Combining Strict Matching and Similarity Assessment for Retrieving Appropriate Cases Efficiently

Toshikazu Tanaka and Naomichi Sueda
Systems and Software Engineering Laboratory,
Research and Development Center, Toshiba Corporation
70 Yanagicho, Saiwaiku, Kawasaki, Kanagawa 210 Japan
E-mail: ttanaka@ssel.toshiba.co.jp

Abstract

It is essential for case-based reasoning (CBR) systems to access truly relevant cases efficiently. Similarity assessment adopted by many CBR systems needs performance improvement, especially if the case library of a CBR system consists of a database in the target domain which is not equipped with suitable indexing for the CBR system. In consideration of this problem, we are exploring use of strict matching, a conventional information retrieval technology, to screen out irrelevant cases effectively and efficiently before similarity assessment.

In this paper, we first clarify the need for combining strict matching and similarity assessment by analyzing two CBR systems we developed: Sales Estimation System and Motor Design Support System. Then, we propose a method for combining the two retrieval techniques, Similarity Assessment with Strict Screening, which first performs strict matching to reduce the size of a set of candidate cases subject to similarity assessment. Central to the method is an algorithm which is complete in that strict matching does not screen out any relevant case which similarity assessment would select.

1 Introduction

It is essential for case-based reasoning (CBR) systems to access truly relevant cases efficiently. CBR systems must retrieve appropriate cases carefully because retrieved cases directly affect the following modification or adaptation process and eventually the solution of a given problem. Too elaborate selection processes, however, tend to violate performance requirements imposed by applications; so an efficient retrieval method is needed. In addition, such a retrieval method must facilitate some similarity assessment in order to access cases which do not exactly match but similar in some way to the problem.

For this purpose, a group of two-phase case retrieval methods which perform similarity assessment based on surface features and then detailed validation of retrieved cases, has been developed in the CBR community. A representative system would be Simoudis' Help Desk system applying CBR[4]. They propose a retrieval process which first performs similarity assessment based on surface features and then performs *model-based* validation of retrieved cases using the contents and knowledge of *probes*, tests that can be used for validation. Such two-phase retrieval processes achieve high recall and high precision with favorable efficiency.

Similarity assessment, however, needs further performance improvement because it is essentially a global process which compares some similarity metric over (sometimes all) stored cases. Hereinafter, we will use *similarity assessment* to mean any retrieval method which selects one or more pieces of information by assessing some metric between a given problem description and pieces of information stored in case (data) bases. As suggested by [4], one way to make efficient the surface feature-based retrieval operation is to organize cases into a generalization hierarchy, a graph structure whose internal nodes are generalizations over a set of cases, by an algorithm such as UNIMEM[3]. Generating such a hierarchy and attaching it to a case library is not always a viable way, especially in the situation where the case library of a CBR system consists of an existing database in the target domain which is not equipped with suitable indexing for the CBR system.

In consideration of this problem, we are exploring the use of strict matching, a conventional information retrieval technology, to screen out irrelevant cases effectively and efficiently before similarity assessment. We will call this method *Similarity Assessment with Strict Screening*. We will use *strict matching* to mean any retrieval method which retrieves a piece of information (case, data, record, etc.) if it satisfies some logical condition (to be called *strict index*). A typical strict matching is a database retrieval using the SQL query language. When strict matching is used in a CBR system, the system is not given such a strict index but must generate one according to a given problem description.

In Section 2, we clarify the need for combining strict matching and similarity assessment by analyzing two CBR systems we developed: the Sales Estimation System and the Motor Design Support System. In section

3, we propose a method for combining the two retrieval methods. Central to the method is an algorithm which is complete in that strict matching does not screen out any relevant case which similarity assessment would prefer if strict matching were not performed.

2 Analyzing Prior Cases for Clarifying the Need to Combine Strict Matching and Similarity Assessment

In this section, we discuss the need for combining strict matching and similarity assessment, using the two systems we have developed.

2.1 Sales Estimation System

Overview of the system. The Sales Estimation System [2] estimates the annual sales of a proposed new franchise store. We adopted CBR because actual value of sales recorded in cases can be used to make estimation accurate. We developed a prototype system in the domain of electric appliances stores to validate the accuracy of sales estimates, and the result was good, i.e. 85% of the trials fell within ±15% relative error. The features of the system include (a) use of similar existing stores for reliable actual sales, (b) extraction of abstract features from input conditions using neural networks and (c) learning of case modification rules by clustering existing cases.

The flow of the system is as follows (see Figure 1).

1. *Feature extraction.* Eight abstract features are extracted from about twenty-five detailed features given by the user.
2. *Case retrieval.* These 8 abstract features of the target store are matched against those of the cases. Similarity assessment is performed, and the most similar case is selected.
3. *Case Modification.* The sales of the selected case is modified according to analogy detected between the target store and the selected case.

The input to the system is a list of about 25 conditions of a new target store and the place of deployment, such as floor area size, parking lot size, population, land area, average income, and distance from the closest train station. They are summarized into 8 abstract features such as economic level of neighborhood, traffic convenience, and population type (summarizing young or old, single or married, white or blue color workers). This summarization is done by eight neural networks associated with each abstract type, and these networks are trained by stored cases so that feature extraction is consistent with the following CBR processes: retrieval and modification. Each case records about 25 conditions, 8 abstract types and annual sales.

A similar existing store is retrieved by assessing a similarity metric defined as weighted sum of feature-wise similarity metrics which are arithmetic formulae. The annual sales of the retrieved case is modified, considering the differences between the new store being planned and the existing store. Domain-specific knowledge is used for similarity metrics and case modification rules.

Analysis of case retrieval. We believe CBR is suitable for estimation problems in such domains as sales of franchise chains, production costs for order-based products, and number of participants to concerts. If CBR is to be applied to such problems in a realistic context, we will be faced with the following issues.

- *Database for the case library.* Although a small case library was handtailored for the prototype Sales Estimation System, it is practical to use a database in any existing information processing system for the case library because automatic accumulation of cases is expected. A problem is that there is no semantic indexing for such a database.
- *Need for high performance.* Because such a database often includes many cases (consider a database which records daily sales gathered from POS (point of sales) terminals), high performance retrieval is required. Even for systems where the number of cases is low such as Sales Estimation System we presented, high performance is essential because such an estimation system will be used in an interactive manner; for example, the user might want to retrieve several cases with different indices (i.e. from different viewpoints).
- *Difficulty in setting strict indices.* In the latest version, we incorporated a screening mechanism using strict matching before similarity assessment. This mechanism uses hierarchies of domain-level concepts so that the user can easily set meaningful indices. For example, the system uses the hierarchy in Figure 2 to generate a retrieval condition like "place: is_a* Kanto" for the input "place: Yokohama", thus allowing cases for stores located in Tokyo, Kawasaki, Saitama, Chiba as well as Yokohama to be selected. Although this mechanism was found to be useful, it is difficult to determine to which level to generalize input values. This difficulty arises especially if there are several features equipped with such hierarchies (e.g. places, store types, etc.) and different weights are given to those features.

2.2 Motor Design Support System

Overview of the system. The Motor Design Support System[5][6] assists a user, a motor designer, in designing industrial-use induction motors. As shown in Figure 3, the system, operating on an engineering workstation (EWS) AS4000 (or Sun4), accepts client requirements and generates a component list, or a list of

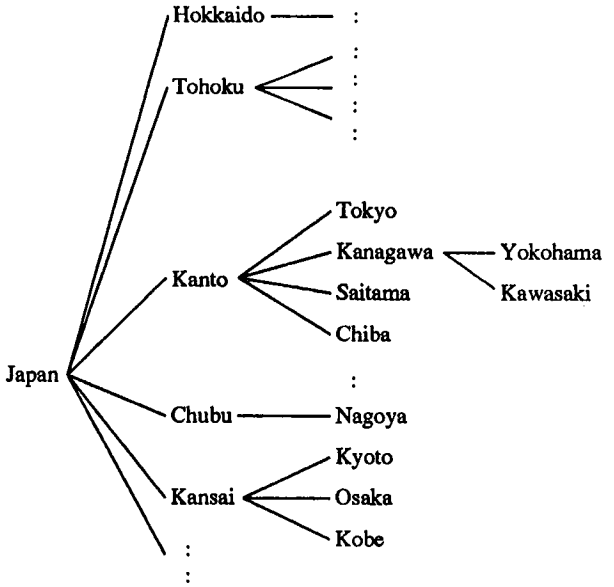


Figure 2: Example of Conceptual Hierarchy (Locations)

pairs of component item and schematic diagram (drawing) ID. The workstation is connected on-line to a TOSFile, a documentation system which uses optical disks to store image data. Reusable drawings are stored on the TOSFile.

The case library contains successful cases, each case consisting of a pair of a requirement list (problem) and a component list (solution). The knowledge base contains domain-specific expertise, mainly a design model which maps requirement differences into components and vice versa. Some examples of mapping knowledge include: if voltage becomes higher, then thicker feeders are needed; if feeders become thicker, then a bigger hole in a terminal box is needed.

The process flow of the system is as follows (see Figure 3).

1. *Reduction (including base case retrieval).* The reduction module retrieves a *base case*, a case which satisfies important requirement items of a given problem. The differences in requirements between the input and the base case are mapped into components to modify, using the design model.
2. *Decomposition.* Components to modify are decomposed into several groups, thus generating a set of subproblems.
3. *Completions (including part case retrievals).* For each subproblem, a *part case*, a case which can provide parts necessary but lacking in the base case, is retrieved and used to solve the subproblem (thus the name completion). Contrary to base case retrieval, only those requirements which affect the necessary parts are used as retrieval indices.

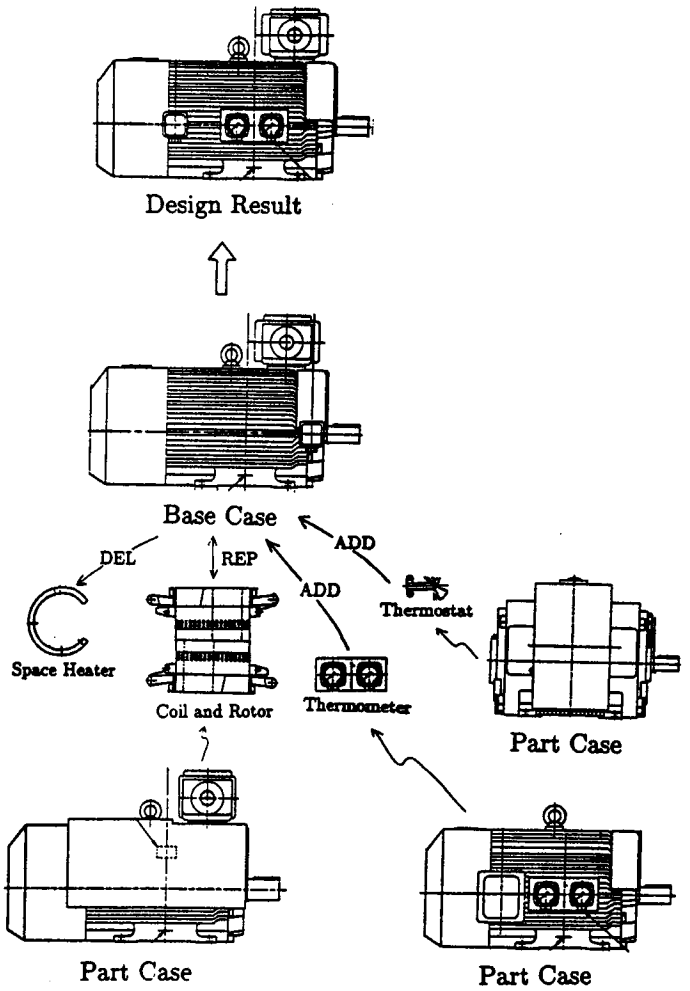


Figure 4: Example of Motor Design Process

4. *Composition.* Components taken from part cases are merged into the base case by using the *add*, *delete* and *replace* operators, thus generating a component list, a design result.
5. *Diagram retrieval.* The user may retrieve actual diagrams on TOSFile to verify the design result.

This design process is illustrated in Figure 4. The base case needs modification in four group of parts: (a) a space heater and associated terminal box, (b) a coil and a rotor, (c) a thermometer, and (d) a thermostat and its terminal box.

Analysis of case retrieval. The following three retrieval methods are used in this system.

- *Strict matching.* It uses two requirement items, electrical and mechanical types, to choose candidates from thousands of cases.
- *Similarity assessment.* It uses the similarity metric, weighted sum of item-wise similarity metrics.

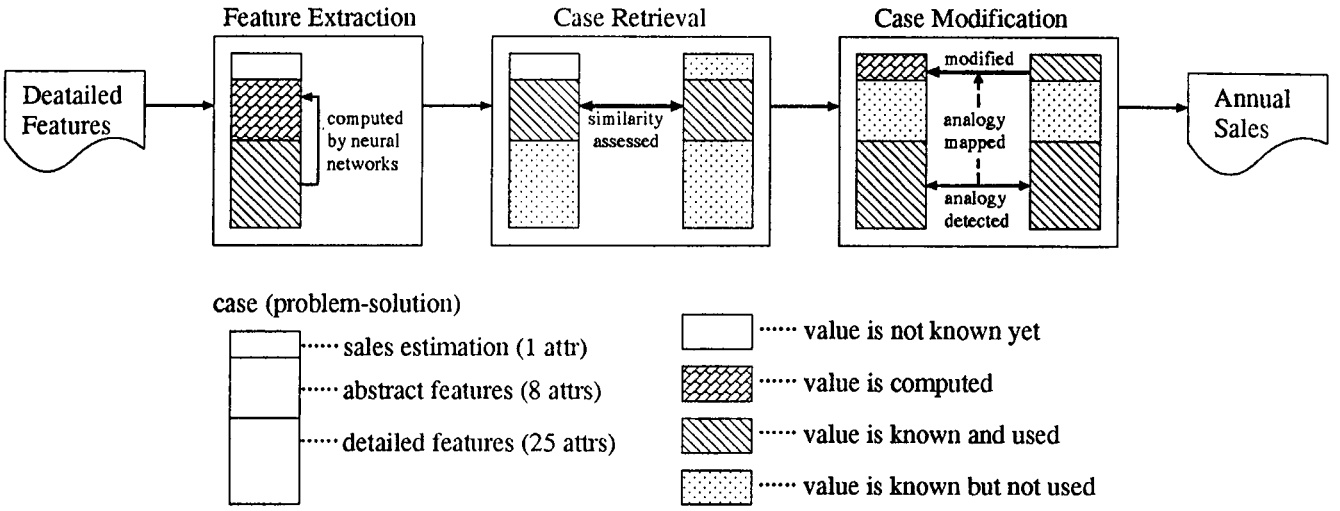


Figure 1: Sales Estimation System

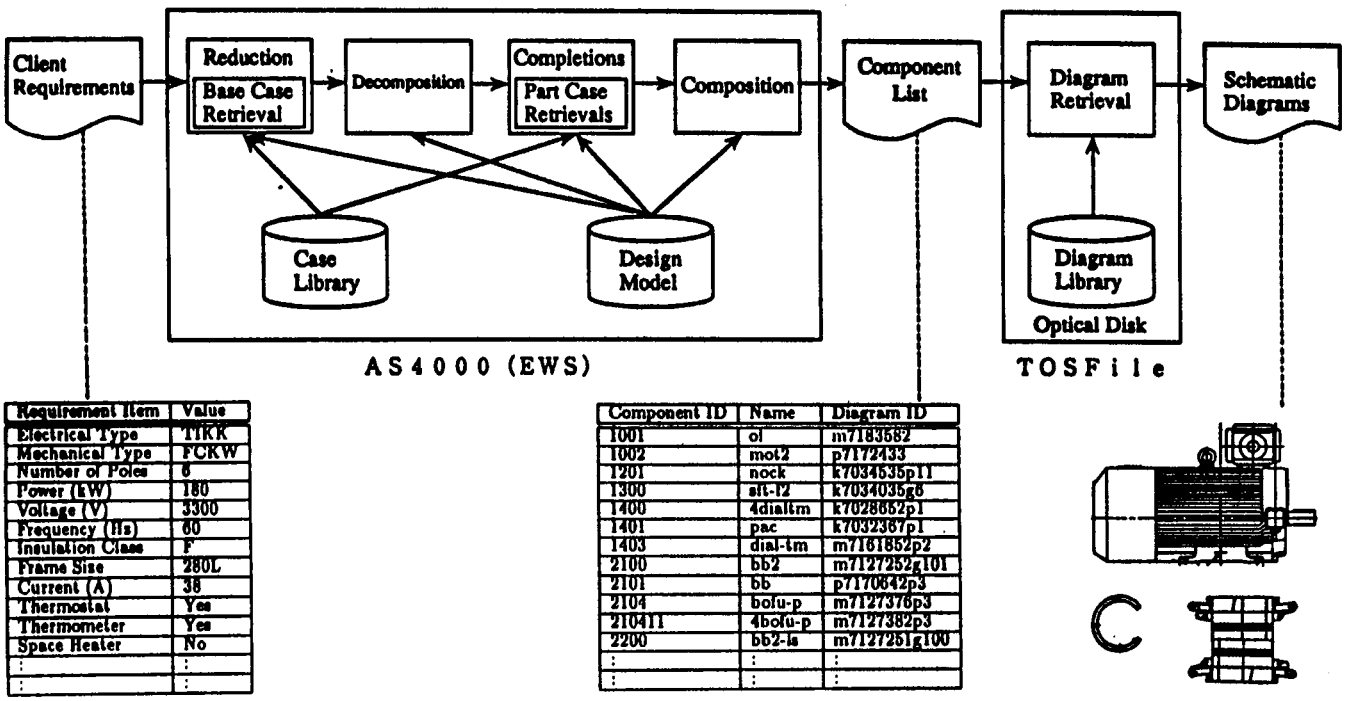


Figure 3: Motor Design Support System

Weights can be dynamically assigned according to the input requirements. Similarity metrics are asymmetric (deletion is easier than addition). Similarity assessment of this system is done in three steps, reducing the candidate set step by step.

- *Detail assessment.* A small set of remaining candidates are subject to pairwise comparison. Domain-specific criteria is used (e.g. standard design is preferred.)

Not all requirement items go through these three retrieval methods. The electrical and mechanical types are only used in strict matching while requirement items such as power, voltage and frequency are only used in similarity assessment. This is because, currently, strict matching using the electrical and mechanical types is sufficiently effective in reducing the number of candidate cases.

If CBR is to be applied to such problems in a realistic context, we will be faced with the following issues.

- *Database for the case library.* One of the requirements for the system to be adopted on the factory floor is to use data accumulated in the existing CAD/CAM system. Therefore, it is practical to use a database in the existing information processing system for the case library. The problem is also that there is no semantic indexing.
- *Need for high performance.* High performance retrievals are required not only due to the fact that the existing database includes many cases (in the order of thousands), but also due to the fact that multiple cases (*base case* and *part cases* as mentioned earlier) are retrieved during a single problem solving session.
- *Difficulty in setting strict indices.* If the screening mechanism incorporated in the Sales Estimation System is to be used in this system, the problem of how to set indices for attributes with numerical values (voltage, power, etc.) arises; it is difficult to set an appropriate interval, considering the relative importance (weight) of attributes.

2.3 Summary of Analysis

Before going on to Section 3 where the retrieval method is described, we restate the needs we found out by developing the two systems.

- Use of an existing database for the case library.
- Need for high performance.
- Easy or automatic way for setting strict indices.

First, we review differences between strict matching and similarity assessment (see Table 1). Strict matching is syntactic in that a strict index is a logical formula whose elemental tests are typically character matching while similarity assessment uses some similarity metric which incorporates domain-specific knowledge. Algorithms for strict matching can be made local in that they only have to access a portion of database if the case library is equipped with an appropriate indexing scheme, as is the case for database management systems. On the other hand, algorithms for similarity assessment are difficult to make local because they must compare the similarity metrics of competing cases. The relative nature of similarity assessment makes indexing for similarity assessment difficult. This difference directly affects the performance of the two retrieval methods. Table 1 suggests the need for reducing the size of a case set subject to similarity assessment using strict matching.

Point of View	Strict Matching	Similarity Assessment
Nature	Syntactic	Semantic
Algorithm	Local	Global
Indexing	Easy	Hard
Efficiency	Fast	Slow

Table 1: Comparison between Strict Matching and Similarity Assessment

Viewpoint	Problem part	Solution part
Contents	Problem	Solution
Objective	Selection	Modification
Access	Frequent	Not frequent
Representation	Attr-value pairs	Divergent
Data length	Fixed	Variable

Table 2: Division into Problem and Solution parts

Second, we restrict problem representation. Table 2 shows the division of a case by into two parts, referred to as the problem and solution parts, using the contents, objective of retrieval, access frequency, suitable representation, and data length for the division criteria. This table suggests that the problem part (a) can be represented by a set of attribute-value pairs and (b) can be easily stored in conventional databases, thus reducing the effort needed to construct a case base.

Once we have restricted problem representation to attribute-value pairs, we classify attributes into four classes in terms of use for strict matching and similarity assessment (see Table 3). The most important attribute is one that is subject to both strict matching and similarity assessment.

From the analysis, we can conclude that a preferable retrieval method should

- first retrieve candidate cases stored in a database using strict matching, and
- then perform similarity assessment (and more detailed validation) over the candidate set.

SM	SA	The Role of Each Retrieval Method
O	O	Strict matching is used for efficiency. We concentrate on this class.
O	X	Strict matching serves as final screening. Careful setting of strict indices needed.
X	O	No screening before similarity assessment. Case library is small, or some other attributes screen a case set.
X	X	To be checked in a later phase, or for a management use (e.g. creation date).

SM: strict matching, SA: similarity assessment
O: used, X: not used

Table 3: Classification of Attributes

3 The Similarity Assessment with Strict Screening Method

Through the analysis in Section 2, we clarified the need for using strict matching to reduce the size of a case set subject to similarity assessment. In this section, we propose a retrieval method which first performs strict matching and then similarity assessment. We will call this method *Similarity Assessment with Strict Screening.*

3.1 Preliminaries

Before presenting the retrieval algorithm, we define several terms and concepts. We assume that retrieval occurs on the fixed set of attributes A_i of domain D_i. We refer to the Cartesian product $D_1 \times \ldots \times D_n$ as *feature space*, denoted $\mathcal{F}$. Problems and cases are represented as points in the space. Those attributes which are subject to both strict matching and similarity assessment are the target of the Similarity Assessment with Strict Screening method we are going to explain.

Strict matching. Strict matching is a retrieval method which uses a *strict index*, or some logical condition to match against target data point. We call a strict index, I, *proper* with respect to a point P if P matches I. From now on, we will only deal with proper strict indices.

Similarity assessment. We limit the similarity assessment function *sim* and the associated distance function *dist* as follows.

$$sim(P,Q) = \sum_i wt_i(P) \times sim_i(P,Q)$$

where wt_i and sim_i is a weighting function and similarity function for attribute A_i, respectively. The range of these functions is $[0,1]$.

$$dist(P,Q) = \sqrt{\sum_i dist_i(P,Q)^2},$$

where $dist_i(P,Q) = wt_i(P) \times (1 - sim_i(P,Q))$.

The metric *sim* might be natural and used often, and *dist* might be somewhat peculiar. However, *dist* is easy to depict in the feature space; and we use it in the following discussion for simplicity of explanation.

Neighborhoods. We define several kinds of *neighborhood* of a point P, a subspace of the feature space $\mathcal{F}$ surrounding a point P.

- A *strict neighborhood* of a point P with respect to a strict index I is defined as
  $N_{str}(P,I) \stackrel{\text{def}}{=} \{ Q \in \mathcal{F} \mid I(Q) \text{ is true } \}$.
  Since we only consider proper strict indeces, $P \in N_{str}(P,I)$; so we denote $N_{str}(P,I)$ as $N_{str}(I)$.
- A *similar neighborhood* of a point P of size δ is defined as
  $N_{sim}(P,\delta) \stackrel{\text{def}}{=} \{ Q \in \mathcal{F} \mid dist(P,Q) \leq \delta \}$.

If we restrict strict index I to be a conjunction of index patterns, each imposing a test on one attribute, then a strict neighborhood can be conceived as a rectangular area or subspace in $\mathcal{F}$. A similar neighborhood can be conceived as a circle or sphere in $\mathcal{F}$.

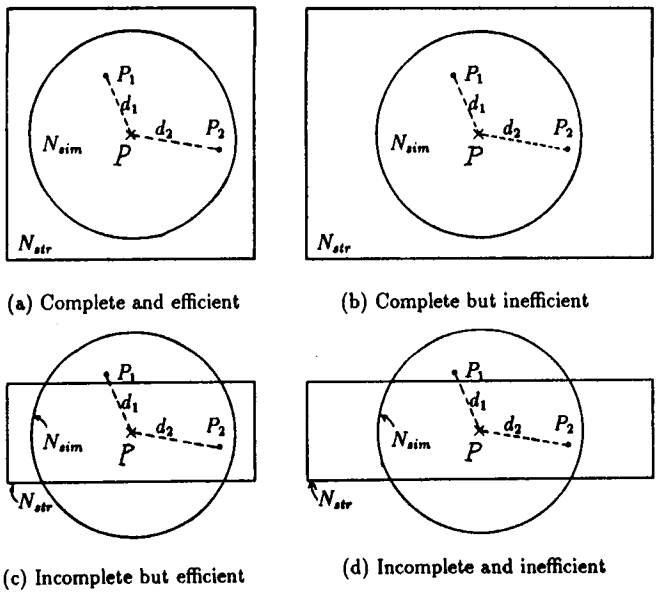
3.2 Completeness of the Similarity Assessment with Strict Screening

We define the Similarity Assessment with Strict Screening method to be *incomplete* if there exists some case point which is screened out by strict matching but included by similarity assessment without strict matching. Otherwise, it is *complete*.

Lemma. The Similarity Assessment with Strict Screening method is complete if and only if $N_{str} \supseteq N_{sim}$.

Proof. For any point Q that is included in N_{sim} and any point Q' that is not included in N_{sim}, $dist(P,Q) \leq dist(P,Q')$ holds.
($\Longleftarrow$) Therefore, if $N_{sim} \subseteq N_{str}$, limiting domain of similarity assessment to N_{str} does not preclude any candidate which would be closer to P than some point in N_{sim}, which guarantees completeness.
($\Longrightarrow$) If $N_{sim} \subseteq N_{str}$ does not hold, there is some point Q that is included in N_{sim} but precluded from N_{str}. If such a point is the closest to P among other points in $N_{str} \cap N_{sim}$, the result of similarity assessment will be different, depending on whether strict matching is done or not, thus violating completeness. □

Figure 5 illustrates situations stated in the above proof. In the figure, two attribute ares depicted (horizontally and vertically). (a) shows a situation in which $N_{str} \supseteq N_{sim}$ holds. The domain of similarity assessment, i.e. N_{sim}, is not narrowed by N_{str}. On the other hand, in (c), $N_{str} \supseteq N_{sim}$ does not hold. There are two



(a) Complete and efficient

(b) Complete but inefficient

(c) Incomplete but efficient

(d) Incomplete and inefficient

Figure 5: Completeness and Efficiency of Similarity Assessment with Strict Screening

points in N_{sim}: P_1 and P_2. P_1 is not in N_{str} while P_2 is in N_{str}, and $d_1 < d_2$ ($d_i = dist(P, P_i)$, $i = 1, 2$). If strict matching is performed, P_1 is precluded, and P_2 is selected. If strict matching is not performed, P_1 as well as P_2 becomes subject to similarity matching, and will be favored over P_2 because $d_1 < d_2$. Note that since completeness is a property of the Similarity Assessment with Strict Screening method and not that of the contents (i.e. actual population) of the feature space, such a situation does not always occur.

Figure 5 also addresses efficiency. The situation of (b) is complete, but strict matching is not so effective since it does not screen out irrelevant points sufficiently. (d) is neither complete nor efficient. Therefore, it is important to generate a strict index which defines a strict neighborhood which minimally surrounds an intended similar neighborhood.

3.3 Complete and Efficient Algorithm for Similarity Assessment with Strict Screening

We present a skeleton of our retrieval algorithm which is both complete and efficient. This algorithm takes four input parameters: P (a given problem), C (an initial case set), n_{min} and n_{max} (criteria for an appropriate size for a set of candidate cases). It goes through an iterative process of setting an appropriate similarity threshold δ and the associated minimal strict index I in response to the actual population of $N_{str}(I)$ (Step 2 and 3). A sequence of strict matching and similarity assessment is executed only once (Step 4 and 5).

Algorithm `retrieve_case`(P, C, n_{min}, n_{max})

1. Set some $\delta_0 > 0$ which defines the similar neighborhood $N_{sim}(P, \delta_0)$ to δ.
2. $I \leftarrow$ `generate_min_str_idx`(P, δ).
   /* Generate a strict index I so that the strict */
   /* neighborhood $N_{str}(I)$ defined by I minimally */
   /* surrounds similar neighborhood $N_{sim}(P, \delta)$. */
3. $n \leftarrow$ `get_population`$(N_{str}(I), C)$.
   /* Get the population of $N_{str}(I)$. */
   (a) If $n_{max} < n$, then make δ smaller and go to Step 2.
   (b) If $n < n_{min}$, then make δ bigger and go to Step 2.
   (c) If $n_{min} \leq n \leq n_{max}$, then go to Step 4.
4. $C_1 \leftarrow$ `strict_matching`(I, C).
   /* Get a set of cases which match I */
5. $C_2 \leftarrow$ `similarity_assessment`(δ, C_1).
   /* Reduce the case set by similarity assessment. */

3.4 Some Implementation Considerations

There are three points which must be considered when implementing the above algorithm:

- How to set and change a threshold distance δ.
- How to generate a strict index I whose strict neighborhood minimally surrounds the given similar neighborhood; in other words, how to implement the `generate_min_str_idx` procedure.
- How to get the population of a strict neighborhood N_{str}; in other words, how to implement the `get_population` procedure.

In the following, we discuss the above implementation issues.

Setting a threshold distance δ. If a case base is rather uniformly populated and a population is known, then a constant δ can be used. If a case base is rather uniformly populated but a population is not known, some automatic or off-line learning about δ should be done. If a case base is unevenly populated, domain-specific knowledge is needed; such knowledge should be learned. One way would be to construct a decision tree which classifies the input problem to estimate the population density.

Generating a minimally surrounding strict index. In order to simplify the following discussion, we limit the syntax of the retrieval condition used in strict matching as follows.

```
<strict_index>  ::=  (and <conjunct>+)
<conjunct>      ::=  (<attr> is_a* <concept>)
                |    (<attr> interval
                          <lower> <upper>)
```

The two kinds of conjuncts correspond to the conceptual test used in the Sales Estimation System and the numeric test considered for the Motor Design Support System. A pattern (<attr> is_a* <concept>) is intended for an attribute whose values constitute a hierarchy of concepts, and means that attribute <attr>'s value of a given point P should be a descendant of <concept>. A pattern (<attr> interval <lower> <upper>) checks for the value of a numeric attribute, and holds when attribute <attr>'s value of a given point P is between <lower> and <upper>. There are two strict index establishing methods for the above syntax.

- Generalize/specialize the concept in a pattern:
  (A is_a* <concept>) $\rightarrow$
  (A is_a* <ascendant>) or
  (A is_a* <descendant>).
  <ascendant> (<descendant>) is chosen considering minimum similarity between a problem value and each child of <ascendant> (<descendant>).
- Widen/narrow the numeric interval in a pattern:
  (A interval <lower> <upper>) $\rightarrow$
  (A interval <new_lower> <new_upper>)
  so that
  $dist_i(p_i, new\_lower) = \delta$ and
  $dist_i(p_i, new\_upper) = \delta$,
  where p_i is P's value of attribute A_i.

For the conceptual case, the appropriate level of generalization (specialization) can be determined by searching a conceptual hierarchy and checking maximal similarity. For the numerical case, the inverse functions of $dist_i$'s are needed to determine a new interval for a numeric attribute.

Estimating the population of a strict neighborhood. Direct counting of matched data is apparently time-consuming; and such statistical data should be maintained by the case (data) base. For example, a method proposed in [1] which uses decision trees to summarize statistics about the content of a database, could be used for this purpose.

4 Conclusion

In this paper, we explored a method for combining strict matching and similarity assessment so that irrelevant cases can be efficiently screened out whereas relevant cases are inadvertently not dropped. The need for combining the two retrieval methods from two different technology fields surfaced from the analysis of two CBR systems we developed:

- the Sales Estimation System, and
- the Motor Design Support System.

Based on the results of the analysis, we proposed a generic retrieval algorithm which performs a sequence of strict matching and similarity assessment. The Similarity Assessment with Strict Screening algorithm

- is complete in that it does not screen out relevant cases during strict matching,
- uses similarity-assessment knowledge (weight and similarity) to establish appropriate strict indices automatically,
- tunes the similarity threshold δ in response to the estimation of the population of cases, and
- is suitable when the conventional database is used as a case library and also when attaching CBR-oriented indexing to it is difficult.

We are planning to implement this algorithm as a part of the CBR Shell which is now under development.

References

[1] Meng Chang Chen and Lawrence McNamee. Summary data estimation using decision tree. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*. AAAI Press / The MIT Press, 1991.

[2] Chieko Kobayashi, Toshikazu Tanaka, and Naomichi Sueda. Learning from modification failure in case-based sales estimation system. In *Proceedings of 6th National Conference of Japan Society for Artificial Intelligence*, pages 805–808, 1992.

[3] Michael Lebowitz. Experiments with incremental concept formation: UNIMEM. In *Machine Learning*, 2:103–138, 1987.

[4] Evangelos Simoudis and James S. Miller. Application of CBR to help desk applications. In *Procs. of DARPA Case-Based Reasoning Workshop*, pages 25–36, 1991.

[5] Toshikazu Tanaka, Masakazu Hattori, and N. Sueda. Use of multiple cases in case-based design. In *Proceedings of Conference on Artificial Intelligence for Applications (CAIA92)*, pages 233–239, 1992.

[6] eds. Toshiba. Mechanical design support system applying case-based reasoning. In *Science and Technology Highlights 1992: A Special Issue of the Toshiba Review*, page 3, 1992.