analysis. The details of these algorithms are presented in the following subsections.

### 3.1 Algorithm 1: Construction of Clustered Case-Base

In order to incorporate modification \( M1 \) in the conventional CBR-based autonomic cycle, algorithm 1 is proposed as shown in Fig. 3. Algorithm 1 takes inputs as training case-base \( CB \), a similarity measure \( SM \), a solution algorithm \( SA \), a clustering algorithm \( CA \) and a test case-base \( TCB \). The algorithm outputs clustered case-base containing \( k \) clusters. The choice of \( k \) is based on the accuracy of solution prediction of the clustered case-base. An optimal number of clusters (\( k_{opt} \)) is searched in this iterative process which results in the highest accuracy. If Algorithm 1 is to be executed for the very first time then the value of \( k_{opt} \) will be taken as 2 otherwise it will be equal the number of clusters in \( CCB \) formed during the previous run of Algorithm 1. Here, \( accuracy_k \) is the accuracy of solution prediction process using the clustered case-base containing \( k \) clusters. By applying the clas-

---

**Input:** A case-base \( CB \) containing \( n \) cases, similarity measure \( SM \), solution algorithm \( SA \), clustering algorithm \( CA \), test case-base \( TCB \), \( k_m \)

**Output:** Clustered case-base \( CCB \)

**Method:**

1. \( currentAccuracy := 0 \)
2. For \( k := k_m \) to \( n/2 \) do
   a. \( accuracy_k := 0 \)
   b. \( CCB := CA(CB, k) \)
   c. For each \( c_p \in TCB \)
      i. \( cluster_p := classify(c_p, CCB) \)
      ii. For each \( c_p \in cluster_p \) and \( p \neq j \)
         1. \( sim_p := \text{findSimilarity}(c_p, c_p, SM) \)
         2. \( w_p := sim_p \)
      iii. End For
   d. End For
   e. If \( (accuracy_k > currentAccuracy) \)
      i. \( currentAccuracy := accuracy_k \)
      ii. \( k_{opt} := k \)
   f. End If
3. End For
4. \( CCB := CA(CB, k_{opt}) \)

---

![Fig. 3](image-url)  
**Fig. 3** Algorithm 1: Construction of the clustered case-base.
sification process, \( c_j \) of the test case \( c_j \) is determined. Subsequently, each member \( c_p \) of \( c_j \) is compared with \( c_j \) to compute similarity of \( c_p \) (sim\(_p\)) using SM. \( w_p \) represents the weight of \( c_p \) which is computed using sim\(_p\) and is stored at \( p - th \) index of overall weight vector \( w \). \( SA \) and \( w \) are used to compute solution of \( c_j \) (sol\(_j\)). accuracy\(_k\) is updated based on the difference between sol\(_j\) and actual solution of \( c_j \).

Algorithm 1 can be adapted for any clustering algorithm. The empirical study presented in this paper investigates the choice of a particular clustering algorithm. As the preprocessing phase of the proposed approach, this algorithm has been devised to determine the appropriate number of clusters for the case-base. This algorithm does not need to be frequently executed. The decision to re-cluster the case-base primarily depends upon the size of the largest cluster. Whereas size of clusters depends on the distribution of the training data. Algorithm 1 is re-executed after merging the pre-built clusters once size of the largest cluster approaches the previous size of the case-base (\( n_{prev} \)) which was used during the previous run of Algorithm 1. This criteria is validated on the retention of every new case which is implemented in Algorithm 2.

3.2 Algorithm 2: Devising the Solution of the New Problem

Algorithm 2 implements the clustered-based CBR cycle as outlined in Fig. 4. Step 1 implements modification M2 and steps 2 to 5 implement modification M3 of the proposed approach. When a new case \( c_p \) comes in, it is classified to one of the relevant clusters found through Algorithm 1. Now the conventional CBR cycle is applied on the new case and its relevant cluster that consists of finding similarity between \( c_p \) and all the cases present in the relevant cluster, finding the weights and finally predicting the solution of \( c_p \). As a result of the application of Algorithm 2, the comparison space for \( c_p \) limits to only its relevant cluster, thus reducing the computational complexity and increasing the efficiency. Step 6 implements the re-clustering of the given case-base. If size of the predictedCluster approaches \( n_{prev} \) then re-clustering is needed. Number of existing clusters of the case-base (\( k_u \)) is computed and then all clusters are merged. Subsequently, Algorithm 1 is applied to re-cluster the case-base.

Given that this approach is applied on large datasets, clustering the case-base will significantly reduce the search space for similarity calculations and retrieval of the set of nearest neighbors. If there are \( n \) number of cases in the case-base and they are divided among \( k \) clusters, then instead of \( n \) comparisons, on average only \( n/k \) comparisons are needed. This reduction in the size of search space will enhance the efficiency of the CBR solution devising procedure. Also, only relevant cases contribute towards the solution of the new case which promises better ARP performance.

3.3 Computational Complexity

Computational complexity of the solution devising process for each new case without clustering is \( O(nm) \) where \( n \) is number of cases in the case-base and \( m \) is the complexity of similarity function. If \( k \)-Means clustering algorithm is used as part of the Algorithm 1, then computational complexity of the \( k \)-Means algorithm is \( O(nkt) \) [18], where \( t \) is number of iterations required to complete the clustering process. For each \( k \), clustering algorithm is applied and each case is treated as the test case using the Leave-One-Out (LOO) validation process. The testing process takes \( n^2/k \) steps as \( n/k \) is the average cluster size. If \( O(m) \) is the computational complexity of similarity measure then the testing process takes \( O(n^2m/k) \) steps. If we deploy \( k \)-Means clustering algorithm as part of Algorithm 1 then overall computational complexity of Algorithm becomes \( O(n(nkt + n^2m/k)) \). Similarly, if FarthestFirst or density-based clustering algorithms are selected, their computational complexity in worst case is \( O(n^2) \). Hence, the complexity of Algorithm 1 will be \( O(n(n^2 + n^2m/k)) = O(n^3) \). As Algorithm 1 is not executed on continuous basis, so computational cost of any of the clustering algorithm does not harm the case retrieval efficiency. It will be executed once significant number of new cases have been adapted, and re-clustering the case-base is needed. However, an appropriate clustering algorithm has to be selected which results in improved performance in terms of ARP analysis. The computational complexity of the proposed approach to devise solution of each new case using Algorithm 2 is \( O(nm/k) \) where \( k \) is number of clusters and in case of large case-bases, it is a significant reduction in the total cost of solution devising process. Value of \( k \) depends on the size of the case-base \( n \), i.e. \( k = f(n) \). Hence, computational complexity of Algorithm 2 is \( O(nm/f(n)) \). The empirical investigation of our experiments reveals that \( k \gg \sqrt{n} \) which supports the significant decrease in the computational cost of the proposed approach as compared to unclustered
in.autonomic approach.

4. Experimental Setup and Results

In order to test performance of the proposed approach presented in Sect. 3, the Autonomic Forest Fire Application (AFFA) [32], [33] was selected. This section briefly describes the AFFA and explains the application of modified CBR cycle including case preparation, classification process and solution formulation. The application of three different clustering algorithms and their comparison with unclustered approach in terms of ARP analysis and efficiency analysis are also presented here.

4.1 Case Study: Autonomic Forest Fire Application

Autonomic Forest Fire Application (AFFA) [32], [33] is a simulation tool that forecasts the strength, speed and direction of the fire as it propagates in a forest under various environmental conditions. This application consists of five components: Data Space Manager (DSM) that represents the forest as a two-dimensional array of cells, Computational Resource Manager (CRM) that keeps inventory of the available resources and updates DSM, Forest Grid that maintains the status of each cell as burnt, burning and unburnt, Rothermel that computes the value and direction of fire spread, and WindModel that provides wind direction and intensity values to the Rothermel.

AFFA, in the context of the present study, has been customized to predict the probability of a particular cell to be burnt. Based on the predicted probability, CRM updates its configurations using an appropriate confusion script (CS) and takes precautionary steps to prevent the unburnt part of the forest from getting burnt.

4.2 Case Preparation

For AFFA, a case consists of seven parameters: five of these parameters define the current state of the forest, the sixth parameter defines the predicted probability of the spread of the forest fire and the seventh parameter represents name of a configuration script which is executed based on the value of predicted probability. The first five state parameters have been extracted from description of various ports of Autonomic Forest Fire Application as given in [32], [33]. These parameters include Number of Burning Cells in the Grid (N), Wind Direction with respect to a cell X (WD), Fire Speed (FS), Wind Intensity (WI) and Minimum Distance from the Burning Cell (MD). Using these five parameters, the modified CBR cycle predicts the probability (PP) that a cell X will be exposed to fire. PP is used to dynamically re-configure the execution environment of AFFA using a configuration script (CS). Each case of AFFA is represented as given in Eq. (1). For experimental purposes, three case-bases of sizes 500, 2000 and 10000 cases were generated and used as part of the knowledge repository in autonomic computing cycle.

Each case of AFFA is represented as a set of the above mentioned parameters:

\[ c_i = \{N, WD, FS, WI, MD, PP, CS\} \] (1)

The overall case-base \( CB \) of size \( n \) is represented as:

\[ CB = \{c_1, c_2, \ldots, c_n\} \] (2)

4.3 Clustering the Case-Base

After generating the case-bases of \( n = 500 \), \( n = 2000 \) and \( n = 10000 \), Algorithm 1 was applied to cluster each of the case-bases. Nine separate case studies were conducted for evaluating the performance of the proposed approach employing a separate clustering algorithm and separate case-base for each case-study. The clustering algorithms include k-Means, FarthestFirst and density-based algorithms. These algorithms were run using their Weka implementations [22], [46], [49]. The value of \( k \) was varied from 2 to \( n/2 \) for each case-study. Various simulations were done using the three clustering algorithms and an analysis to find the most appropriate value of \( k \) that satisfied the objective was conducted.

4.4 Implementing the Clustered-CBR Cycle

As the first step of the clustered CBR-cycle, the new case was classified among one of the relevant clusters. The similarity of the current case with all cases within that particular cluster was found. Euclidean distance based similarity measure was used for this purpose as given in Eq. (3).

\[ d_{ij} = \sqrt{\sum_{k=1}^{\text{size(cluster)}} z_k (c_{ik} - c_{jk})^2} \] (3)

Where \( d_{ij} \) is the distance between \( i \)th and \( j \)th cases, \( c_{ik} \) and \( c_{jk} \) are the \( k \)th attributes of cases \( c_i \) and \( c_j \) respectively and \( z_k \) is the weight of \( k \)th attribute. Similarity between the new case and each case in the cluster was used to weigh the contribution of each case towards the solution as given in Eq. (4).

\[ w_i = \frac{1/d_{ij}}{\sum_{k=1}^{\text{size(cluster)}} 1/d_{jk}} \] (4)

Where \( w_i \) is the weight of \( i \)th case for solution finding process. The solution of the new case is computed using the weighted average [30] of all the cases in the corresponding cluster, as shown in Eq. (5).

\[ sol_p = \frac{\sum_{i=1}^{\text{size(cluster)}} w_i \cdot sol_i}{\sum_{i=1}^{\text{size(cluster)}} w_i} \] (5)
4.5 Performance Analysis Method

The proposed approach has been implemented and tested using Leave-One-Out (LOO) validation technique. Every case among the n cases was treated as the test case and remaining n – 1 cases acted as the training cases. There are two dependent solution parameters: PP and CS. The CS parameter is a nominal variable, so it is treated as a class label. There are four different configuration scripts which result in a four-class problem. We construct a binary confusion matrix for every class versus all other classes. For each experiment, four confusion matrices are constructed and accuracy, recall and precision are computed individually. Later, they are averaged to compute the overall performance of the experiment. Format of a confusion matrix is shown in Table 2. In this matrix, a represents positive examples classified correctly, b represents positive examples misclassified, c represents negative examples misclassified and d represents negative examples classified correctly. Based on the values of a, b, c and d, three evaluation measures Accuracy, Recall and Precision are computed as shown in the Eqs. (6), (7) and (8) [43], [48].

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} \quad (6)
\]

\[
\text{Recall} = \frac{a}{a + c} \quad (7)
\]

\[
\text{Precision} = \frac{a}{a + b} \quad (8)
\]

Sample computations of one experimental setup for n = 500 using k-Means clustering algorithm at k = 60 have been outlined in the Tables 3, 4, 5, 6 and 7. Similar experiments were conducted for each n and each clustering algorithm while varying value of k from 2 to n/2 and results have been aggregated in the Figs. 5 to 13.

Table 2 Sample confusion matrix for CS4.

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Accuracy</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS1</td>
<td>0.95</td>
<td>0.44</td>
<td>0.90</td>
</tr>
<tr>
<td>CS2</td>
<td>0.79</td>
<td>0.80</td>
<td>0.74</td>
</tr>
<tr>
<td>CS3</td>
<td>0.80</td>
<td>0.77</td>
<td>0.73</td>
</tr>
<tr>
<td>CS4</td>
<td>0.96</td>
<td>0.62</td>
<td>0.84</td>
</tr>
<tr>
<td>Overall</td>
<td>0.87</td>
<td>0.66</td>
<td>0.80</td>
</tr>
</tbody>
</table>

4.6 Performance Comparison of Clustered versus Unclustered Approach

As discussed in the earlier sections, the main objective of the proposed approach is performance improvement. The performance is measured in terms of ARP analysis and efficiency analysis. The detailed ARP and efficiency analysis is
4.6.1 ARP Analysis

Figures 5, 6 and 7 represent the behavior of the three clustering algorithms in terms of accuracy with different cluster sizes \( k \) for three datasets of sizes 500, 2000 and 10000 respectively. FarthestFirst clustering algorithm performs best across all experimental setups among the other two clustering algorithms. Unclustered approach performance has also been pointed out in these figures using a \( \star \) sign. These results reveal that clustered approach performs almost the same or marginally better than the unclustered approach leading up to 94% accuracy (performance of FarthestFirst clustering algorithm for \( n = 10000 \) at \( k = 2012 \)). In terms of recall, the behavior of FarthestFirst clustering algorithm is much better than the other clustered algorithms and unclustered approach as shown in Figs. 8, 9 and 10 resulting in up to 87% recall performance (performance of FarthestFirst clustering algorithm for \( n = 10000 \) at \( k = 2012 \)). Here, again the conventional unclustered CBR results deteriorate in comparison with the clustered approach. Our proposed approach exhibits considerable improvement as compared to unclustered approach in terms of the recall because solution space is now confined to the relevant cluster. Precision of the clustered approach is much higher than the conventional unclustered approach leading up to 85% precision as evident from Figs. 11, 12 and 13 (performance of FarthestFirst clustering algorithm for \( n = 10000 \) at \( k = 2012 \)). In terms of overall ARP analysis, FarthestFirst based clustered CBR approach performed much better than the other two clustered approaches and conventional CBR approach, and is suggested to be opted as part of Algorithm 1 based on the empirical results. Density-based clustering algorithm seems
Table 8 Percentage improvement in retrieval efficiency for optimal values of $k$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>Clustering Algorithm</th>
<th>Optimal Value of $k$</th>
<th>Best Case</th>
<th>Average Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>$k$-Means</td>
<td>36</td>
<td>99.80%</td>
<td>96.40%</td>
<td>93.00%</td>
</tr>
<tr>
<td>500</td>
<td>FarthestFirst</td>
<td>123</td>
<td>99.80%</td>
<td>98.60%</td>
<td>97.40%</td>
</tr>
<tr>
<td>500</td>
<td>Density-based</td>
<td>63</td>
<td>99.00%</td>
<td>90.10%</td>
<td>25.20%</td>
</tr>
<tr>
<td>2000</td>
<td>$k$-Means</td>
<td>500</td>
<td>99.95%</td>
<td>99.47%</td>
<td>99.00%</td>
</tr>
<tr>
<td>2000</td>
<td>FarthestFirst</td>
<td>440</td>
<td>99.95%</td>
<td>99.55%</td>
<td>99.15%</td>
</tr>
<tr>
<td>2000</td>
<td>Density-based</td>
<td>305</td>
<td>99.90%</td>
<td>52.75%</td>
<td>5.60%</td>
</tr>
<tr>
<td>10000</td>
<td>$k$-Means</td>
<td>3021</td>
<td>99.99%</td>
<td>99.88%</td>
<td>99.77%</td>
</tr>
<tr>
<td>10000</td>
<td>FarthestFirst</td>
<td>2012</td>
<td>99.99%</td>
<td>99.91%</td>
<td>99.84%</td>
</tr>
<tr>
<td>10000</td>
<td>Density-based</td>
<td>4001</td>
<td>99.99%</td>
<td>99.08%</td>
<td>98.16%</td>
</tr>
</tbody>
</table>

Fig. 12 Precision of clustered approach vs unclustered approach for $n = 2000$.

Fig. 13 Precision of clustered approach vs unclustered approach for $n = 10000$.

more sensitive to the changing values of $k$ as compared to FarthestFirst and $k$-Means clustering algorithms. Sensitivity analysis of the clustering algorithms may be part of the future work.

4.6.2 Efficiency Analysis

The efficiency analysis presented in this paper is based on the computational performance, specifically cost incurred in the retrieval phase of the CBR cycle, of the clustered versus unclustered approach. In case of the unclustered approach, 500, 2000 and 10000 comparisons were required to find out a new solution for $n = 500$, $n = 2000$ and $n = 10000$ respectively. Whereas, the efficiency on retrieving the similar cases in the proposed clustered CBR based approach exhibited a significant improvement as shown in Table 8. The table presents the improvement in retrieval efficiency in terms of best case, worst case and average case analysis for the three clustering algorithms used in this study. The best, worst and average case analyses are based on the smallest cluster size, largest cluster size and average cluster size respectively obtained through the application of Algorithm 1 of the proposed approach at optimal values of $k$. Optimal values of $k$ in each experimental setup were determined using Algorithm 1.

Efficiency improvement is defined and computed in terms of percentage decrease in terms of number of comparisons needed in retrieval phase of CBR cycle. Results presented in Table 8 reveal that FarthestFirst clustering algorithm is the best candidate in terms of efficiency as well and performs better in all experimental setups.

4.6.3 Effect of $k$ on Performance

For the proposed approach to work, a reasonable value of $k$ has to be determined. The value of $k$ should be such that the overall retrieval cost in the CBR cycle reduces to a reasonable level without affecting the accuracy of the new solution significantly. For the case study presented in this paper, $k$ was varied from 2 to $n/2$ and a record of average absolute error obtained for a given scenario was maintained. This study revealed that $k \gg \sqrt{n}$ which gives a clue about the impact of $k$ on the improvement of overall efficiency of the proposed approach.

5. Conclusion and Future Work

Conventional CBR based systems present an efficiency bottleneck in terms of computational cost, and especially for self-managing applications. This paper focuses on resolving efficiency bottleneck by introducing a new clustered CBR based framework for autonomic systems. Consequently, this approach reduces the computational cost and retrieval time without adversely affecting the overall performance of the system in terms of Accuracy, Recall and Precision (ARP). In the proposed framework, when a new problem is presented as a case, it is classified into one of the predetermined clusters that acts as a comparison space for the present case rather than the complete case-base. The CBR cycle is then applied to the selected cluster only, thus considerably reducing the number of comparisons. This reduces the time complexity and enhances the case retrieval efficiency. To show the effectiveness of the proposed approach, the proposed framework has been applied to three experimental setups of AFFA. For clustering the case-base, experiments with three different clustering algorithms have been conducted and compared using their ARP performance.
and efficiency performance. At an optimal value of $k$, Accuracy, Recall and Precision of the proposed approach can be achieved up to 94%, 87% and 85% respectively. Our results also show that the proposed approach outperforms the conventional unclustered approach with an efficiency improvement of up to 99%. The empirical investigation conducted in this paper reveals that FarthestFirst clustering algorithm is the optimal option to be used as building block of Algorithm 1. The proposed approach has been compared with the conventional CBR approach which has been used as the comparison benchmark.

In the future, we intend to enhance the performance of the proposed approach by devising an adaptation algorithm to refine the suggested solution. Density-based clustering algorithm shows more rippling effect as compared to the other clustering algorithms and is sensitive to the changing value of $k$. Sensitivity analysis of the clustering algorithms is part of the future work. The current approach does not fully handle uncertainty and vagueness. One way to handle this would be to fuzzify the clustered CBR case-base. The combined fuzzy-clustered CBR approach would be a reasonable way to capture uncertainty and vagueness in real systems while keeping the ARPA analysis and computational cost under control. As part of the future work, the capability of the proposed approach will be tested on other self-management capabilities of autonomic systems like self-optimization, self-protection and self-healing applications.

References


Malik Jahan Khan is a Ph.D. candidate at Department of Computer Science, LUMS, Lahore, Pakistan. He received his MS in computer science in 2005 from LUMS. Currently, he is working as Graduate Research Assistant in Software Engineering Research Lab (SERL) at LUMS. His Ph.D. research work is focused on achieving self-managing behavior in autonomic systems using case-based reasoning. He is working on improvement of diagnosis and planning phases of autonomic managers in terms of efficiency as well as accuracy. His other research interests include application of artificial intelligence techniques in software engineering domain.

Mian Muhammad Awais is associate professor at Department of Computer Science, LUMS, Lahore, Pakistan. Dr. Awais received his Ph.D. in 2000 from Imperial College, University of London. Prior to joining LUMS, Dr. Awais conducted European Union research and development projects for a UK based SME. His Ph.D. work related to the development of online models for parametric estimation of solid fuel-fired industrial boilers. Dr. Awais has also conducted research work on a class of iterative methods pertinent to Krylov subspaces for optimization, such as the oblique projection and implicitly restarted model reduction methodologies. His current research interests include development of application of artificial intelligence techniques to model and control intricate phenomenological processes, imitative learning in robots, evolutionary computing, inverse modeling, autonomic computing, case-based reasoning and data management.

Shafay Shamail is associate professor at Department of Computer Science, LUMS, Lahore, Pakistan. He completed his BSc Electrical Engineering from UET Lahore in 1985, MSc Electronics from University of Wales in 1987, and Ph.D. degree in Electrical Engineering from University of Bath U.K in 1990. Before joining LUMS Dr. Shamail worked in both the Software industry as well as the academia. While working in software industry, he gained experience in e-commerce technologies, especially those from Microsoft. Dr. Shamail has a vast experience of curriculum design and implementation. He is a member of IEEE. His research interests include software engineering, e-government, software processes and autonomic computing.