Computational Intelligence, Volume 17, Number 2, 2001

A CASE-ADDITION POLICY FOR CASE-BASE MAINTENANCE

QIANG YANG AND JUN ZHU

School of Computing Science, Simon Fraser University

A major problem in many practical applications of case-based reasoning (CBR) and knowledge reuse is how to keep the case bases concise and complete. To solve this problem requires repeated maintenance operations to be applied to case bases. Different maintenance policies may result in case bases with very different quality. In this article, we present a case-addition maintenance policy that is guaranteed to return a concise case base with good coverage quality. We demonstrate that the coverage of the case base computed by the case-addition algorithm is no worse than the optimal case-base coverage by a fixed lower bound. We also show that the algorithm implementing the case-addition policy is efficient. Our result also highlights benefit reduction as a key factor in influencing the convergence of case-base coverage when cases are added to a case base. Through our theoretical analysis, we analytically derive the well known coverage convergence curves commonly displayed in CBR experiments and show that benefit reduction can be used as a predictor for convergence speed.

Key words: case-based reasoning; case-base maintenance; theoretical foundations of case-based reasoning; case-base maintenance policies.

1. INTRODUCTION

Case-base maintenance (CBM) refers to the task of adding, deleting, and updating cases, indexes and other knowledge in a case base in order to guarantee the ongoing performance of a case-based reasoning (CBR) system. CBM is particularly important when a CBR system becomes a critical problem-solving system for an organization. In many practical applications, new cases arrive at a very fast rate, resulting in a case base that contains many redundant cases. As a result, the sizes of case bases often increase with time, creating significant barriers to the efficiency of reasoning and to the users' ability to understand the results.

In response to these problems, there has been a significant increase in CBM research. One branch of research has focused on the ongoing maintenance of case-base indexes through training and case-base usage (Cunningham, Bonzano, and Smyth 1997; Fox and Leake 1995; Aha and Breslow 1997; Zhang and Yang 1999). Another branch of research has considered preserving the overall competence of a case base while performing CBM (Smyth and Keane 1995; Markovich and Scott 1998; Domingos 1995; Aha, Kibler, and Albert 1991; Racine and Yang 1997;). The aim of the research is similar to the utility-based control-rule deletion policies (Minton 1990). Excellent surveys of this field can be found in Leake and Wilson (1998) and Watson (1997).

This recent surge of interest in CBM is highlighted by Smyth and Keane's (1995) seminal work on competence-preserving case-deletion policies. In this work, the cases in a case base are classified into a type hierarchy based on their coverage potential and adaptation power. The deletion policy then selectively deletes cases from a case base guided by the classification of the cases until a limit on the case-base size is reached. The algorithm was shown empirically to preserve the competency of a CBR system and to outperform a number of previous deletion-based strategies.

In this article, we present a different CBM policy that is based on case addition rather than deletion. By this policy, cases in an original case base are repeatedly selected

Address correspondence to Qiang Yang, School of Computing Science, Simon Fraser University, Burnby, British Columbia, Canada V5A 1S6.

and *added* to an empty case base until a certain size limit is reached, producing an updated case base with high coverage. The addition-based policy will allow a more global view of the case base as a result of comparisons of the benefits of cases when they are added to a case base. We show that both Smyth and Keane's deletion-based policies and our addition-based policies have the same time complexity. On top of this, the advantage of the addition-based policy is that we can place a lower bound on the coverage of the resulting case base; we demonstrate that the coverage of the computed case base cannot be worse than the optimal case base in coverage by a fixed lower bound and often is much closer to the optimal coverage. In the process of derivation, we also highlight an important factor in CBM: the concept of benefit reduction. The benefit of an added case to a case base is the additional coverage it brings to the case base. We show that the faster the reduction in benefits as cases are added in, the higher is the coverage.

This article is organized as follows. In Section 2 we define the terminology in CBM work and present some key observations that motivate our case-addition policy. In Section 3 we give a case-addition policy to build the case base and prove that it is at least 63 percent as good as the optimal coverage. In Section 4 we describe the relationship between coverage and case-base size. In Section 5 we describe experimental results. We conclude the article in Section 6.

2. CBM AND CASE-DELETION POLICIES

2.1. Related Work

Recently, there has been an intense interest from the CBR research community in the problem of CBM. Leake and Wilson (1998) gave an in-depth summary and analysis of this field. For our purpose, the problem of CBM is divided into two broad categories: maintaining the case-base indexes and maintaining the case-base contents. In case-base index maintenance, Cunningham, Bonzano, and Smyth (1997) present an introspective learning approach to learn adjusted case-base indexes by monitoring the run-time processes of a case-based reasoner. An extended approach is developed in Zhang and Yang (1999), where a layered architecture is adopted for representing case base indexes and a neural-network algorithm is adapted for maintaining the feature weights. Fox and Leake (1995) and Aha and Breslow (1997) consider case-base index-revision policies that improve the performance of a case base in response to events such as plan failures.

Researchers in case-base content maintenance are mainly concerned with the issue of *optimization*. Due to the large size of some case bases, it is necessary to delete cases as time goes by and when retrieval becomes increasingly expensive (Smyth and Keane 1995). This issue is called the *swamping problem*. The main strategy is to decide which cases to delete based on an adaptation structure. These strategies include a random deletion strategy as advocated by Markovich and Scott (1998) and more sophisticated deletion based on the frequency with which each case is retrieved as well as the matching cost (Minton 1990). The problem with both these approaches is that "important" cases can be deleted by mistake. Various approaches have been designed to address this problem. Domingos (1995) and Aha, Kibler, and Albert (1991) consider instance-based learning approaches for reducing the size of a case base without decreasing its problem-solving power. Smyth and Keane (1995) consider a competence-preserving approach to case deletion. Watson (1997) presents methodologies for a human designer of a case

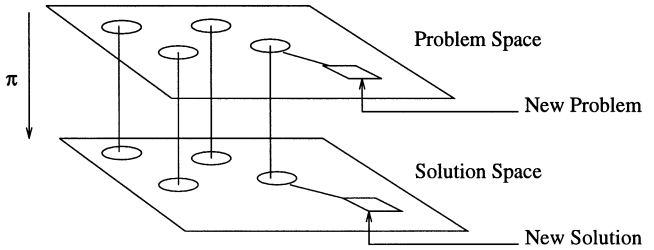

FIGURE 1. Illustrating Case-Based Reasoning (after Leake 1996).

base to consider for CBM. Racine and Yang (1996) consider the problem of removing redundancy and inconsistency from a large semistructured case base in order to improve the case-base performance.

2.2. Coverage and Neighborhood Functions

We define a *case* as a problem-solution pair. That is, each element x of a case base X is a pair $x = (p, s)$, where s is a solution to a problem p. As shown in Figure 1, in order to solve a new problem, a CBR system first finds a similar problem (represented by the circles on the top part of the figure) in the problem space. Then it finds a corresponding solution in a solution space (bottom of the picture). The solution is adapted to solve the new problem. It is therefore important for problem-solution pairs to be representative enough that most of the input problems can be solved by adapting a stored solution.

Let $N(x_1)$ be the set of cases x_2 whose solution $\pi(x_2)$ for x_2 is "close" to the solution $\pi(x_1)$ of x_1. More formally,

$$N(x_1) = \{x_2 \mid D[\pi(x_1), \pi(x_2)] \leq L\}$$

where $D[\pi(x_1), \pi(x_2)]$ represents the cost of adapting the solution of case x_1 in order to obtain the solution for case x_2, and L is a constant lower bound on distance. Essentially, $N(x_1)$ defines the cases that x_1 can "cover" through a small amount of adaptation on $\pi(x_1)$. We call $N(x_1)$ the *neighborhood* of x_1. Based on the function $D[\pi(x_1), \pi(x_2)]$, we will consider how to compute a new near-optimal case base X_1 from an input case base Z.

The neighborhood of a case base X_1, denoted by $N(X_1)$, is the union of the neighborhoods of all cases in X_1. In addition, given a case base X and a subset X_1 of X such that $N(X_1) \subseteq X$, the coverage of X_1 is defined as

$$\text{Coverage}\,(X_1) = \frac{|N(X_1)|}{|X|}$$

Finally, given a case base X, an *optimal case base of size k* is any subset of X of size k or less with the largest coverage. The purpose of this article is to find a near-optimal case base of size k efficiently.

3. CASE-ADDITION-BASED POLICY

Suppose that the neighborhoods of all cases in a case base are obtained. We say a case is *good* if its neighborhood is large. To select good cases, the distribution of the

cases or the frequency of cases occurring also should be considered. For instance, in a travel domain, suppose that more people prefer a travel plan between City 1 and City 2. We should put this plan in a case base in order to minimize the cost of searching for such plans. Taking this into account, we redefine case coverage as follows:

Suppose that we are given a problem domain with a case base X. X is the "universe" of our consideration, where the neighborhood of X is X itself. Let $x \in X$ be a case. As before, we denote $N(x)$ to be the neighborhood of x and $N(X_1) = \cup_{x \in X_1} N(x)$. $N(X_1)$ contains cases that are close to some other cases in X_1. Suppose that F is a frequency function of the cases, giving a total count of the cases' occurrence. Equivalently, there is a distribution of cases. The coverage of X_1 is defined as

$$\text{Coverage}(X_1) = \frac{\sum_{x \in N(X_1)} F(x)}{\sum_{x \in X} F(x)}$$

Since X_1 is a subset of X, the case coverage is a real number between 0 and 1. For simplicity of discussion, we will subsequently assume that $F = 1/|X|$, a constant function.

Let X be a case base and W be a subset of X. The benefit of a case $x \in X$ with respect to W is defined as $\text{Benefit}(x, W) = N(x) - N(W)$, where $N(W) = \cup_{x \in W} N(x)$. The benefit of a set of cases $\{x_1, x_2, \ldots, x_k\}$ is defined as the union of their benefits.

Suppose we want to build the case base X_1 with at most k cases based on a set X of cases. We formulate this optimization problem as follows:

> Choose cases $X_1 = \{x_1, x_2, \ldots, x_k\}$ from case base X to maximize the benefit of X_1 with respect to $W = \varnothing$.

A case base of size k that satisfies this criterion is called an *optimal case base*. A case base X_1 of size k is called a *near-optimal case base* if $\text{Coverage}(X_1) \geq C \times \text{Coverage}(Y)$, where Y is an optimal case base of size k and $C \leq 1$ is a positive constant.

Given a case base X, the problem of finding an optimal case base of a smaller size k is NP-complete. One can prove this by a reduction from set-covering (Garey and Johnson 1979). Thus we look for heuristics to find approximate solutions that can find a near-optimal case base.

To motivate our case-addition algorithm, consider the example case-base structure in Figure 2. In this figure, a directed arc from a case x to a case y denotes that y can be adapted from x within the adaptation-cost limit L. This implies that y is within the neighborhood of x. Likewise, cases a, b, and c are within the neighborhood of y. We observe that a maintenance policy should select cases based on the size of the neighborhood that each case has in a case base. Therefore, case y should be selected first into a new case base.

The decision to select the next case and insert it into a new case base is critical in determining the quality of a case base. For example, consider the case-base structure in Figure 3. Suppose that we have a case-base size limit of one. We observe that the case y provides the maximal "benefit" with respect to an empty case base. If we choose y in the new case base, four cases are covered. If, on the other hand, one of the x_i's is selected, then only two cases can be covered (x_i and y). If the case-base size limit is greater than one, then the selection for the next case should be based on what additional cases are covered by the new case. These additional cases are the "benefits" of the new case with respect to the case base that we have constructed so far.

We formalize the preceding discussions in the following case-addition algorithm. In the algorithm, we will select cases from an original case base X and add them to a new case base. The new case base X_1 is initially empty.

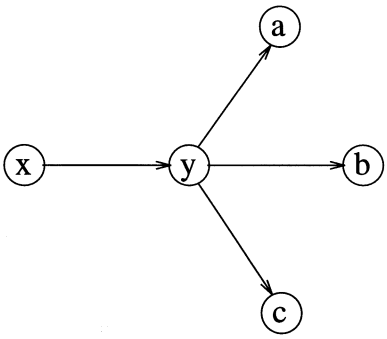

FIGURE 2. Case-Base Structure Graph.

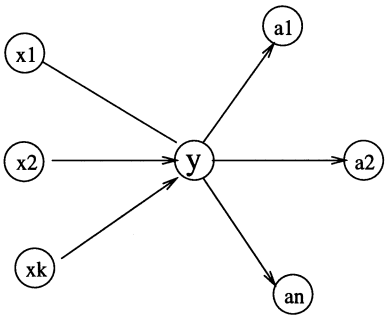

FIGURE 3. A Second Case-Base Structure Graph.

Case-Addition Algorithm

1. Determine the neighborhood $N(x)$ for every case $x \in X$.
2. Set $X_1 = \varnothing$.
3. Select a case from $X - X_1$ with the maximal benefit with respect to $N(X_1)$ and add it to X_1.
4. Repeat step 3 until $N(X) - N(X_1)$ is empty or X_1 has k elements.

Remark. Here we consider the case coverage as the benefit. In fact, the benefit can be defined on other notions as long as it captures the concept of usefulness.

The case-addition algorithm is a greedy algorithm. Therefore, it may not give the best choice of X_1 with respect to the case coverage. However, we can prove that its case coverage is at least 63 percent of the optimal case base for any fixed case-base size k. This makes the algorithm a near-optimal one.

Theorem 1. The case-addition algorithm produces a case base X_1 such that the coverage of X_1 is no less than 63 percent of the coverage of an optimal case base.

The proof for this theorem is similar to a proof given in Harinarayan, Rajaraman, and Ullman (1996) for constructing a data cube. In fact, we are inspired by that proof. In Harinarayan, Rajaraman, and Ullman (1996), the number of rows in a database view is used as the cost measure, and the benefit of a view is defined by how it can improve the cost of evaluating other views including itself. A greedy algorithm is given for selecting a set of k views to maximize the total benefit. Both proofs and algorithms have their roots in approximation algorithms for set covering.

We now give a sketch of the proof.

Sketch of the Proof. Let X_1 be a case base with k cases chosen from a case base X. First, if $N(X) - N(X_1) = \varnothing$, then X_1 has the maximal case coverage, so it is optimal. Hence we do not need to prove in this case.

Now suppose that $N(X) - N(X_1) \neq \varnothing$; then $X_1 = \{x_1, x_2, \cdots, x_k\}$ has k elements labeled by the order of selection. Let a_i be the benefit of $x_i, 1 \leq i \leq k$. Note that $a_1 \geq a_2 \cdots \geq a_k$. Suppose that $\{y_1, y_2, \ldots, y_k\}$ is an optimal choice for X_1. Let $b_i, 1 \leq i \leq k$, be the benefit of y_i under the index order. It is possible to show that

$$b_i \leq a_1 \qquad \text{for all } i \tag{1}$$

This is so because if this not the case, then y_i would be chosen first by the case-addition algorithm instead of x_1.

Let $c_{ij} = \sum_{x \in (N(x_j) - \cup_{s=1}^{j-1} N(x_s)) \cap (N(y_i) - \cup_{s=1}^{i-1} N(y_s))} P(x)$; c_{ij} is the common benefit offered by both y_i and x_j. Then

$$\sum_{i=1}^{k} c_{ij} \leq a_j \qquad \text{for all } j \tag{2}$$

This is so because the summands come from different pieces of $N(x_j)$.

$$b_i - \sum_{s=1}^{j-1} c_{is} \leq a_j \qquad \text{for all } i \tag{3}$$

This is so because otherwise y_i would be able to offer a bigger benefit for the remainder of the case base and thus would have been chosen to replace x_j in the jth iteration of the greedy algorithm.

Hence we have

$$\sum_{i=1}^{k} b_i \leq \sum_{i=1}^{k} \left(a_j + \sum_{s=1}^{j-1} c_{is} \right) \leq (k-1)a_j + \sum_{s=1}^{j} a_s$$

or

$$\sum_{i=1}^{k} b_i \leq k a_1 \tag{4}$$

$$\sum_{i=1}^{k} b_i \leq (k-1)a_2 + \sum_{s=1}^{2} a_s \tag{5}$$

$$\sum_{i=1}^{k} b_i \leq (k-1)a_3 + \sum_{s=1}^{3} a_s \tag{6}$$

$$\cdots$$

$$\sum_{i=1}^{k} b_i \leq (k-1)a_k + \sum_{s=1}^{k} a_s \tag{7}$$

From the preceding inequalities, we see that $\sum_{i=1}^{k} b_i$ is less than or equal to the minimum of the right sides of the inequalities. If these right sides are not equal, we can

take away some amount from a_i and add it to a_{i+1} or a_{i-1} so that all the right sides are equal in the end. This can be done because the inequalities imply many sequences of $x_i, i = 1, \ldots, k$, which have the same $\sum_{j=1}^{k} a_j$ and satisfy the inequalities. Out of these sequences, we can select the sequence that make the right sides equal and can provide the same $\sum_{j=1}^{k} a_j$ as a representative in subsequent derivations. Then we see that the difference between the ith right side and $i+1$st right side is $ka_{i+1} - (k-1)a_i$. Since the difference is set to 0, we conclude that $a_{i+1} = [k - 1/k]a_i$, or $a_i = [(k-1)/k]^{i-1}a_1$.

Using inequality (4), the ratio between the case coverage of $\{x_1, x_2, \ldots, x_k\}$ and the case coverage of $\{y_1, y_2, \ldots, y_k\}$ is

$$\frac{\sum_{i=1}^{k} a_i}{ka_1} \geq \frac{1}{k}\sum_{i=1}^{k}\frac{a_i}{a_1} = \frac{1}{k}\sum_{i=1}^{k}\left(\frac{k-1}{k}\right)^{i-1}$$

$$= 1 - \left(\frac{k-1}{k}\right)^{k} \geq 1 - e^{-1} \approx 0.63$$

The last inequality holds because $[(k-1)/k]^k \longrightarrow e^{-1}$, as $k \longrightarrow \infty$. This completes the sketch of our proof.

Considering the computational complexity of the algorithm, the case-addition algorithm updates the benefits of all remaining cases after each iteration. Thus the computational complexity is $O(n^2)$, where n is the size of the case base. This is so because the algorithm is dominated by computing the coverage of cases, which takes quadratic time, as is the case with Smyth and Keane's case-deletion algorithm. Linear time approximation can be considered using clustering algorithms such as the k-means algorithm (Hartigan 1975).

4. RELATING CASE-BASE SIZE TO COVERAGE

We have so far assumed that the case-base size k is given. Having a different k will result in a case base with a different coverage value. This fact is well known to CBR and machine-learning communities. In this section we show that it is in fact possible to analytically derive this relationship precisely by assuming different benefit functions for a_i, where a_i is the benefit of adding a case x_i to the new case base. With this analytical relationship, a case-base designer can *predict* the effectiveness of future algorithms. In essence, we ask: How should a case-base maintainer choose an appropriate size for a case base?

Let X_1 be a set of cases. We would like to estimate the ratio between $|X_1|$ and $|N(X_1)|$. Suppose that $x_1, x_2, \ldots, x_n$ are cases selected by our case-addition algorithm with the benefits of $a_1, a_2, \ldots, a_n$. Then

$$a_1 \geq a_2 \geq \cdots \geq a_n$$

Hence the ratio $|X_1|/|N(X_1)|$ is between $1/a_1$ and $1/a_n$. However, this estimation is rather rough.

Our theorem below points out a more precise relationship between case-base sizes and case-base coverage:

Theorem 2. Let M be the size of the original case base before maintenance. Let k be the size of the case base after maintenance. Suppose that when constructing the result case base, the benefits of new cases decrease *linearly*. Then we have

$$\text{Coverage} \geq R(2 - R)$$

where $R = k/M$.

Proof. We define n to be the case number such that when X_1 has n cases, the case-base coverage reaches 100 percent. Thus we may assume that $a_{n+1} = 0$. n can be trivially set to M. However, in general, n can be smaller than M.

Let $\bar{a}$ be the average of differences $(a_k - a_{k+1})$, that is, $\bar{a} = a_1/n$. Then $|N(X_1)| = \sum_{i=0}^{n}(a_1 - \bar{a}i)$. Let $k = rn$ be an integer, where $0 \leq r \leq 1$. Then

$$\frac{\sum_{i=1}^{k} a_i}{|N(X_1)|} = \frac{\sum_{i=0}^{k}(a_1 - \bar{a}i)}{\sum_{i=0}^{n}(a_1 - \bar{a}i)}$$

Notice that $\sum_{i=0}^{k}(a_1 - \bar{a}i)$ can be approximated by

$$\int_0^k (a_1 - \bar{a}x)dx = a_1 k - \tfrac{1}{2}\bar{a}k^2$$

and $n = a_1/\bar{a}$. Hence

$$\frac{\sum_{i=1}^{k} a_i}{|N(X_1)|} = \frac{a_1 k - \frac{1}{2}\bar{a}k^2}{a_1 n - \frac{1}{2}\bar{a}n^2} = \frac{a_1 rn - \frac{1}{2}\bar{a}(rn)^2}{a_1 n - \frac{1}{2}\bar{a}n^2}$$

$$= r\frac{a_1 - \frac{1}{2}\bar{a}r(a_1/\bar{a})}{a_1 - \frac{1}{2}\bar{a}(a_1/\bar{a})} = r(2 - r)$$

Recall that X_1 is assumed to have 100 percent coverage. Then $N(X_1) = M$, and the left hand side of the equation equals coverage. Let $R = k/M$. Then $R \leq k/n = r$. Therefore, Coverage $\geq R(2 - R)$. This completes the proof. ■

With linear reduction in benefits, roughly 15 percent of the cases can attain about 50 percent coverage, while 50 percent of cases can attain about 85 percent coverage. In general, we can do much better; the difference $(a_k - a_{k+1})$ can decrease faster than linear as k increases. The resulting graph of coverage is similar, but the desired coverage can be reached much sooner. This is shown in the following theorem about exponential decline in the benefits.

Theorem 3. Let M be the size of the original case base before maintenance. Let k be the size of the case base after maintenance. Suppose that when constructing the result case base, the benefits of new cases decrease *exponentially*. Then the coverage of the case base also increases exponentially.

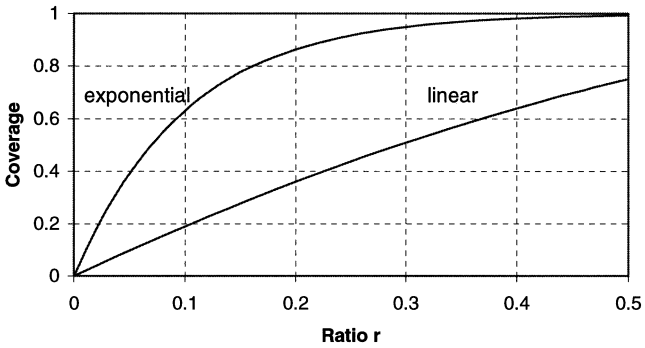

FIGURE 4. Coverage Graph with Linear and Exponential Benefit Reduction.

Proof. Assume that the benefit of the ith case added by the case-addition algorithm decreases exponentially as

$$a_i = a_1 \frac{1 - e^{i-n}}{1 - e^{-n}} \qquad i = 1, 2, \ldots, k$$

where the factor $1 - e^{-n}$ is inserted for normalization. Then the total benefit of a case base of size k is

$$\sum_{i=1}^{k} a_i = Cn\left(r - \frac{e^{n*(r-1)}}{n}\right)$$

where C is a constant and r is the ratio k/n. Dividing this formula by M, we get

$$\text{Coverage} \geq C\left(R - \frac{e^{R-1}}{M}\right)$$

where $R = k/M$ and C is a constant. This completes the proof. ■

A graph comparing both the exponential benefit reduction and linear reduction is shown in Figure 4. As can be seen from the figure, the exponential graph can quickly cover a much larger portion of a case base than its linear-benefit counterpart.

5. EXPERIMENTS IN CASE-BASED PLANNING DOMAINS

We have so far discussed case-addition policy in the context of general CBR. In this section we instantiate the case-addition algorithm in case-based planning using a state-space representation. We verify the effectiveness of the case-addition policy in two experimental domains, a Towers of Hanoi domain and a path-finding domain.

In both domains, a planning problem is a pair of states (s_i, s_g), where s_i is an initial state and s_g is a goal state. A case is a pair $(p, soln)$, where p is a planning problem and $soln$ a sequence of actions transforming the initial state s_i to s_g. Also for both domains, given two cases $c_1 = (p_1, soln_1)$ and $c_2 = (p_2, soln_2)$, the adaptation cost from case c_1 to case c_2 is defined as the number of steps that need be added to the common parts

of $soln_1$ and $soln_2$ in order to obtain $soln_2$. Adding steps to their common parts will "connect" the parts with the initial and goal states. In other words,

$$\text{Adaptation Cost}(c_1, c_2) = |soln_2 - soln_1 \cap soln_2|$$

Note that this cost can be easily computed in a state space representation. Based on this cost, we can define the neighborhood of a case x as those cases in the case base that are within an adaptation cost of L or less from x. In the experiments, we set δ to be 2.

In the first domain, our problem is four-disk Tower of Hanoi problem. In this problem, there are three pegs and four different sized disks initially placed on one of the pegs. The problem is to move them to another peg such that in the process no disk of a smaller size is placed on top of a larger one. A problem in this domain is a pair $\langle s_i, s_g \rangle$, where s_i and s_g can be any states.

In the problem definition, the number of states is $3^4 = 81$. Therefore, the largest case base would contain $81^2 = 6561$ cases. This is the original case base X that we will consider. We first randomly selected 200 planning problems from the state space and solved these problems from scratch using a best-first forward-chaining planner (see Yang 1997). This gave us 200 randomly chosen cases from the original case base of 6561. By computing the neighbors of these problems, we obtained a coverage of 5649 cases. This represents coverage of 86 percent of the entire case base.

With the case-addition policy, we can do better. Applying the case-addition algorithm, we select cases to cover the original case base, and our result is shown in Table 1. From this table, we can see that the case-addition algorithm selected 58 cases to attain full competence of the case base X that contains all 6561 planning problems. These 58 problems performed much better than the randomly chosen 200 cases earlier. From the table, we also can see that the last few problems are much less important. For example, the last 38 problems only contribute to 10 percent of the coverage.

We also have performed a comparative study of case-addition policy against case-deletion policy in the Towers of Hanoi problem. In this experiment, we implemented the case-deletion algorithm (Smyth and Keane 1995) by deleting the auxiliary cases and spanning and support cases before the pivotal cases down to a specified number of cases that is the case-base size. For each case-base size, we then compared the case-base coverage under the two algorithms. The result, shown in Figure 5, demonstrates that the case-addition policy indeed performed better than case-deletion policy in this domain.

We also designed a path-finding domain whose state graph is shown in Figure 6. The problem is to find a path between any two states that represent two cities. In this problem, an agent can move horizontally or along the main diagonal lines. The number

TABLE 1. Towers of Hanoi Domain Coverage Test

Case-base size	1	2	3	4	5	6	7	8	9
Benefits	676	616	593	508	459	354	243	236	232
Coverage	0.12	0.23	0.33	0.42	0.50	0.57	0.61	0.65	0.69
Case-Base Size	10	15	20	25	30	35	—	58	59
Benefits	228	106	48	28	24	12	—	2	0
Coverage	0.73	0.84	0.90	0.94	0.96	0.97	—	1	1

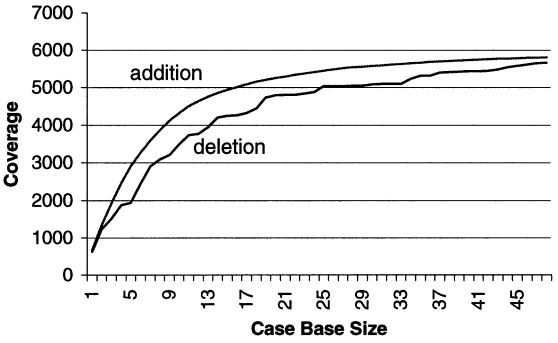

FIGURE 5. Coverage Comparison between Case-Addition (Upper Curve) and Case-Deletion Algorithms in the Towers of Hanoi Domain.

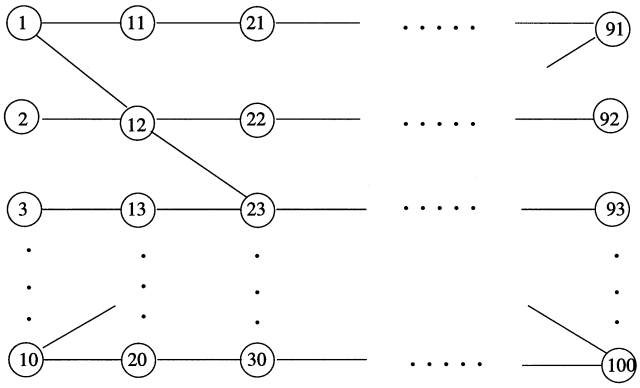

FIGURE 6. State Graph of a Travel Domain.

of states is 100. Thus the problem space has $100^2 = 10,000$ problems. We randomly select 80 problems from the problem space and solve these problems from scratch by a forward-chaining best-first planner. By computing the neighbors of these cases, we see that the union of all these neighbors covers 3138 cases. This represents 30 percent coverage, as shown in Figure 7.

6. CONCLUSIONS

CBM is critical in ensuring the continuing success of a CBR system. Different maintenance policies will result in case bases with different quality. In this article, we have presented a case-addition based policy for computing a case base with near-optimal coverage. We attribute this property to the fact that we use a case-addition-based policy in deciding the composition of the case base. Furthermore, through Theorems 2 and 3, we show that the reduction speed in case benefits as cases are added to a case base has significant impact on coverage speed. We also have demonstrated two experimental domains where the case-addition algorithm performs well.

We would like to further qualify our performance claim by stating that while the case-addition policy has a worst-case guarantee, we *do not* claim that on average it is better than other CBM policies. Such average case analysis, which is beyond the scope

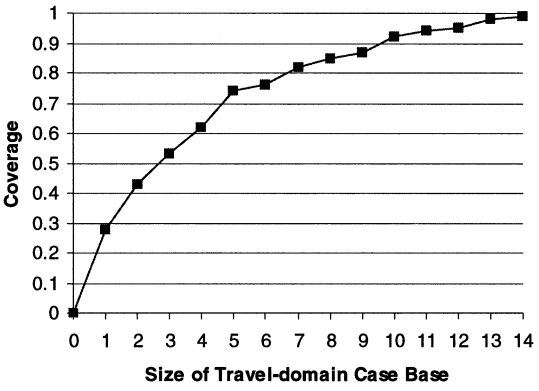

FIGURE 7. Coverage Graph of a Travel Domain.

of this article, would require a careful analysis of problem distribution in the real world. In the future, we plan to build algorithms that keep track of the adaptation costs and neighborhood estimates in order to incrementally decide how to update a case base.

ACKNOWLEDGMENTS

We were supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), the IRIS-III Project, an Ebco/Epic NSERC Industrial Chair Fund, BC Advanced Systems Institute, and Canadian Cable Labs Fund.

REFERENCES

AHA, D. W., and L. BRESLOW. 1997. Refining conversational case libraries. *In* Proceedings of the Second International Conference on Case-Based Reasoning (ICCBR-97). Providence, RI. Springer-Verlag, Berlin, pp. 267–276.

AHA, D., D. KIBLER, and M. ALBERT. 1991. Instance-based learning algorithms. Machine Learning, **6**:37–66.

CUNNINGHAM, P., A. BONZANO, and B. SMYTH. 1997. Using introspective learning to improve retieval in car: A case study in air traffic control. *In* Proceedings of the Second International Conference on Case-Based Reasoning (ICCBR-97). Providence, RI. Springer-Verlag, Berlin, pp. 291–302.

DOMINGOS, P. 1995. Rule induction and instance-based learning. *In* Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, pp. 1226–1232.

FOX, S., and D. B. LEAKE. 1995. Learning to refine indexing by introspective reasoning. *In* Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada, August 1995, Morgan Kaufmann, San Mateo, CA.

GAREY, MICHAEL R., and DAVID S. JOHNSON. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York.

HARTIGAN, J. A. 1975. Cluster Algorithms. John Wiley, New York.

HARINARAYAN, V., A. RAJARAMAN, and J. D. ULLMAN. 1996. Implementing data cubes efficiently. *In* ACMSIGMOD. ACM Press, New York, pp. 311–320.

LEAKE, DAVID. 1996. Case-Based Reasoning: Experiences, Lessons and Future Directions. AAAI Press/MIT Press, Cambridge, MA.

LEAKE, D. B., and D. C. WILSON. 1998. Categorizing case-base maintenance: Dimenions and directions. *In* Proceedings of the 1998 European Workshop on CBR (EWCBR-98), Springer-Verlag, Berlin.

MINTON, S. 1990. Qualitative results concerning the utility of explanation-based learning. Artificial Intelligence, **42**:363–391.

MARKOVICH, S., and P. SCOTT. 1988. The role of forgetting in learning. *In* Proceedings of the Fifth International Conference on Machine Learning. Morgan Kaufmann, San Mateo, CA, pp. 459–465.

RACINE, KIRSTI, and QIANG YANG. 1996. On the consistency management of large case base: The case for validation. AAAI Technical Report-Verification and Validation Workshop, Menlo Park, CA.

RACINE, KIRSTI, and QIANG YANG. 1997. Maintaining unstructured case bases. *In* Proceedings of the Second International Conference on Case-Based Reasoning (ICCBR-97). Providence, RI. Springer-Verlag, Berlin, pp. 553–564.

SMYTH, B., and M. KEANE. 1995. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. *In* International Joint Conference on Artificial Intelligence, Vol. 1, Morgan Kaufmann, San Mateo, CA, pp. 377–382.

WATSON, IAN. 1997. Applying Case-Based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann, San Francisco.

ZHANG, ZHONG, and QIANG YANG. 1999. Dynamite refinement of feature-weights in CBR using quantitative introspective learning. *In* Proceedings of the International Joint Conference In Artificial Intelligence 1999 (IJCAI-99). Morgan Kaufmann, San Mateo, CA.

YANG, QIANG. 1997. Intelligent Planning: A Decomposition and Abstraction Based Approach. Springer-Verlag, Berlin.