IUCBRF Lesson: Case Base Maintenance Policies *

Steven Bogaerts and David Leake
Computer Science Department, Indiana University, Lindley Hall 215
150 S. Woodlawn Avenue, Bloomington, IN 47405, U.S.A.
{*sbogaert,leake*}@*cs.indiana.edu*

March 24, 2005

1 Introduction

Case-base maintenance revises the contents or organization of a case base to improve system performance. As case base sizes increase and CBR systems receive long-term use, maintenance becomes an important concern. Consequently, maintenance has received considerable research attention (e.g., (Leake *et al.* 2001)).

This assignment will use experiments with IUCBRF, a case-based reasoning framework, to allow you to analyze the effects of four simple case base maintenance approaches on the performance of system components.

The assignment begins with a description of the maintenance strategies to examine and high-level assignment tasks, followed by a technical description of how to set up and run the framework. You are advised to read the entire assignment before starting work.

2 Strategies to Examine

Consider the following four case base maintenance techniques already implemented in IUCBRF. Each is described by its policies for addition to and deletion from the case base. For addition, the problem description and the system-deduced solution form the case that is added.

1. `NullMaintenance`
   Addition: never
   Deletion: never
2. `AlwaysAddMaintenance`
   Addition: always
   Deletion: never
3. `UnusedRemovedMaintenance`
   Addition: always
   Deletion: Periodically, all cases are checked offline. A case is removed if it has not been used sufficiently frequently since the last check, taking into account the lower probability of case use in larger case bases. More precisely, define case age as the number of episodes a CBR system has seen since the case was added to the case base. For a case base of size s, and parameter γ, a case of age a used u times is removed if $u < \gamma \times (a/s)$. We will assume γ is set to 1.

4. `ThresholdBasedAdditionMaintenance`
   Addition: A case is added if the difference between it and the closest case in the case base surpasses a fixed threshold.
   Deletion: never

3 Tasks

3.1 Question 1: Hypotheses

Before running any experiments, consider the following questions and write down your predictions, to hand in: What kind of performance do you expect for the different policies? How will final case-base sizes and solution quality compare? Explain your rationale.

3.2 Experiments

Run the prepared experiments (see technical details below for help) for the UCI Letter domain (Blake & Merz 2000) and examine the results.

Note that this may take a while. For preliminary exploration you may wish to work with a smaller test set by copying letter.test to letter.smalltest, for example, and removing some cases with a text editor. As you run the data, monitor processing to get a feeling for how efficiency changes over time.

3.3 Question 2: Results And Analysis

What are the results? Select an illustrative subset of the data to graph quality and efficiency. Your results may be surprising. How do you account for this? In developing your explanation, examine the maintenance policy descriptions carefully.

4 System Setup and Execution

IUCBRF is a freely-available framework for case-based reasoning, written in Java (Bogaerts & Leake 2005). For this assignment you will be using built-in procedures, and the instructor will email the code to the class.

4.1 Installation of IUCBRF Code

Choose a directory (for example, javaroot) in which to unpack iucbrf.jar, using the command:

```
jar xvf iucbrf.jar
```

Unpack iucbrflessons.jar in the same directory. Make sure that your Java classpath points to this directory (e.g., javaroot). Note that, in accordance with Java naming conventions, the top-level subdirectory containing the IUCBRF code will be "edu".

Though it is not required for this assignment, we encourage you to browse the IUCBRF code and the documentation described at the end of this writeup, and to try out some of the small demonstration systems provided with this code.

4.2 Execution

The directory edu/indiana/iucbrflessons/domains/letter contains all code and data specific to the letter domain:

- LetterSystem.java - The implementation of a CBR system for the letter domain.
- letter.names - A description of the letter domain.
- letter.data - The raw case data.
- letter.test - The raw test data.

You must generate a case base file (.cb) from each file of case and test data that you will use. To generate a case base file from raw data (either the files above, or smaller files you create for test purposes), use a command of the following form:

```
java edu.indiana.iucbrflessons.CBReader <system class> <raw file name>
```

For example, in edu/indiana/iucbrflessons/domains/letter, run (all typed on one line):

```
java edu.indiana.iucbrflessons.CBReader
edu.indiana.iucbrflessons.domains.letter.LetterSystem
letter.data
```

to generate letter.data.cb in that directory. Do the same for letter.test.

Then, to run experiments for the four maintenance methods above, run (all typed one line):

```
java edu.indiana.iucbrflessons.maint.MaintTestClass
<system class>
<serialized data cb file name>
<serialized test cb file name>
```

For example, in edu/indiana/iucbrflessons/domains/letter, run (all typed on one line):

```
java edu.indiana.iucbrflessons.maint.MaintTestClass
edu.indiana.iucbrflessons.domains.letter.LetterSystem
letter.data.cb
letter.test.cb
```

On a Unix system, the `script` command can be used to maintain a transcript of your results as the system runs. As the runs may be lengthy, you can consult the state of the in-progress results periodically to get a sense of the processing time, and then can examine the transcript data as the basis for your graphs.

5 Interpreting Output from the Experiments

The experiments set up in IUCBRF proceed as follows:

- Load the test set file.
- EPOCH_SIZE := 500
- episodeID := 0
- For each case c in the test file:
  - Provide the problem description of c to the CBR system and obtain a proposed solution.
  - Check the proposed solution against the actual solution in c.
  - Run the addition portion of the maintenance policy to determine if c should be added to the case base.
  - If (episodeID mod EPOCH_SIZE == 0) then
    - Output a performance monitor report, and then reset all statistics. (Thus, reports are **not** cumulative from one set of EPOCH_SIZE episodes to the next.)
    - Check the entire case base for cases to remove, according to the deletion portion of the maintenance policy.
  - episodeID := episodeID + 1

The following statistics are shown in the performance monitor report:

- System Age - The number of episodes the system has seen *with this performance monitor*. Note that because the performance monitor is reset after each set of 500 test problems, the system age according to a single performance monitor will never exceed 500.
- CB Size - The size of the case base.

- Avg Retrieve Time - The average time for each case retrieval, in ms.
- Avg Adapt Time - The average time for processing by the adaptation code, in ms (this will be nearly 0 because no adaptation is done in this experiment).
- Avg Solution Quality - For this test domain, solution quality is simply set to 1.0 if the proposed solution exactly matches the actual solution, and 0.0 otherwise.
- Solved Well - This measures the percentage of episodes with solution quality (as defined above) over some threshold. For this domain, the solution quality threshold for a problem to be "solved well" is 1.0. Any lower solution quality (that is, 0.0) means that the problem was not solved well. Thus, due to the simple solution quality measure defined above for this domain, average solution quality and solved well are the same measure, but for a more refined solution quality measure, the solved well percentage could use a threshold anywhere between 0.0 and 1.0 to indicate that a solution is acceptable.

6 About IUCBRF

The Indiana University Case-Based Reasoning Framework (IUCBRF), developed by Steven Bogaerts under the guidance of David Leake, is an open source domain-independent framework for case-based reasoning (CBR) system development in Java. For the official manual (Bogaerts & Leake 2005) and all other available documentation and to obtain the source code, please follow the IUCBRF link from the IU CBR resources page, http://www.cs.indiana.edu/∼leake/cbr_resources.

References

Blake, C., and Merz, C. 2000. UCI repository of machine learning databases.

Bogaerts, S., and Leake, D. 2005. IUCBRF: A framework for rapid and modular CBR system development. Technical Report TR 608, Computer Science Department, Indiana University, Bloomington, IN.

Leake, D.; Smyth, B.; Wilson, D.; and Yang, Q., eds. 2001. *Maintaining Case-Based Reasoning Systems*. Blackwell. Special issue of *Computational Intelligence*, 17(2), 2001.