Improving a Distributed Case-Based Reasoning System Using Introspective Learning

Ian Watson
AI-CBR
University of Salford
Salford, M7 9NU
UK
ian@ai-cbr.org
www.ai-cbr.org

Dan Gardingen
Western Air Ltd.
McCabe Street, North Fremantle
Fremantle
Western Australia

Abstract

This paper describes the improvements to a fielded case-based reasoning (CBR) system. The system, called "Really Cool Air" is a distributed application that supports engineering sales staff. The application operates on the world wide web and uses the XML standard as a communications protocol between client and server side Java applets. The paper briefly describes the distributed architecture of the application, the two case retrieval techniques used, and improvements made in three areas: software infrastructure, retrieval algorithm and most crucially the automatic adjustments of query relaxation parameters by an incremental learning algorithm. The results of the incremental learning algorithm are quantified.

1 Introduction

A paper describing the design, implementation, testing, roll-out and benefits of the "*Really Cool Air*" system was presented to this conference in 1998 [Gardingen & Watson 1998] and subsequent improvements to the fielded system were briefly described in a paper presented at IJCAI-99 [Watson & Gardingen 1999]. However, the publication of the later paper came before detailed evaluation of the improvements could be reported. This paper is therefore able to present detailed quantitative results of experiments to test the efficiency of an introspective learning algorithm that automatically learns query relaxation parameters. For the benefit of readers who have not read the previous papers the systems purpose and architecture is briefly described before describing the improvements and evaluating the incremental learning strategy.

Thus, section 2 describes the background to the system, it's raison d'être, its implementation and architecture, case representation and retrieval strategy and its interface design. Section 3 describes the improvements to the system's architecture including an evaluation of the efficiency gains obtained through the implementation of an introspective learning algorithm. Section 4 concludes the paper.

2 Background

Western Air is a distributor of HVAC (heating, ventilation and air conditioning systems in Australia with a turnover in 1997 of $25 million (US dollars). Based in Fremantle the company operates mainly in Western Australia, a geographic area of nearly two million square miles. The systems supported range from simple residential HVAC systems to complex installations in new build and existing factories and office buildings.

Western Air has a distributed sales force numbering about 100. The majority of staff do not operate from head office but are independent, working from home or a mobile base (typically their car). Until recently, sales staff in the field would gather the prospective customer's requirements using standard forms and proprietary software, take measurements of the property and fax the information to Western Air in Fremantle. A qualified engineer would then specify the HVAC system. Typically the engineer would have to phone the sales staff and ask for additional information and the sales staff would have to make several visits to the customer's building and pass additional information back to the head office engineer.

Western Air felt that basing a quote on the price of a previous similar installation gave a more accurate estimation than using prices based on proprietary software, catalogue equipment prices and standard labour To try to help engineers make use of all the past installations a database was created to let engineers search for past installations. The database contained approximately 10,000 records, each with 60 fields describing the key features of each installation and then a list of file names for the full specification. Initially the engineers liked the database and it increased the number of past installations they used as references. However, after the honeymoon ended, they started to complain that it was too hard to query across more than two or three fields at once. And that querying across ten or more fields was virtually impossible. In fact most of them admitted to using the database to laboriously browse through past installations until they found one that looked similar to their requirements.

2.1 Implementation

Western Air decided that merely improving the efficiency of the engineers in Fremantle would not solve the whole problem. Ideally they would like the sales staff to be able to give fast accurate estimates to prospective customers on the spot. However, they were aware that there was a danger that the less knowledgeable sales staff might give technically incorrect quotes.

The solution they envisaged was to set up a web site that sales staff could access from anywhere in the country. Through a forms interface the prospect's requirements could be input and would be passed to a CBR system that would search the library of past installations and retrieve similar installations. Details of the similar installations along with the ftp addresses of associated files would then be available to the sales staff by ftp. The sales staff could then download the files and use these to prepare an initial quote. All this information would then be automatically passed back to an engineer to authorise or change if necessary. Once an installation was completed its details would be added to the library and its associated files placed on the ftp server.

2.2 Implementation Plan

Web based CBR applications have been demonstrated for a few years now such as the FAQFinder and FindME systems [Hammond et al., 1996] and those at Broderbund and Lucas Arts [Watson, 1997]. The solution they envisaged was to set up a web site that sales staff could access from anywhere in the country. Through a forms interface the prospect's requirements could be input and would be passed to a CBR system that would search the library of past installations and retrieve similar installations. Details of the similar installations along with the FTP addresses of associated files would then be available to the sales staff by FTP. The sales staff could then download the files and use these to prepare an initial quote. All this information would then be automatically passed back to an engineer to authorise or change if necessary. Once an installation was completed its details would be added to the library and its associated files placed on the ftp server.

Since a simple nearest neighbour retrieval algorithm would suffice implementing our own system was a viable option. Java (Visual Café) was chosen as the implementation language for both the client and server side elements of the CBR system. XML (eXtensible Markup Language) [WWW Consortium, 1997] was used as the communication language between client and server-side applets. The World-Wide Web Consortium (W3C) finalised XML 1.0 in December 1997 as a potential successor to HTML [WWW Consortium].

2.3 System Architecture

On the sales staff (client) side a Java applet is used to gather the customer's requirements and send them as XML to the server. On the server side another Java applet (a *servlet*) uses this information to query the database to retrieve a set of relevant records. The Java servlet then converts these into XML and sends them to the client side applet that uses a nearest neighbour algorithm to rank the set of cases.

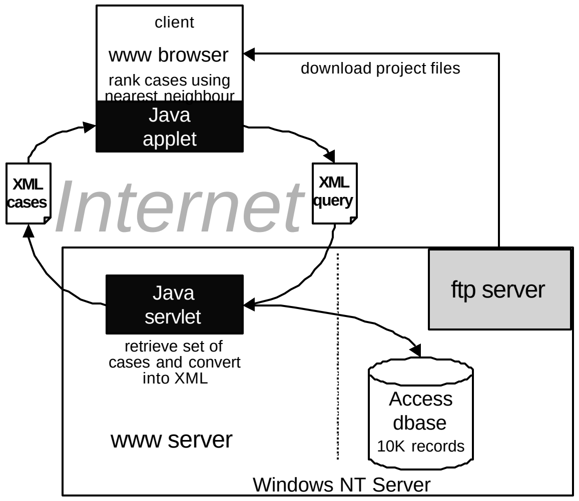


Figure 1. System Architecture

2.4 Case Representation

Cases are stored within a database. Each record (case) comprises 60 fields used for retrieval and many more used to describe the HVAC installations. In addition, links to other files on the FTP server are included to provide more detailed descriptions. Once retrieved from the database the records are ranked by a nearest neighbour algorithm and dynamically converted into XML for presentation to the client browser. An XML case representation is used by our system [Shimazu, 1998]. XML pages can contain any number of user defined tags defined in a document type definition (DTD) file. Tags are nested hierarchically from a single root tag that can contain any number of child tags. Any child tag in turn can contain any number of child tags. Each tag contains a begin statement (e.g. <Case>) and an end statement (e.g. </Case>). This is illustrated in Figure 4.

2.5 Case Retrieval

Case retrieval is a two stage process. In stage one the customer's requirements are relaxed through a process of *query relaxation*. This process takes the original query and relaxes terms in it to ensure that a useful number of records are retrieved from the database. This is similar to the technique used by Kitano & Shimazu [1996] in the SQUAD system at NEC, although as is discussed in section 3, we have improved it efficiency using an introspective learning heuristic.

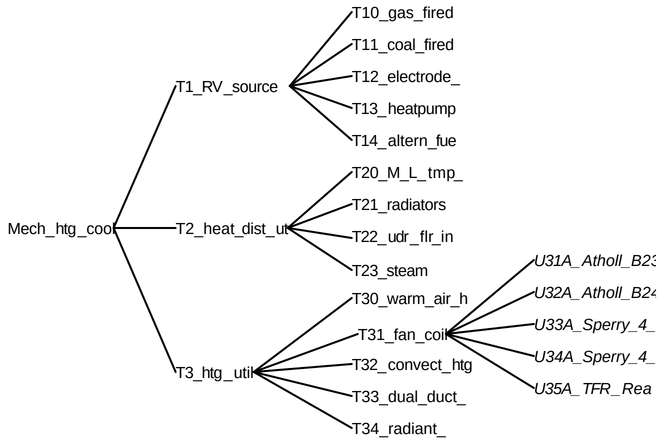


Figure 2. A Portion of a Symbol Hierarchy for Mechanical Heating & Cooling Systems

For example, assume we are trying to retrieve details of installations using Athol B25 equipment. An SQL query that just used "Athol_B25" as a search term might be too restrictive. Using an ordered symbol hierarchy (as in Figure 2) our system knows that "Athol B25" is a type of "Fan Coil" system so the query is relaxed to "Where (((EquipmentReference) = "T31_fan_coil""))..)). This query will include equipment from Athol, Sperry and TFR. An ordering of each set of symbols in the hierarchy is obtained through the reference number suffixes to each symbol (e.g. T10, T11, T12, T13, T14 as shown in Figure 2). The symbol hierarchies are stored in a table in the database.

Other specific criteria, elevations or temperatures that are numbers (integers or reals) can be relaxed by using simple ranges (e.g. a temperature of 65° F. could be relaxed to "Between 60 And 70"). Knowledge engineering was required to determine by what amounts numeric features should be relaxed. The relaxation is expressed as a term ± a percentage (e.g., "Relax_Temp = ± 10%"). These relaxation terms are stored in a table in the database.

In the second stage the small set of retrieved records are compared by the client-side applet with the original query and similarity is calculated using the simple nearest neighbour algorithm shown in Figure 3. The resulting similarity measure is normalised to give a percentage range of 0% (i.e. completely dissimilar) to 100% (i.e. completely similar). The weighting on the features by default is set to 1 (i.e., all features are by default considered of equal importance) However, the sales engineers can change the feature weightings to reflect client priorities or their own preferences.

$$Similarity(T,S) = \sum_{i=1}^{n} f(T_i, S_i) \times w_i$$

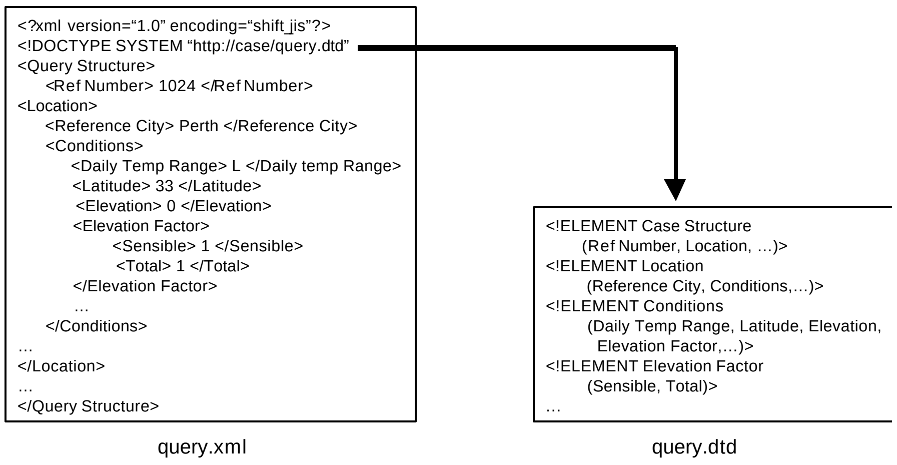
where:
T is the target case, *S* is the source case, *n* is the number of features in each case
i is an individual feature from 1 to *n, f* is a similarity function for feature *i* in cases *T* and *S* and *w* is the importance weighting of feature *I*

Figure 3 The Nearest Neighbour Algorithm

Once an HVAC installation is completed its details are added to the database and its associated files placed on the FTP server. Having a database management system for the case repository has proved essential since it makes it easier to generate management reports and ensure data integrity. It would be almost impossible to maintain a collection of 10,000 cases without a DBMS.

2.4 Interface Design

The interface to the system is a standard Java enabled web browser (Netscape or Internet Explorer). The forms within the Java applet were designed to look as similar to the original forms, HVAC specification tools and reports that the sales staff were already familiar with. Microsoft FrontPage 98 and Macromedia's DreamWeaver were the primary tools used to create the web site. A sample screen from the client side Java applet can be seen in Figure 6.



```
<?xml version="1.0" encoding="shift_jis"?>
<!DOCTYPE SYSTEM "http://case/query.dtd"
<Query Structure>
    <Ref Number> 1024 </Ref Number>
<Location>
    <Reference City> Perth </Reference City>
    <Conditions>
        <Daily Temp Range> L </Daily temp Range>
        <Latitude> 33 </Latitude>
         <Elevation> 0 </Elevation>
        <Elevation Factor>
              <Sensible> 1 </Sensible>
               <Total> 1 </Total>
        </Elevation Factor>
        …
    </Conditions>
…
</Location>
…
</Query Structure>
```

query.xml

```
<!ELEMENT Case Structure
      (Ref Number, Location, …)>
<!ELEMENT Location
       (Reference City, Conditions,…)>
<!ELEMENT Conditions
       (Daily Temp Range, Latitude, Elevation,
         Elevation Factor,…)>
<!ELEMENT Elevation Factor
       (Sensible, Total)>
…
```

query.dtd

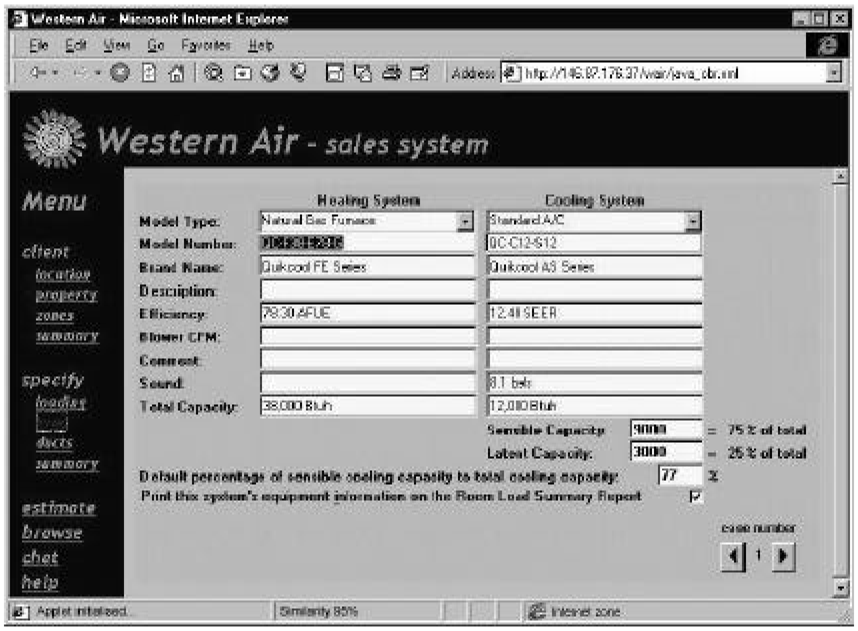
Figure 4. A Sample of the XML Case Description

```
SELECT Location.ReferenceRegion, Location.DailyTempRange,
Location.Lattitude, Location.Elevation, Location.ElevationFactorS,
Location.ElevationFactorT, Location.DryBulbTempWin, Loca-
tion.DryBulbTempSum, Location.WetBulbTemp,
…
FROM Location
WHERE (((Location.ReferenceRegion)="SW") AND ((Loca-
tion.Elevation) Between 0 And 100) AND ((Loca-
tion.DryBulbTempWin) Between 50 And 60) AND ((Loca-
tion.DryBulbTempSum) Between 60 And 70))
…
```

Figure 5. Example of an SQL Query That Has Been Relaxed

3 System Enhancements

Although the initial testing and roll-out of the system was judged a success, not least in commercial terms the original implementation of the system experienced increasing load performance problems. The Java servlet approach suffered from poor performance because the web server loads, executes and terminates a new servlet program for each user access. Large data sets and complex queries especially burden the system because data querying takes place via the Java servlet program rather than directly via the database. This coupled with the fact that MS Access is not a particularly fast database caused time out problems as the server load increased. To rectify this problem the database was ported to mySQL (http://www.tcx.se) a freeware database with much better performance.



Western Air - sales system

	Heating System	Cooling System
Model Type:	Natural Gas Furnace	Standard A/C
Model Number:	[illegible]	QC-C12-S12
Brand Name:	Quikcool FE Series	Quikcool AS Series
Description:		
Efficiency:	78.30 AFUE	12.41 SEER
Blower CFM:		
Comment:		
Sound:		8.1 bels
Total Capacity:	38,000 Btuh	12,000 Btuh

Sensible Capacity: 9000 = 75 % of total
Latent Capacity: 3000 = 25 % of total
Default percentage of sensible cooling capacity to total cooling capacity: 77 %
Print this system's equipment information on the Room Load Summary Report ✓

case number 1

Applet initialized. | Similarity 85% | Internet zone

Western Air - Microsoft Internet Explorer
File Edit View Go Favorites Help
Address http://146.87.176.37/wair/java_cbr.html
Menu
client
location
property
zones
summary
specify
loadings
ducts
summary
estimate
browse
chat
help

Figure 6. Client Side Java Applet Showing Retrieved HVAC Case

In addition Netscape's LiveWire database integration tool was used. This product has excellent database query functions, and importantly, because the LiveWire engine runs within the Netscape Web Server process it can share database connections across all Web accesses.

3.1 Introspective Learning

The initial query relaxation method of first performing a precise query and then relaxing the query through successive iterations until a sufficiently large set of cases was retrieved also compounded the performance problems. A suggestion was made to turn this process around – namely, why not relax the initial query far enough to ensure that a large set of cases would be retrieved (e.g. several hundred cases) and then refine the query to reduce the sub-set to around twenty cases). The obvious speed advantage in this approach is that only a small sub-set of the whole case-base is used in any subsequent iterations as opposed to the entire case-base in the original query relaxation approach.

However, deciding how much to relax the query was not straightforward so an introspective learning approach was taken. This is an approach in CBR where the reasoning system itself learns over time to modify its internal representation to improve its performance [Markovitch & Scott, 1993]. For example, CBR systems may learn to modify feature weights, adaptation rules [Leake, et al., 1995; Hanney & Keane, 1996], or even learn to forget redundant cases [Smyth & Cunningham, 1996].

A decision was made to log each time a feature was relaxed during the query relaxation process. When the same query term is encountered again the query is automatically relaxed by one of three methods:

1. by the precise amount it was relaxed the previous time,
2. by the average amount it was relaxed the previous *N* times it had been relaxed, and
3. by the mean amount it was relaxed the previous *N* times it had been relaxed.

Experiments were then conducted to see which, if any, of these simple heuristics most improved retrieval efficiency. This of course required that we decided what we meant by efficiency, was it:

- Time efficiency, i.e. purely a measure of retrieval speed, or
- Accuracy, i.e., a measure of the quality of the final suggested set of cases.

It was felt by the developers that both metrics were important so both were considered. Because of problems due to network and server loads and the way command stacks were executed it was difficult to directly measure retrieval speed accurately so the concept of *precision* was used as an analogue. That it how many cases were returned in the set of retrieved cases. The fewer the number the more precise. A smaller number of cases would always be processed faster by the system, hence retrieval time would be quicker (unless network traffic slowed down the exchange of XML information).

Therefore, if the query relaxation algorithm returned a single case that was a 100% match to the target case we would have an accuracy rating of a perfect 100% with a precision of 100%. It is worth remembering that this only tests the first stage of retrieval. In the second stage a nearest neighbour algorithm selects the most similar case from the retrieved set.

Testing

Test were conducted by partitioning the case-base. Approximately one fifth of the cases (2000) were removed at random from the case-base. These were then used in a random sequence as probes to query the case-base. Our hypothesis therefore was that if the introspective learning algorithm worked either the precision or accuracy or both of the system should improve over time.

Results

The results are presented as a series of graphs. three methods were used by the introspective learning algorithm to learn how to relax the query:

1. *precise relaxation* – i.e., relax query features by the precise amount they were relaxed the previous time.

2. *average relaxation* – i.e., relax query features by the average amount they were relaxed the previous n times they had been relaxed (where n is the total number of times it has been relaxed), and

3. *mean relaxation* – i.e., relax query features by the mean amount they had been relaxed the previous n times they had been relaxed.

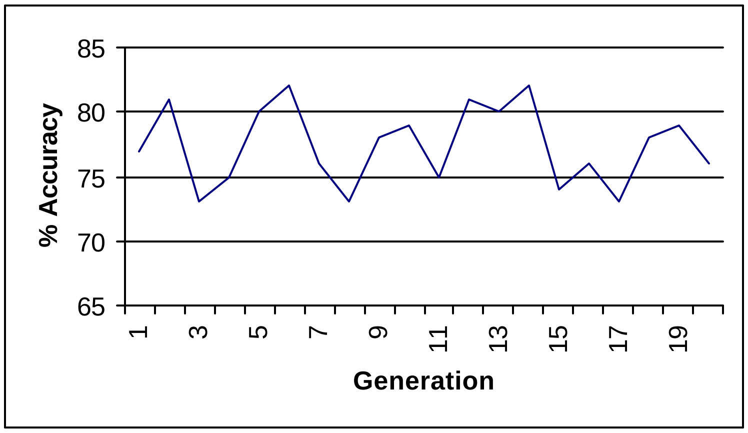


Figure 7 % Accuracy for Precise Relaxation
(Generation is in 100s)

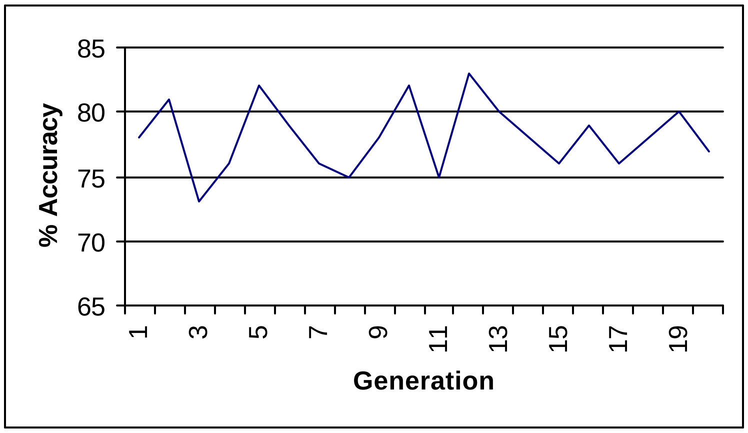


Figure 8 %Accuracy for Average Relaxation

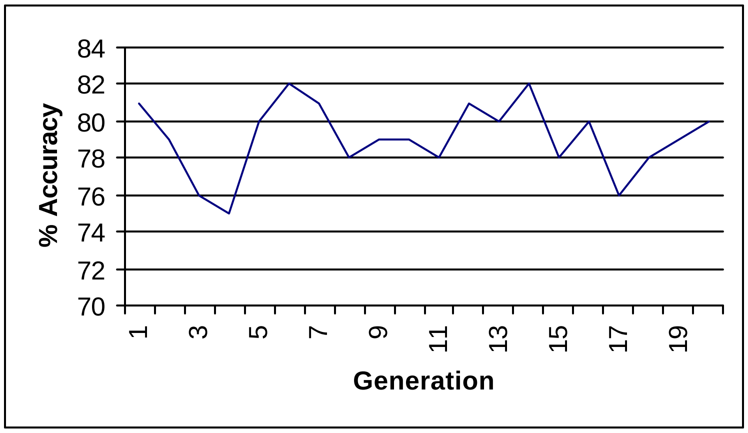


Figure 9 % Accuracy for Mean Relaxation

The results for percentage accuracy (Figures 7-9) were entirely inconclusive. % Accuracy did not improve using either of the three introspective learning algorithms. This is not surprising since the objective of the query relaxation method is not to retrieve the “best” matching case but rather to retrieve a set of good candidate cases upon which the nearest neighbour algorithm can work. However, conversely there was no evidence that introspective learning reduced the accuracy of the set of retrieved cases.

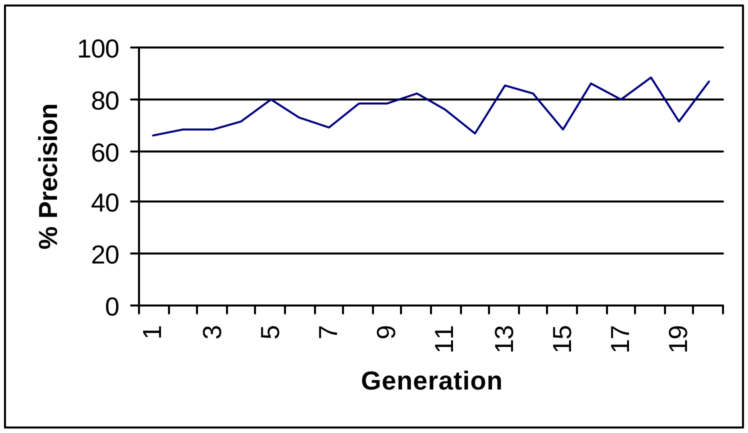


Figure 10 % Precision for Precise Relaxation

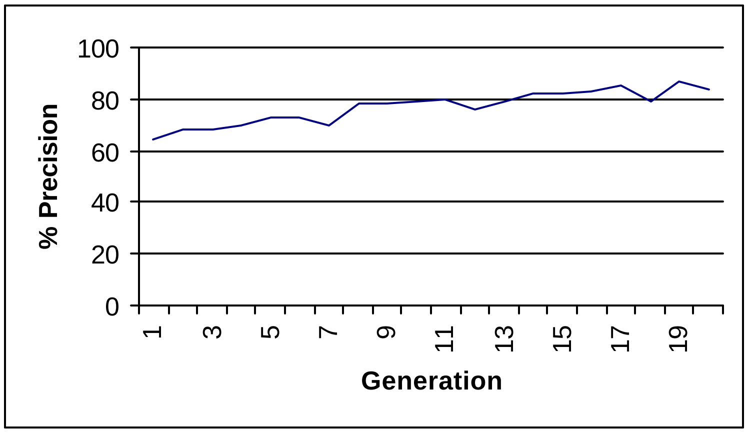


Figure 11 % Precision for Average Relaxation

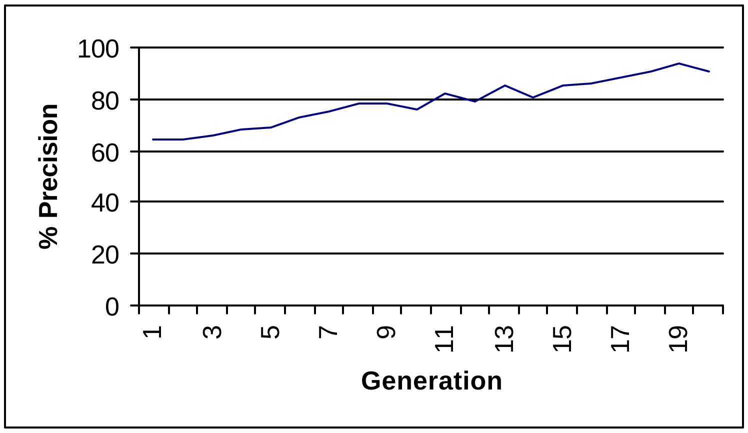


Figure 12 % Precision for Mean Relaxation

The results for precision were more encouraging (Figures 10-12), except for when the precise relaxation learning algorithm was used. It's results unsurprisingly showed wild fluctuations as each query relaxation was based purely on the previous query relaxation. Sometimes this worked and sometimes it didn't; in effect the learning algorithm had no long term memory. However, the average and mean relaxation results showed improvement with time. Precision for average relaxation improved from about 64% to 84%, whilst the mean relaxation improved from about 64% to 94%. In addition, mean relaxation showed smaller fluctuations. This shows that the system has been able to improve its precision (i.e., retrieve a smaller set of candidate cases) without harming its accuracy. This rescues the system's overall retrieval time.

4 Conclusions

This system has proved very effective. It was implemented quickly and made a positive return in investment in its first year [Gardingen & Watson, 98]. However, under increasing use its performance started to become an issue that was reducing its effectiveness. Several measures were taken to improve performance including using a faster database to store the cases, improving the way database access was managed on the server side and improving the way the query relaxation algorithm worked. This later improvement was significant because it involved the use of an introspective learning a accuracy of the algorithm that learns by how much to relax case features during the relaxation process. Our experimental findings show that although the learning algorithm does not improve the accuracy of the system it does improve its precision. For the future we would like to find a way to make the learning algorithm consider the interaction of case features. Currently features are considered in isolation which is obviously a weakness since there a re certainly both strong and weak dependencies between case features.

References

[Doyle, et al., 1998] Doyle, M., Ferrario, M.A, Hayes, C., Cunningham, P., Smyth, B. (1998). CBR Net: Smart Technology Over a Network, Internal Report Trinity College Dublin, TCD-CS-1998-07.
http://www.cs.tcd.ie/Padraig.Cunningham/publications.html

[Gardingen & Watson, 1998]. Gardingen, D. & Watson, I. (1998). A Web Based Case-Based Reasoning System for HVAC Sales Support. In, Applications & Innovations in Expert Systems VI. Milne, R., Macintosh, A. & Bramer, M. (Eds.), pp. 11- 23. Springer, London. ISBN 1-85233-087-2

[Hammond et al., 1996] Hammond, K.J., Burke, R., & Schmitt, K. (1996). A Case-Based Approach to Knowledge Navigation. In, Case-Based Reasoning: Experiences, Lessons, & Future Directions. Leake, D.B. (Ed.) pp.125-136. AAAI Press/The MIT Press Menlo Park, Calif., US.

[Hanney & Keane, 1996] Hanney, K. & Keane, M. (1996). Learning Adaptation Rules From a Case-Base. Advances in Case-Based Reasoning, Smith, I. & Faltings, B. (Eds.) Lecture Notes in AI # 1168 pp.179-192. Springer-Verlag, Berlin.

[Hayes, et al., 1998] Hayes, C., Doyle, M., Cunningham, P., (1998). Distributed CBR Using XML, Internal Report Trinity College Dublin, TCD--CS-1998-06.
http://www.cs.tcd.ie/Padraig.Cunningham/publications.html

[Kamp et al., 1998] Kamp, G. Lange, S. & Globig, C. (1998). Case-Based Reasoning Technology: Related Areas. In, Case-Based Reasoning Technology: From Foundations to Application. Lenz, M. et al (Eds.) LNAI # 1400 pp.325-351. Springer-Verlag, Berlin.

[Kitano & Shimazu, 1996] Kitano, H., & Shimazu, H. (1996). The Experience Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control. In, Case-Based Reasoning: Experiences, Lessons, & Future Directions. Leake, D.B. (Ed.) pp.235-268. AAAI Press/The MIT Press Menlo Park,Calif., US.

[Leake, et al., 1995] Leake, D.B., Kinley, A. & Wilson, D. (1995). Learning to Improve Case Adaptation by Introspective Reasoning and CBR. In, Case-Based Reasoning Research & Development, Veloso, M. & Aamodt, A. (Eds.), Lecture Notes in AI # 1010, pp.229-240. Springer-Verlag, Berlin.

[Markovitch & Scott, 1993] Markovitch, S. & Scott, P.D. (1993). Information Filtering. Selection mechanisms in Learning Systems. Machine Learning, 10, pp.113-151.

[Richter, 1998] Richter, M. (1998). Introduction - the basic concepts of CBR. In, Case-Based Reasoning Technology: from foundations to applications. Lenz, M., Bartsch-Sporl, B., Burkhard. H-D. & Wess, S. (Eds.). Lecture Notes In AI # 1400 Springer-Verlag, Berlin.

[Shimazu, 1998] Shimazu, H. (1998). Textual Case-Based Reasoning System using XML on the World-Wide Web. To appear in the Proc. Of the 4th European Workshop on CBR (EWCBR98), Springer Verlag LNAI.

[Smyth & Cunningham, 1996]. Smyth, B., & Cunningham,). (1996). The Utility Problem Analysed: A Case-Based Reasoning Perspective. Advances in Case-Based Reasoning, Smith, I. & Faltings, B. (Eds.) Lecture Notes in AI # 1168 pp.392-399. Springer-Verlag, Berlin.

[Watson, 1997] Watson, I. (1997). Applying Case-Based Reasoning: techniques for enterprise systems. Morgan Kaufmann Publishers Inc. San Francisco, CA.

[Watson, 1998] Watson, I. (1998). Case-Based Reasoning is a Methodology not a Technology. Research & Development in Expert Systems XV, Mile, R., Moulton, M. & Bramer, M. (Eds.), pp.213-223. Springer-Verlag, London.

[Watson & Gardingen, 1999] Watson, I. & Gardingen, D. (1999). A Distributed Case-Based Reasoning Application for Engineering Sales Support. IJCAI'99 31st. July - 6th August 1999, Vol. 1: pp. 600-605. Morgan Kaufmann Publishers Inc.

[Wilke, et al., 1998] Wilke, W. Lenz, M. Wess, S. (1998). Intelligent Sales Support with CBR. In, Case-Based Reasoning Technology: from foundations to applications. Lenz, M., Bartsch-Sporl, B., Burkhard. H-D. & Wess, S. (Eds.). Lecture Notes In AI # 1400 91-113. Springer-Verlag, Berlin.

[WWW Consortium 1997] World Wide Web Consortium, (1997). Extensible Markup Language 1.0, recommendation by W3C: www.w3.org/TR/PR-xml-971208

[WWW Consortium] World Wide Web Consortium home page: www.w3.org

further information on all aspect of case-based reasoning can be found at www.ai-cbr.org