Faster Case Retrieval Using Hash Indexing Technique

Mohamad Farhan Mohamad Mohsin *farhan@uum.edu.my*
College of Arts & Sciences
Universiti Utara Malaysia
Kedah, 06010, Malaysia

Maznie Manaf *maznie@kelantan.uitm.edu.my*
Faculty of Computer Science & Mathematic
Universiti Teknologi Mara (Kelantan)
Kelantan, 18500, Malaysia

Norita Md Norwawi *norita@usim.edu.my*
Faculty of Science & Technology
Universiti Sains Islam Malaysia
71800, Nilai, Negeri Sembilan, Malaysia

Mohd Helmy Abd Wahab *helmy@uthm.edu.my*
Faculty of Engineering and Electrical Engineering
Universiti Tun Hussain Onn
Johor, 86400, Malaysia

Abstract

The main objective of case retrieval is to scan and to map the most similar old cases in case base with a new problem. Beside accurateness, the time taken to retrieve case is also important. With the increasing number of cases in case base, the retrieval task is becoming more challenging where faster retrieval time and good accuracy are the main aim. Traditionally, sequential indexing method has been applied to search for possible cases in case base. This technique worked fast when the number of cases is small but requires more time to retrieve when the number of data in case base grows. As an alternative, this paper presents the integration of hashing indexing technique in case retrieval to mine large cases and speed up the retrieval time. Hashing indexing searches a record by determining the index using only an entry's search key without traversing all records. To test the proposed method, real data namely Timah Tasoh Dam operational dataset, which is temporal in nature that represents the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia ranging from year 1997-2005, was chosen as experiment. Then, the hashing indexing performance is compared with sequential method in term of retrieval time and accuracy. The finding indicates that hashing indexing is more accurate and faster than sequential approach in retrieving cases. Besides that, the combination of hashing search key x produces better result compared to single search key.

Keywords: Hashing Indexing, Sequential Indexing, Case Retrieval, Case Base Reasoning.

1. INTRODUCTION

Case-based reasoning (CBR) is a model of reasoning that mimics a human deal with unseen problem. It focuses on the human problem solving approach such as how people learn new skill and generates solution about new situations based on their past experience. Similar mechanism to human that intelligently adapts his experience for learning, CBR replicates the processes by considering experiences as set of old cases and problem to be solved as a new case. To derive to a conclusion, it executes four steps that are retrieve the most similar cases, reuse the retrieved cases to solve the problem, revise the reused solution, and finally retain the revised experience in case base for future decision making. Figure 1 illustrates the CBR decision making processes.

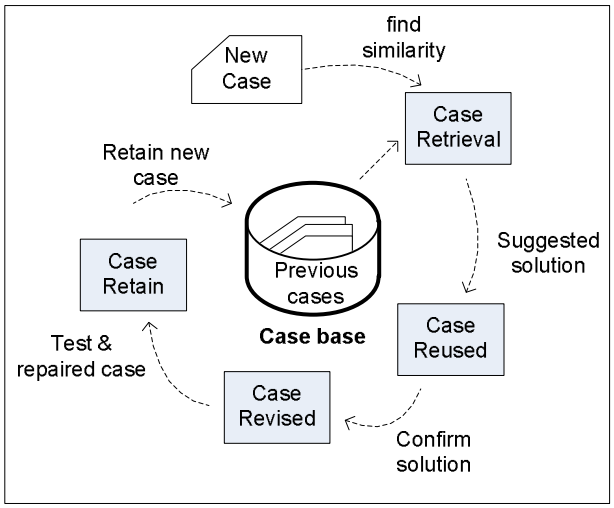

FIGURE 1: The CBR decision making processes [13]

Since it was introduced back in 1970, CBR has had a significant impact to many domains. For example, the technique is widespread across in biology [1], medical for diagnostic and therapeutic task [2], treatment [3], image retrieval [6, 12], project management and planning [7], education and tutoring [8]. The advantages of CBR such as flexibility in knowledge modeling that offers incremental case learning has made possible for CBR to be applied to extremely diverse application domains. Due to the complexity of problem, CBR also has been integrated with soft computing technique such as fuzzy logic [9], neural network [10], and genetic algorithm [11].

Theoretically, CBR maps the similarity between old and new case to derive conclusion. Therefore, the number of old cases is important to lead CBR in producing good decision [3]. It relies heavily on the quality of old cases but practically, to obtain a quality case is difficult to come by [4], [5]. Nowadays, CBR has capability to store million cases in case base due to the advance of data storage technology. With a parallel moving to that scenario, many researchers have undertaken study on case retrieval mainly on the case indexing technique for faster retrieval time. The selection of indexing type is important because it permits the system to match right case at the right time [13].

In general, there are two types of indexing structures which are sequential and non-sequential indexing. Sequential indexing- a conventional technique which has been applied to search for possible cases in case base. Through sequential technique, cases are retrieved case by case following a sequence until the most similar case is matched. It works fast when the number of cases is small but the problem arises when the number of cases contain in case base is huge which consume more time to retrieve.

In this study, a new approach for case indexing in CBR is proposed. This study researches the non-sequential indexing called hashing as an alternative to cater large cases and achieve faster retrieval time in CBR. Hashing indexing searches a record by determining the index using only an entry's search key without traveling to all records [14]. It utilizes small memory, faster retrieval time, and easier to code compared to other indexing technique like data structure [15]. This paper presents the review of the literature of both indexing methods and the integration of hashing indexing in case retrieval with the aim to improve the retrieval performance. To test the proposed method, a real data on Timah Tasoh Dam daily operation was chosen as an experiment. The dataset is a temporal data representing the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia in the year 1997-2005. Then, the hashing indexing performance is compared with sequential method in term of retrieval time and accuracy.

This paper is organized as follows. Section 2 outlines the literature of case retrieval and hashing indexing. Then, the integration of hashing indexing technique in CBR is discussed in section 3. It will be followed by a discussion on the research design of the study in Section 4. Section 5 describes the experiment data used in this study. In Section 6, the finding and result of the study will be presented and final sections conclude this work.

2. CASE RETRIEVAL AND HASHING INDEXING

Decision making in CBR starts with the case retrieval. It involves the process of finding possible cases in case base that are closest to the new case. For a given new case $\hat{C}=\{x_1, x_2, \ldots, x_n, \theta\}$, where θ is the decision to be determined. The case retrieval is the process of finding old cases C that are close to $\hat{C}$. The mapping of C and $\hat{C}$ is represented as $(\hat{C}, C)$ where cases, $C = \{C_1, C_2, C_3, \ldots, C_n\}$ and $\hat{C}$ is a query case. The similarity between both cases will be determined $(\hat{C} \epsilon C)$ based on the similarity, $Sim = (\hat{C}|C)$.

Two important criterion need to be determined for a quality case retrieval. Firstly, the mechanism to control how the case base is searched and secondly, the suitable search key x to guide searching [13]. In reality, the case retrieval process is highly exploited computer memory and time consuming due to the searching process in huge case base. Therefore, the case indexing technique plays a very important role to determine the searching process either search for an entire case or portion of it. According to [16], indexing and database representation is a fundamental problem for efficient clustering, classification, and data retrieval. The main concern in case retrieval in CBR is how to assess the similarity between cases in order to retrieve appropriate precedents and how to adapt old solution to the case [17]. Beside the most similar, the minimum time consume during the process is also important.

One of the indexing technique uses in case retrieval is sequential indexing. It is a conventional approach applied in the early database technology which cases are retrieved case by case following a sequence until the most similar case is matched. Since it scans the case base following a sequence, this method is not efficient when the number of cases in case base is huge which consume more time to retrieve. As a solution, hashing indexing method with search key x is proposed in database technology.

2.1 **Hashing Indexing**

Hashing indexing is commonly used in database application during data retrieval. This technique has been developed to access large files residing on external storage, not just for accessing fixed size files but flies that grow over their time. The idea of hashing is to map each records to a hash key x using hash function; $H(x)$. $H(x) = i$ whereby i is an index to the hash table, HT and $HT = [1, \ldots, N]$. HT represents an array of size n. The $H(x)$ will take a search key and produces an integer type index representing each case in HT. After that, the case can be directly retrieved at respective address from the HT. The address or search key, x is generated from the function $H(x) = x \ mod \ n$ whereby x is the search key, n is the table size and mod is the modulo operator.

The efficiency of $H(x)$ can be seen in memory management. The approach requires substantial less amount of memory as well as easier to code compared to tree data structure in traditional indexing approach. It also works without requiring a complete file organization and rehashing [15].

In practice, $H(x)$ can map two or more addresses into HT. The $H(x)$ is capable to store more than two items in the same array location in HT or tends to open other address. This occurrence is called collision and the item involved are often called as synonyms [18]. An example of hashing indexing technique which is adopted from [19] is shown in Figure 2.

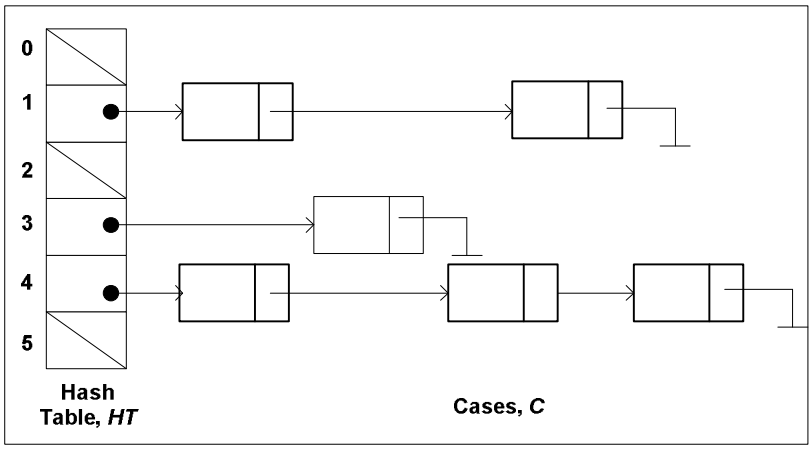

FIGURE 2: Hashing Indexing Technique [19]

One of the HT limitations is when the records become full. It will start working very badly unless separate chaining which is capable to handle collision is used. This is the reason why [18] suggested that HT should never be allowed to get full. To determine either HT is full, the ratio of the number entry located in HT need to be calculated. The ratio is known as load factor. Generally, HT size should be automatically increased and the records in the table should be rehashed when the ratio of table is reached 0.7 (70% full) or 0.8 (80% full) for open addressing [14, 18].

Recently, many applications utilized hashing mechanism to solve specific problem such as in programming that uses HT to keep track of declared variables in source code [14, 18, 19]. HT is an ideal application for this kind of problem because only two operations are performed: insert and find; identifiers are typically short, so the $H(x)$ can be computed quickly. In this application, most searches are successful.

Another common use of HT is in game programs. As the program search through different lines of play, it keeps track of positions that it has encountered by computing $H(x)$ based on the position (and storing its move for the position). If the same position recurs, usually by a simple transposition of moves, the program can avoid expensive recalculation.

3. THE MERGING OF HASHING INDEXING IN CASE RETRIVEVAL

The advantages of hashing indexing in data retrieval are faster retrieval time and minimize the usage of computer resources. This motivation has lead to the merging of hashing indexing in CBR since case retrieval requires a fast solution to retrieve case from case base. Figure 3(a) depicts the concept of this technique and Figure 3(b) is a sequential indexing method. Sequential indexing is a conventional technique practiced in CBR's case retrieval.

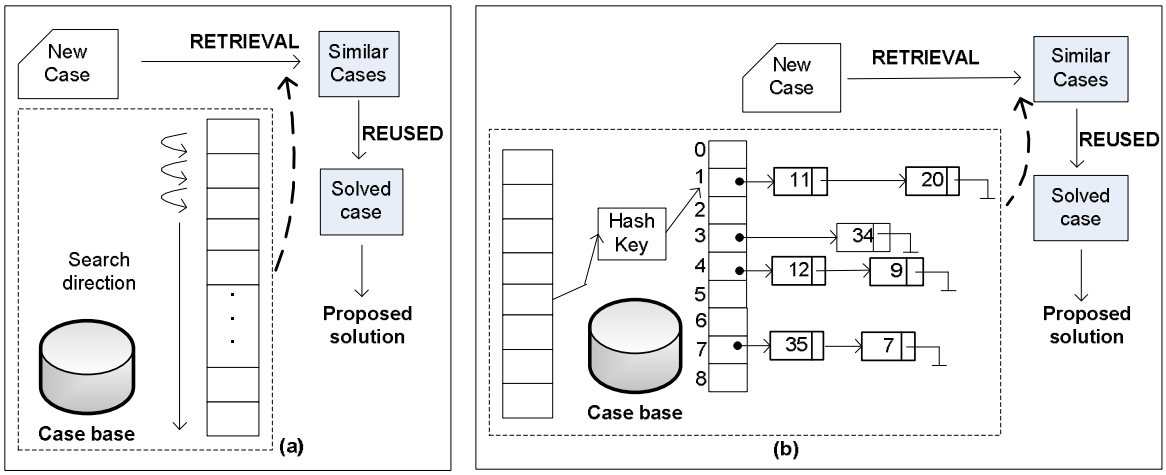

FIGURE 3 (a): The Sequential Indexing Method and **FIGURE 3 (b):** The Hashing Indexing Method

A good $H(x)$ is should be fast to compute, minimize collisions, and distribute entries uniformly through the HT. In the proposed hash model, the separate chaining or close addressing is chosen to resolve collisions. Through this method, specific location is allowed to store more than one value called bucket, b. A new x map or address can be simply placed into a particular location and associated value placed all the cases which have the related attribute in b. The Figure 4 summarizes the separated cases location in hash table, HT illustratively.

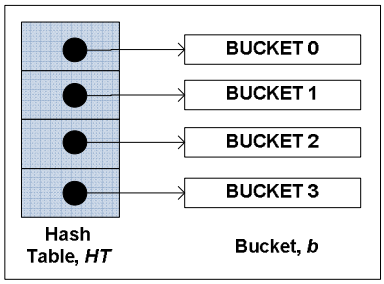

FIGURE 4: The Separated Cases Location in *HT*

The modified hashing indexing algorithm in CBR involves two main tasks that are storing new cases and retrieving a case. Process flow in Figure 5 represents the process of storing a new case in case base. It starts with calculation of $H(x)$ to determine b location at hash table and then store the current cases in b. The algorithm of storing a case into hash table is shown in Algorithm 1

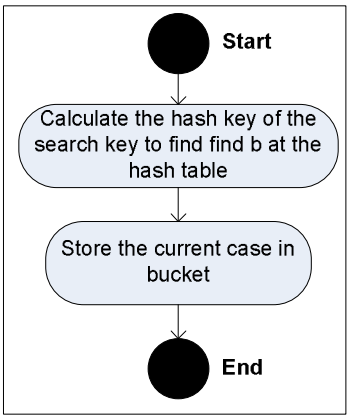

FIGURE 5: Storing a Case into Case Base

Algorithm 1: **Storing a case into hash table**

Input: Timah Tasoh Dam Dataset; search key x; size of data n; size of attribute m; the selection attribute to calculate $selectionColumn$; bucket quantity b, range of search key r

```
1 START
2    get data from dataset
3    store in array oldCases [n][m]
4    set int i = 0, j = 0, counter = 0;
5    WHILE (i = 0)
6        read oldCases[i][selectionColumn]
7        calculate x
8        WHILE (counter << b)
9           determine the range of r
11          store oldCases[i][j] in array bucket_counter [n][m];
12          counter ++
13       ENDWHILE
14      i ++
15   ENDWHILE
16 END
```

The case retrieving process based on CBR's hashing indexing is shown in Figure 6. It starts with calculating the hash key and map to the *HT*. The result is retrieved by finding the search key x after entering a new case. The $H(x)$ formula is used to find the address in HT. Finally, the similarity of cases in similar bucket is calculated to get the predicted result. The similarity of the cases is calculated based on the local similarity and global similarity. Equation 1 and 2 displays the calculation of both similarities.

$$sim(\hat{c}|c) = \frac{\hat{c}-c}{range} \quad (1)$$

where $sim(\hat{c}|c)$ is local similarity, $\hat{c}$is a new case and c is an old case

$$simG(\hat{C}|C) = \sum_{i=1}^{p} w_i sim_i(\hat{c_i}, c_i) \quad (2)$$

where $sim(\hat{C}|C)$ is global similarity, $\hat{C}$is a new case and C is an old case, p is total cases in case base, $sim_i(\hat{c_i}, c_i)$ the local similarity calculate the attribute i, and w_i is the weight of the attribute i

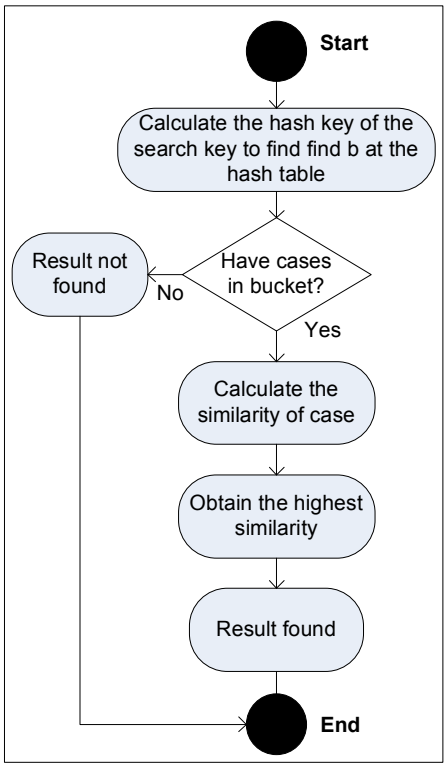

FIGURE 6: Retrieving a Case from *HT*

In this study, three search keys, x are defined. The x are mean of average rainfalls $(\overline{R_m})$, change water level (ΔWL), and combining mean average rainfall and change water level $(\overline{R_m} \wedge \Delta WL)$ which are considered as $H(x)$ as written in (2). Different x are used to determine which x will produces better result mainly in high accuracy and low time retrieval. The x represents the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia in the year 1997-2005. Next section will describes this data set in detail.

$$H(x) = \begin{cases} \overline{R_m} \ mod \ n \\ \Delta WL \\ \overline{R_m} \wedge \Delta WL \end{cases} \quad (3)$$

Where n is the table size, Mod is the modulor operator, $\overline{R_m}$ refer to Equation 3, ΔWLrefer to Equation 4.

To calculate search key,

$$\overline{R_m} = \frac{\overline{R_{t-1}} + \overline{R}}{2} \quad (4)$$

Where $\overline{R}$is the average rainfall at time t , $\overline{R}$is the average rainfall at time *t -2*, *t* is the time index

and to calculate Δ search key

$$\Delta WL_t = \frac{WL_{t-1} + W.}{WL_{t-2}} \quad (5)$$

Where $WL$is the average rainfall at time $t-1, Wl$ is the average rainfall at time t , *t* is the time index

Every types of x will have different size of hash table or called bucket, b. The number of b will depends on the type of its x. For example, the change of water level (ΔWL) has three types of water level, which are Alert, Warning and Danger [15]. Therefore, ΔWL has three buckets. Table 1 shows the ΔWL key, the number of bucket, and the range of case ΔWL. From Table I, Figure 7 represents the bucket arrangement of ΔWL.

TABLE 1: *Type of ΔWL and The Number of b*

Search key: ΔWL		
b	**Type of water level**	**Range of / m**
0	Alert	$x \leq 0.0034$
1	Warning	$0.0034 < x < 0.0061$
2	Danger	$x \geq 0.0061$

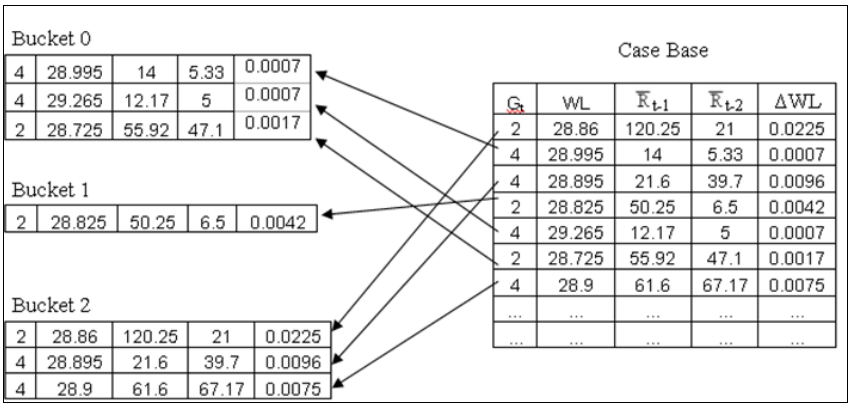

FIGURE 7: The b Arrangement Using ΔWL Key

For mean of average rainfall m key, it has four buckets which represent type of rainfall that are Light, Moderate, Heavy and Very Heavy. Table 2 elaborates the type of rainfall while Figure 8 illustratively represents the bucket arrangement of m key. The Figure 9 portrays the total number of b for the combination of m and ΔWL as thirds search key

TABLE 2: Type of Rainfall and The Number of b

Search key:		
b	**Type of Rainfall**	**Range of Rainfall / mm**
0	Light	$x \leq 11$
1	Moderate	$11 < x < 32$
2	Heavy	$32 < x < 62$
3	Very Heavy	$x \geq 62$

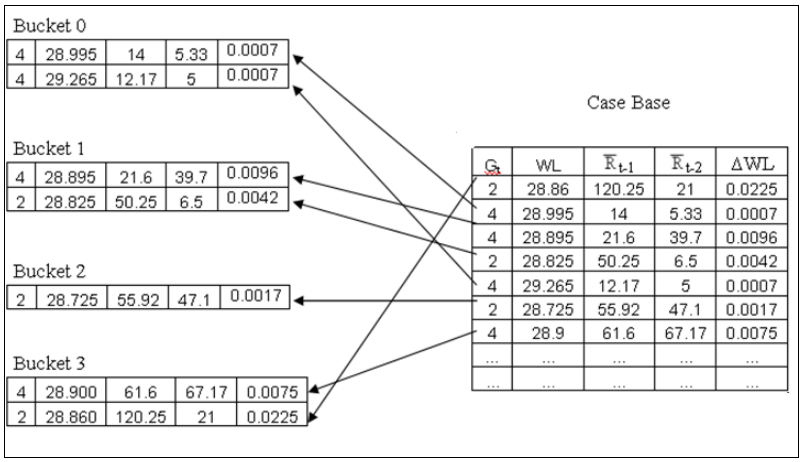

FIGURE 8: The b arrangement using $\overline{R_m}$ key

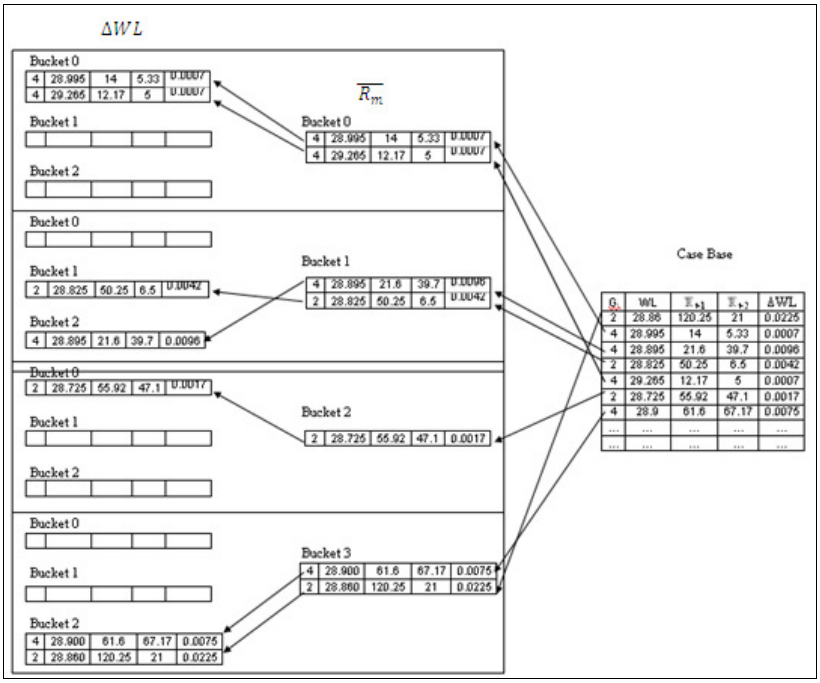

FIGURE 9: The b arrangement using $\overline{R_m}$^ ΔWL key

4. RESEARCH DESIGN

This section describes the research design used in this study which is illustrated in Figure 10. There are three phases which start with development, then preparing data for mining, and lastly is Case Mining.

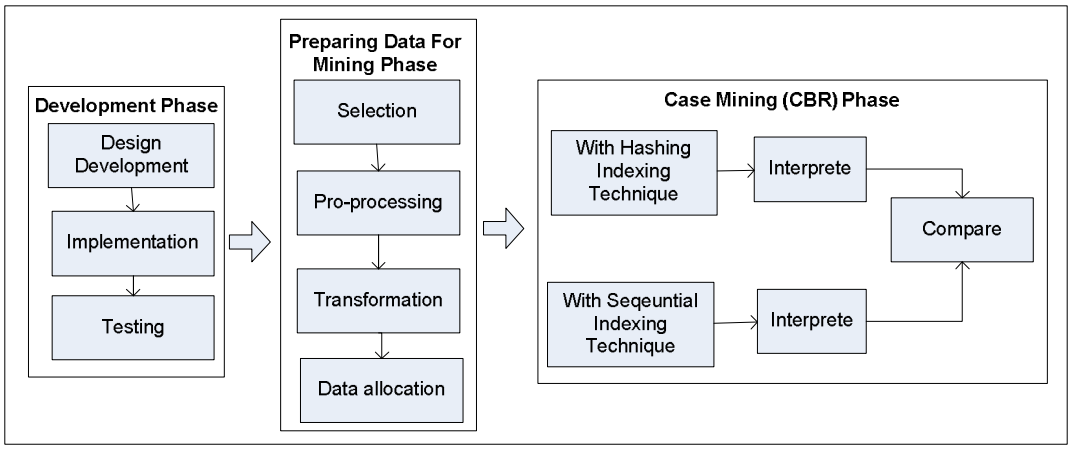

FIGURE 10: The Research Design

The development phase focuses on the algorithm modification. This phase covers three steps which are design development, implementation and testing. In the design development, two approaches: sequential indexing and hashing indexing technique are designed and integrated into CBR using Microsoft Visual C++. After that, the model will be tested. The aim of the testing is to check the accurateness of the hash table and the similarity calculation during mining.

The second phase is preparing data for mining which includes four activities – selection, pre-processing, transformation, and data partition. The aim of this process is to clean and prepare the Timah Tasoh Dam dataset before presenting into the CBR mining system. The selection, pre-processing, and data transformation process are explained in section 5. In data allocation, the experiment data is divided into five folds with different set of training and testing data allocation. The multiple folds are used for a variation set of result. The folds (training: testing) are 90:10, 80:20, 70:30, 60:40 and 50:50.

The last phase is case mining. It involves the mining of TImah Tasoh Dam data set with both indexing methods. During experiment, two measurement metrics are recorded that are accuracy and retrieval time. Then their results are compared. In order to measure the accuracy, the algorithm is tested using various data partition by taking cases in case based as a test set. The measurements are adopted from [15]. This is due to the fact that the real datasets consists of unbalanced data where the number of occurrences of event is lower as compared to non-event occurrence. The accuracy of the model is evaluated base on Equation 6.

$$acc\ (\%) = \frac{t_p + t_n}{t_n + f_n + t_p + f_p} \quad (6)$$

Where t_p is the number of event correctly predicted, f_p is the number of predicted event but in actual non-even, t_n is the number of non-event correctly predicted, and f_n the number of predicted non-even but in actual even

Second measurement is retrieval time which refers to time taken to search for the similarity case from case base. The time is tested by selecting one case from case base and the selected case will be measured for both hashing and sequential technique. The retrieval time will be recorded five times before calculate the average. A special loop is used to perform the task as shown in coding in Figure 11.

```
Double start = clock();

//dummy looping
for (i=0; i<900000; i++){
      for(i=0; i<column; i++){
            for (j=0; j<row; j++){
                  oldCases[i][j];
}}}

/****************************************
    CBR ENGINE (get the global similarity)
*/

Double end = clock();

//calculate retrieval time
cout << Time Taken :: " << (end-start)/60 << millisecond
```

FIGURE 11: Algorithm to Calculate Retrieval Time

5. TIMAH TASOH DAM DATASET

The experiment dataset set used in this study has 15 attributes of temporal data called Timah Tasoh Dam dataset. It comprises the historical hydrological data of daily Timah Tasoh dam operation in Perlis, Malaysia in the year 1997-2005. The preliminary observation on the raw dataset, found out that some attributes are not related to study and certain values were missing. Therefore, the dataset are pre-processed using temporal data series approach which was adopted from [14,15].

During data preprocessing, only relevant attributes were selected. Out of 15 attributes, 4 attributes were chosen that are current water level, average rainfall, current change water level, and current gate. Those attributes represents reservoir water level, rainfall measurement from 6 telemetry stations (Padang Besar, Tasoh, Lubuk Sireh, Kaki Bukit, Wang Kelian, and Guar Jentik) and the number of spillway gates. Spillway gate refers to a structure that makes it possible for the excess water to be released from the dam. Timah Tasoh has six gates and normally the water will be released using Gate 2, Gate 4, and Gate 6 depending on the situation. The selection is made using sliding window technique which is adopted from [14, 15]. After that, the data are re-scaled into a suitable representation to increase mining speed and minimize memory allocation.

Table 3 is a sample of clean data which is ready for mining using CBR's hashing indexing and CBR's sequential indexing model. Based on the table, the current water level (*WL*), average rainfall at $t-1$and $t-2$, current change water level (ΔWL) and current gate (*GT*) are the final input to be mined using CBR.

TABLE3: A Sample of Timah Tasoh Dam Dataset After Pre-process

GT	*WL*	*Average Rainfall* $t-1$	*Average Rainfall* $t-2$	ΔWL
2	29.275	7.33	5.375	0.0007
2	29.025	22.75	11	0.001
4	28.9	61.6	67.17	0.0075
4	28.895	21.6	39.7	0.0096
4	28.995	14	5.33	0.0007
2	29.32	17.5	32	0.0057

6. RESULT & FINDING

This section reports the finding of the integration of hashing indexing technique in case retrieval. The tested model was compared with case retrieval function embedded with sequential indexing technique. As elaborates in 4, the evaluation is conducted using two criteria that are accurateness of the model to obtain similar cases and how fast it takes to retrieve cases. The notation of the experiment is given as follows: The accuracy of the mining as **%**, and retrieval time in millisecond is ***Ms***, The result of the experiment is visually represented in Table 3.

TABLE 3: The Mining Result of Hashing and Sequential Indexing Technique in Ms and %

Data Partition	Sequential Indexing Technique		Hashing Indexing Technique (Search Key *x*)					
			$\overline{R}_m$		ΔWL		$\overline{R}_m \wedge \Delta WL$	
	Ms	%	Ms	%	Ms	%	Ms	%
90 : 10	15.27	50	15.09	75	14.36	50	13.96	75
80 : 20	12.03	38	11.68	38	11.09	50	10.41	57
70 : 30	10.31	46	10.26	38	10.02	46	9.95	42
60 : 40	9.85	47	9.74	35	9.02	41	8.96	50
50 : 50	8.69	38	8.64	38	8.49	52	8.02	61

The analysis starts with the retrieval time of both methods. The result indicates that hashing indexing method required less time for case retrieval in all experiments. For example, in the fold 60:40, sequential technique needs 9.85 *ms* to map all cases however the time taken are lesser in hashing indexing technique with different search key ($\overline{R_m}$= 9.74 *ms*, ΔWL= 9.02 *ms*, $\overline{R_m} \wedge \Delta WL$= 8.96 *ms*). Moreover, the finding also reveals the combination of hashing search key $\overline{R_m} \wedge \Delta WL$ is looked as the most efficient key to mine cases faster compared to single search key. The graph in figure 12 summarizes illustratively the retrieval time taken of both methods and figure 13 shows the retrieval time taken in 60:40 fold as discussed in this paragraph.

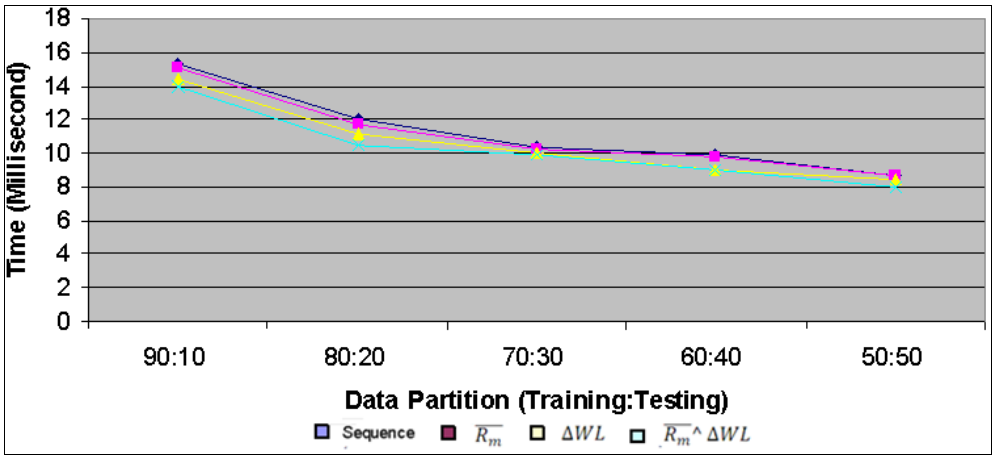

FIGURE 12: The Retrieval Time Taken Hashing and Sequential Indexing

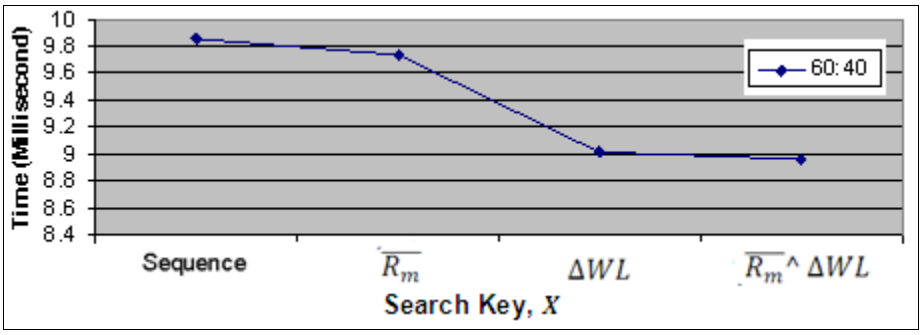

FIGURE 13: Retrieval Time Taken in 60:40 fold

Then, the accurateness of CBR to predict new case is evaluated. In this analysis, the CBR modeling with hashing indexing technique leads the high accuracy. The graph in figure 14 summarizes the accuracy of both methods. Similar in time retrieval evaluation, the $\overline{R_m}$^ ΔWL search key is out performed the single search key $\overline{R_m}$ and ΔWL. It consistently obtains high accuracy in all folds except in 70:30. Interestingly, the result also indicates that the sequential indexing technique also capable to obtain good accuracy when overcome the hash indexing in 70:30 with 46% accurate and left behind the $\overline{R_m}$ (38%) and $\overline{R_m}$^ ΔWL (42%).

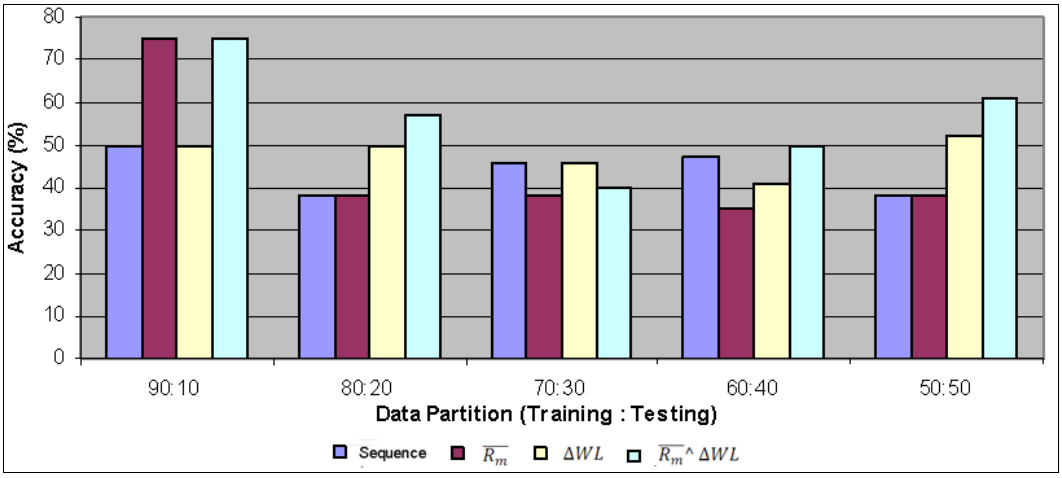

FIGURE 14: The Accuracy of Hashing and Sequential Indexing

Table 4 below summarizes the best technique of the whole experiments. The best technique is selected based on the highest accuracy and shortest time taken to mine Timah Tasoh Dam Dataset. From the table, it is clearly indicates that hashing indexing method has retrieved cases faster that sequential with the combination search key $\overline{R_m}$^ ΔWL as the best search key. In term of accuracy, hashing indexing has scored higher then sequential technique. Out of 5 folds, hashing indexing obtain better accuracies in 4 folds except in fold 70:30, the sequential indexing generates similar accuracy with ΔWL. Lastly, the combination search key $\overline{R_m}$^ ΔWL is chosen as the best search key due to is capability to generate high accuracy and retrieve case faster for Timah Tasoh Dam Dataset.

TABLE 4: The Summarization of the Best Technique based on Accuracy and Retrieval Time

Data Partition	Performance measurement metrics	
Setting	Accuracy	Case Retrieval Time
90 : 10	$\overline{R_m}$ and $\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL
80 : 20	$\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL
70 : 30	Sequence **and** ΔWL.	$\overline{R_m}$ ^ ΔWL
60 : 40	$\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL
50 : 50	$\overline{R_m}$ ^ ΔWL	$\overline{R_m}$ ^ ΔWL

7. CONCLUSION

This research integrates the hashing indexing technique in case retrieval with the aim to cater large cases stored in case base and faster retrieval time. Its performance is compared with the sequential indexing technique using two criteria that are accuracy and retrieval time. From the experiment towards temporal dataset called Timah Tasoh Dam, the hashing indexing is more accurate and faster than sequential in retrieving cases. The finding of this study offers an alternative technique for case base representation and case retrieval. The finding also can assist future miner to mine cases faster, obtain better accuracy and minimize the computer resources usage. For future study, the case retrieval with hashing indexing approach will be tested with other type of data from various domains.

8. REFERENCES

[1] I. Jurisica, and J.I. Glasgow. “Applications of case-based reasoning in molecular biology”. AI Magazine, American Association for Artificial Intelligence, vol. 25(1), pp. 85-95, 2004.

[2] R. Schmid, and L. Gleri. “Case-based Reasoning for Medical Knowledge-based Systems”. International Journal of Medical Informatics, vol. 64, pp. 355, 2000.

[3] Yang, Z., Matsumura, Y., Kuwata, S., Kusuoka, H., and Takeda, H. “Similar Cases Retrieval From the Database of Laboratory Test Results”. Journal of medical systems (J. med. syst.), vol 27, pp. 271-282, 2003.

[4] E. Armengol, S. Ontanon, and E. Plaza. “Explaining Similarity in CBR”. Artificial Intelligence Review. Vol. 24, 2002

[5] P. Rong, Q.Yang, and J.P. Sinno**.** “Mining Competent Case Bases for Case-Based Reasoning”. Journal Artificial Intelligence, vol. 171, 2007.

[6] D.O. Sullivan, E. McLoughlin, B. Michela, and D.C. Wilson. *“*Capturing and reusing case-based context for image retrieval,*”* In Proc. of the 19th International Joint Conference on Artificial Intelligence, 2005.

[7] M.Emilia, N. Mosley, and C. Steve. “The Application of Case-Based Reasoning to Early Web Project Cost Estimation,” In Proc. of the 26 the Annual International Computer Software and Applications Conference (COMPSAC’02), 2002.

[8] K.S. Leen, and B. Todd. “Integrating Case-Based Reasoning and Meta-Learning for a Self-Improving Intelligent Tutoring System”. International Journal of Artificial Intelligence in Education table of contents archive, vol. 18(1):27-58, 2008.

[9] C.K.P. Wong. “Web access path prediction using fuzzy-cased based reasoning, Phd Thesis, Hong Kong Polytechnic University, Hong Kong, 2003.

[10] J.M. Corchodo and B. Lees. "Adaption of cases for case-base forcasting with neural network support," in Soft computing in case based reasoning, 1st ed., vol.1. S.K.Pal, S.D.Tharam, and D.S. Yeung, ed. London: Springer-Verlag, 2001, 293-320.

[11] K.S. Shin and I.Han. "Case-based reasoning supported by genetic algorithm for corporate bond rating". Expert system with application, vol. 1266, pg.1-12. 1997.

[12] H. Hamza, Y. Belaid, and A. Belaid. "A case-based reasoning approach for unknown class Invoice Processing," in Proc. of the IEEE International Conference on Image Processing, (ICIP), 2007, pp. 353-356.

[13] K.P. Sankar and K.S. Simon. Foundation of Soft Case-Based Reasoning, John Willey & Sons Inc, 2004, pp. 1-32.

[14] F. M. Carrano, and W. Savitch. Data Structures and Abstractions with Java. USA: Pearson Education, 2003.

[15] M. Griebel and G. Zumbusch. "Hash-Storage Techniques for Adaptive Multilevel Solvers and Their Domain Decomposition Parallelization". In Proc. of Domain Decomposition Methods 10 (DD10), 1998.

[16] X. He, D. Cai, H. Liu, and W. Ma. "Locality Preserving Indexing for Document Representation," in Proc. of the 27th conference on research and development in information retrieval, 2004.

[17] E. Armengol, S. Ontanon, and E. Plaza. "Explaining Similarity in CBR". Artificial Intelligence Review. vol. 24(2), 2004.

[18] W. D. Maurer, and T.G. Lewis. "Hash Table Methods". ACM Computing Surveys (CSUR), vol 1, pp. 5-19, 1975.

[19] N.M. Darus, Y. Yusof, H. Mohd, and F. Baharom. "Struktur data dan algoritma menggunakan java". Selangor, Malaysia: Pearson Prentice Hall, vol. 1, 2003.