Investigating different Methods for efficient Retrieval of Generalized Cases

Rainer Maximini Alexander Tartakovski
Ralph Bergmann
University of Hildesheim, Data- and Knowledge Management Group
PO-Box 101363, D-31113 Hildesheim, Germany
{r_maximi|tartakov|bergmann}@dwm.uni-hildesheim.de

Abstract: Generalized cases are cases that cover a subspace rather than a point in the space, spanned by the case's attributes and can be represented by a set of constraints between them. For such representations, the similarity assessment between a point query and generalized cases is a difficult problem that is addressed in this paper. The task is to determine the distance (or the related similarity) between the point query and the closest point of each area covered by a generalized case. We present three ideas how this problem can be solved: by using methods of mathematical optimization, by a sampling conversion to point cases, and by using techniques from 3D real time computer graphics.

1 Introduction

In CBR applications, the traditional concept of a case is that of a point in the space spanned by the case's attributes. This space is called *problem–solution space* when the attributes can unambiguously be related to the problem description or the solution description, respectively. In other applications, each query fixes the current problem attributes and all non-specified ones are automatically the solution attributes. In this case, like in the superordinate one, the space is just called *attribute space*. Irrespective, during case-based problem solving, cases are retrieved from a case base using a similarity function, which compares the case descriptions with the current query.

Driven by examinations of several new applications, we proposed the concept of *generalized cases* [BVW99, BV99, Ber02]. A generalized case covers not only one point of the attribute space, but a whole subspace of it. A single generalized case immediately provides solutions to a set of closely related problems rather than to one single problem only. The solutions a generalized case represents are very close to each other; basically they should be considered as (slight) variations of the same principle solution. In general, a single generalized case can be seen as an implicit representation of a (possibly infinite) set of traditional "point cases". We assume, that the similarity to a generalized case is the similarity to the most similar point of the case.

We also want to make clear, that the idea of generalizing cases is not a radically new con-

cept. It was already implicitly present since the very beginning of CBR and instance-based learning research [Kol80, Bar89, Sal91]. However, in this paper we explore a more formal and systematic view on generalized cases by using constraints to express the dependencies between several attributes. This partially covers also the above mentioned related work.

1.1 An Application: Representing Electronic Design IPs

Increasingly, electronics companies integrate *Intellectual Properties* (IPs) from third parties within their complex electronic systems. An IP is a design object whose major value comes from the skill of its producer [Lew97], and a redesign of it would consume significant time. However, a designer who wants to reuse designs from the past must have a lot of experience and knowledge about existing designs, in order to be able to find candidates that are suitable for reuse in his/her specific new situation. Currently, searching electronic IP databases can be an extremely time consuming task because of two main reasons: On the one hand, the public-domain documentation of IPs is very restricted and on the other hand there are currently no intelligent tools to support the designer in deciding whether a given IP from a database meets (or at least comes close to) the specification of his/her new application. This is one objective of the current project *IPQ: IP Qualification for Efficient Design Reuse*[1] funded by the German Ministry of Education and Research (BMBF) and the related European Medea project *ToolIP: Tools and Methods for IP*[2].

IPs usually span a design space because they are descriptions of flexible designs that have to be synthesized to hardware before they actually can be used. The behavior of the final hardware depends on a number of *parameters* of the original design description. The valid value combinations for these parameters are constrained by different criteria for each IP.

1.2 The Research Problem: Similarity Assessment and Retrieval

The important basic research issues involved when using generalized cases are related to representation formalisms, similarity assessment, and retrieval. One serious complication when studying these issues is that they are strongly connected with each other. Depending on the expressiveness of the representation formalism used for generalized cases, similarity assessment is getting computationally more difficult, which also impacts the overall computational effort for retrieval from a large case base.

In this paper we do not want to restrict the sets of constraints used to represent generalized cases; the only limitation is that they must be computable. We will present retrieval ideas which have not been researched or evaluated so far and probably may have special restrictions.

[1] IPQ Project (12/2000 - 11/2003). Partners: AMD, Fraunhofer Institute for Integrated Circuits, FZI Karlsruhe, Infineon Technologies, Siemens, Sciworx, Empolis, Thomson Multi Media, TU Chemnitz, University of Hildesheim, University of Kaiserslautern, and University of Paderborn. See www.ip-qualifikation.de

[2] See toolip.fzi.de for partners and further information.

Nevertheless, the main problem is the computational complexity of such restrictionless methods why it is important to develop procedures that distinguish between an offline and an online phase. During the offline phase all computations should be done which are independent from the query. Unlike, the online phase is query dependent and should be very fast to reduce the response time to the user. Consequently, the more calculations can be moved to the offline phase, the faster will be the online online phase.

1.3 Retrieval Ideas

In [MB02] and [BVW99], Mouguoie, Bergmann, and Vollrath have analyzed methods from optimization theory to solve the similarity problem for generalized cases. However, there are several alternative approaches to solve this problem. This paper points out three radically different approaches to similarity assessment and retrieval. Briefly, the methods discussed are:

Mathematically Optimization: The idea is to rank only the generalized cases to find the most similar ones. Therefore, the upper and lower bound of each case are determined, compared and if necessary refined. This refinement process can be solved with mathematical optimization techniques.

Sampling: This technique bases on the idea to transform the generalized cases into point cases by using further information like the similarity measures or user preferences. On the point cases a well known traditional retrieval method can be applied with only limited modifications.

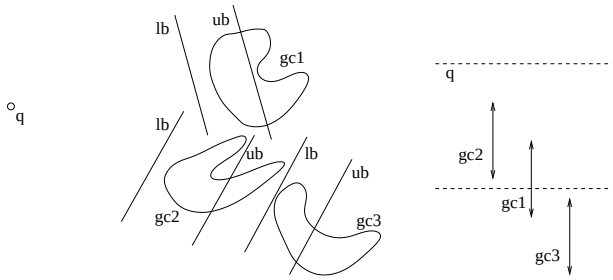
Computer Graphics: Illustrating the case base graphically (e.g. for three attributes), it seems to be possible to adapt methods from the real time 3D calculations of current computer games for the retrieval of generalized cases.

It has to be expected that each idea has its own advantages and disadvantages, dependent on the kind of attributes and constraints. Our goal is to examine each idea and to combine the methods to a general one which hopefully handles each kind of attribute and constraint.

The following three sections present each idea, as far as we have elaborated them. They should not be understood as complete descriptions of evaluated methods, but more as a survey that points to important issues of future research.

2 Methods of Mathematical Optimization

The idea is instead of calculating the exact similarity between a query and each generalized case to only rank the cases and find the most similar ones. Therefor, the upper and lower bound for each case have to be calculated in relation to the query, so that they can be compared afterwards (see figure 1). This idea is similar to the fish and shrink algorithm proposed in [Sch96].



(a) define (b) compare

Figure 1: Lower and Upper Bounds

A case can be ignored, if the lower bound of the case is higher than the upper bound of another case, e.g. in figure 1 the upper bound of gc_2 is lower than the lower bound of gc_3 why gc_3 can be removed. Until now, no statement can be made about gc_1 and gc_2, so their bounds have to be refined. The following algorithm presents the idea:

```
function findMostSimilarCases(caseBase, query)
  remainingCases = caseBase
  while(isRefinementOfBoundsPossible())
    refineBoundsForEachCase()
    removePossibleCases(remainingCases)
  return remainingCases
```

This similarity assessment problem can now be formulated as an optimization problem:

max $sim(q, x)$ s.t. $h(x) \leq 0$,

where h is a set of m functions $h_1, h_2, \ldots$ that represent the generalized case gc by $h(x) \leq 0$, i.e., $x \in gc \iff \forall i\ h_i(x) \leq 0$. In this optimization problem, max is the *objective function* to be maximized.

Thereby, the main problem is the refinement of the bounds. In [MB02], Mougouic and Bergmann analyze this idea for generalized cases that are represented through constraints over an n-dimensional Real-valued vector space. It is shown that the difficulty depends on whether the generalized case is convex or nonconvex which is defined by the constraints. For convex constraints and by usage of convex similarity measure, the Topkis-Veinott method can be easily applied to determine exactly the similarity between a query (point case) and generalized cases. If the similarity measure is nonconvex or the generalized case contains also nonconvex constraints, the problem is more difficult. For this situation an algorithm is proposed that allows to incrementally compute sequences of upper and lower bounds for the similarity and assures the convergence of the algorithm. It allows to rank generalized cases without the exact computation of all similarity measures.

The presented algorithm has two main disadvantages. Firstly, it is only analyzed for real

valued attributes and secondly, the calculation of the bounds is complex and query dependent, that means, it has to be done in the online phase.

3 Sampling

The idea of the sampling method is to convert the generalized cases to point cases, because for them a lot of efficient retrieval techniques exist and can be used. The core idea of this method is the hypothesis that a retrieval on a case base of point cases is faster than a retrieval on a case base with generalized cases, even if the point case base is larger. Of course, this hypothesis is over general and must be refined.

This is of course an approximative technique and the resulting quality mainly depends on the sampling quality and the number of result cases. The method has a big offline phase where the generalized cases are converted in an intelligent way into point cases; section 3.1 and 3.2 describe this intelligent way. The online phase is nearly the same as for retrieval with traditional cases (see figure 2). It has only to be granted that all retrieved cases originally belonged to different generalized or point cases, which is described in section 3.3.

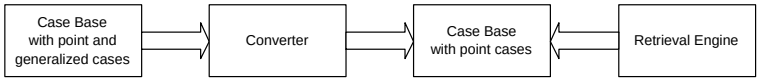


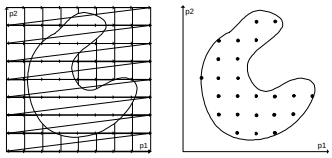
Figure 2: Basic Concept of the Sampling Method

3.1 Scanning the space

The domains of the cases' attributes span the attribute space which is an infinite space, even if only one attribute is defined as real. So the idea is to discretise this space by selecting n attribute values from each attribute domain and check if the resulting *scan point* (the value combination of all attributes) lies within a case. For example, in figure 3 attribute p_2 is scanned first and then p_1 for each selected value of p_2. If the resulting scan point lies within a generalized case, a point case is created at this impact point.

This scan can easily be done by a function `sampleCB` which converts a *generalized case base*, including generalized cases, into a *point case base*, only including point cases. The function is called with an empty point case base:

```
function sampleCB(generalizedCaseBase, pointCaseBase){
  calculateScanPoints()
  while(isScanPointLeft())
    point = getNextScanPoint()
    foreach case in generalizedCaseBase
```



(a) scanning (b) points

Figure 3: Scanning of the two Dimensional Attribute Space

```
if(case.includes(point))
  pointCaseBase.add(createPointCaseOf(case,point))
```

This function has several disadvantages:

1. For each scan point all cases have to be checked. This can be very time consuming in large case bases.
2. If the scan points are selected unskillfully many generalized cases could be missed or too many point cases may be build.
3. There is no influence to the number of generated point cases.

A lot of possible improvements are imaginable and some of them will now be presented.

3.1.1 Individual scan for each generalized case

To address problems 2 and 3 the calculation of the scan point should not only depend on the domain of the attributes. A better result can be achieved by using additional information about the cases. Therefore, the body of function `sampleCB` is changed to an individual scan for each generalized case.

```
foreach case in generalizedCaseBase
  calculateScanPoints(case);
  while(isScanPointLeft())
    point = getNextScanPoint();
      if(case.includes(point))
        pointCaseBase.add(createPointCaseOf(case,point))
```

Of course, this method is more complex because the attribute space has to be checked for each generalized case. Most of the knowledge and intelligence is placed in the subfunction `calulateScanPoints(case)`, which reduces the number of possible scan points, e.g. only the attributes for which constraints exist must be scanned, the other attributes have always fixed values. The realization of this function has major impact on the conversion quality and can offer the case base administrator further influence to the process.

3.2 Calculate Scan Points

In this section several techniques are presented to improve the sampling process. The idea of most of the techniques is based on numerical attributes, but can probably be adapted to other kinds of attribute, e.g. taxonomies. To illustrate them, a complex two dimensional case is taken. The case is defined by two attributes p_1 and p_2 with a constraint set which is not defined any further.

3.2.1 Bounding Box

A big improvement would be if only the region around the generalized case would have to be scanned. Therefor, for each attribute the maximum and minimum values can be calculated which defines the bounding box around the case. The calculation of the scan points are then based on the bounding box and not on the domain of the attributes any more.

3.2.2 Number of Scan Points

The number of scan points for each attribute influences the number of point cases which are built. Three strategies to determine this number are imaginable:

Manual: The user specifies the number of scan points. This can be done by specifying the number for all cases by one global parameter or for each case individually, e.g. by a special case attribute.

Automatic: The converter calculates the number of scan points itself by using additional knowledge like the similarity measures. For example, the information about the global similarity can be used to specify the number of scan points. An attribute with a global similarity weight of zero can be ignored, one with a value of one should be scanned very exact.

Semi-Automatic: The user specifies the maximum and/or minimum number of resulting point cases for each generalized case. With this information the converter can estimate the number of scan points. However, if too many or to less point cases are created, the estimation has to be revised and the sampling must be performed again, until an appropriate number is created.

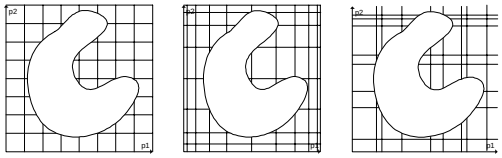
3.2.3 Size of the Scan Points' Intervals

If too few scan points are choosen a generalized case can be missed. If too many are choosen, too many point cases are build. This problem is not necessarily related to the previous one, only if a fixed interval size between the scan points is assumed. But several other techniques are possible which can be realized by the `createScanPoints` function:

linear: The attributes domains are sampeled in intervals of equal size.

logarithmic: The intervals are small in the region of zero and grow to the borders. By this function the center of a generalized case would be scanned very exact and the borders very poor. For the retrieval we expect, that the opposite could offer a better result. That means, the borders are checked very exact and the center only less. Nevertheless, this function only works with a bounding box for each case and is not useable for the whole attribute space.

random: Each scan point has a random position like typical for probabilistic methods. On average a good scan can be anticipated.



(a) linear (b) logarithmic (c) random

Figure 4: Three interval kinds

To conclude, independent from the scan technique several parameters to control the conversion can be identified, among them the maximum number of scan points to create, the maximum number of point cases in the point case base, and the minimum number of point cases for each generalized case.

3.3 Retrieval Modifications

After the conversion a case base only including point cases is available where well known retrieval methods can be applied. But one problem still exists: if the user defines a query and wants to retrieve the best five cases, he or she probably retrieves five point cases generated from the same generalized case (see figure 5).

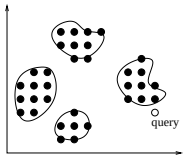


Figure 5: Retrieval Problem

To solve this problem two modifications have to be done:

1. Each point case needs an identifier which either defines the case as an original point case or defines to which generalized case it belongs.

2. If a case is added to the retrieval result is must be checked if another case, which also belongs to the same generalized case, is already in the result list. In this case, the retrieved case with the highest similarity must be placed in the result.

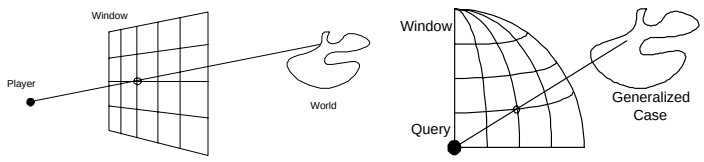
4 Computer Graphics based Retrieval Techniques

The idea is to use methods and algorithms from computer graphics for the retrieval of generealized cases. Interesting are methods from the area of 3D real time calculating that are applied in 3D games. The known techniques for space dividing, removing of hidden surfaces or others can probably be useable for the retrieval of generalized cases.

Realtime 3D applications demand very high requirements on the efficiency of the used techniques. The amount of items grows continuously and requires more powerful hard- and software. Even if not all of the used techniques can be adapted to CBR purposes, there could be some methods which possible can be used or are able to improve existing retrieval techniques.

4.1 Problems and Challenges

- Most of the 3D calculations work only in three dimensions, but in CBR, the attribute space is usually n-dimensional. It has to be examined if the algorithms are adaptable (see [Ban90]).
- The space in 3D games is an Euclidean space. This is usually not the case in CBR applications; here the similarity measures deform the space. But probably, a deformation matrix can be used.
- The player in a game looks always through a predefined window (see figure 6 a) which is screened. For each raster element a ray is sent from the player through the raster element into the virtual world. This view has to be enlarged like pictured in figure 6 b. Additionally, it has to be checked if it is necessary to modify the ray to a pyramid to catch all generalized cases.



(a) Games (b) CBR

Figure 6: Different kind of ray tracing

- Objects which are completely invisible because they are placed behind other objects

are removed in computer graphics. But this is not the case in CBR: an object behind another object has in general a smaller similarity value, but should also be retrieved.

4.2 Techniques from Computer Graphics for CBR Retrieval

This section shortly presents some well known techniques from Computer Graphics and assesses their abilities to improve the retrieval of generalized cases. A complete list and detailed description can be found at [3DE], [Ban90] and [Kel99].

4.2.1 World Conversion in Polygons

The world (in CBR the attribute space) is converted in an offline phase into polygons, e.g. each object is transformed into a set of smaller objects which represent its surface. With these smaller and well known objects the real time calculations are much easier. Additionally, the normal vector of each polygon represents the information whether the viewpoint is placed inside or outside the original object.

If this technique is also possible for n-dimensional spaces it provides a great performance improvement for the retrieval of generalized cases.

4.2.2 Data Reduction

Depending on the structure of the generalized cases the amount of resulting polygons could be very high. Therefor, it may be necessary to reduce this amount of data. In computer graphics this is done during the online phase with several techniques:

- Removing of invisible polygons with *visible surface determination* (VSD) and *hidden surface removal* (HSR). Of course, this techniques can not directly be used in CBR, because also a case which is placed behind another case could be one of the most similar cases. But the technique is possible for the polygons of the same generalized case.
- Reduction of number of polygons for objects which are far away from the view point with *level of detail* (LOD) techniques. Therefor, for each object several more or less complex sets of polygons are created and in relation to the view point a more exact or rougher one is taken. Probably, this technique can directly be adapted to CBR.

4.2.3 Data Structure

To improve the access to the data, an efficient data structure is necessary. For static scenes (a case base can be understood like that) exist two very interesting structures which are presented in the following:

Oct-trees and Quad-trees: A tree structure which is only taken for static scenes to improve ray tracing tests.

BSP-Tree: A *Binary Space Partition Tree* is a well balanced tree which is capable to handle also n-dimensional spaces. The creation of the tree is very complex, but the tree provides in the online phase a very good performance.

4.2.4 Further Algorithms

A lot of other algorithms from computer graphics may be helpful and their applicability in the domain of CBR has to be analyzed in future. The most promising approaches are:

Back-face Culling: Removing of polygons where the normal vector shows away from the view point. For example, independent from the view point only three sides of a cube are visible, thus the other ones can be ignored.

Z-Buffer: For each pixel of a polygon the distance to the view point is saved in the Z-buffer. A new pixel is only taken, if it has a smaller value, i.e., is nearer to the view point.

Hierarchical Z-Buffer: A very interesting extension which orders objects in a hierarchy and uses for each hierarchy a separate Z-buffer.

Warnock Algorithm: A removing algorithm with a conservative strategy of data reduction: polygons are only removed, if they are invisible without doubt.

5 Conclusion and Future Work

All the different ideas have their own pro and cons depending on the kind of attributes and constraints. Possibly, a combination of the different methods can be implemented to built the core of a general retriever for all kinds of attribute and computable constraint.

In future, the techniques have to be evaluated and tested to receive more information about their quality and strength. Especially the area of 3D computer graphics provides a big pool of efficient algorithms which have to be analyzed whether they are adaptable to n-dimensional spaces.

The first tests have shown, that the retrieval complexity for generalized cases is much higher than for point cases. Therefore, it is even more important to move as much effort as possible to the offline phase. This will be one of the great challenges we will have to face during our future research.

References

[3DE] 3D Engines Lis. http://cg.cs.tu-berlin.de/simki/engines.html.

[Ban90] Thomas F. Banchoff. *Beyond the Third Dimension*. Scientific American Library, New York NY, 1990.

[Bar89] Ray Bareiss. *Exemplar-Based Knowledge Acquisition: A unified Approach to Concept Representation, Classification and Learning*. Academic Press, 1989.

[Ber02] R. Bergmann. *Experience Management: Foundations, Development Methodology, and Internet-based Applications*. Springer, forthcomming, 2002.

[BV99] R. Bergmann and I. Vollrath. Generalized Cases: Representation and Steps Towards Efficient Similarity Assessment. In W. Burgard, Th. Christaller, and A. B. Cremers, editors, *KI-99: Advances in Artificial Intelligence.*, LNAI 1701. Springer, 1999.

[BVW99] R. Bergmann, I. Vollrath, and T. Wahlmann. Generalized Cases and their Application to Electronic Designs. In E. Melis, editor, *7. German Workshop on Case-Based Reasoning (GWCBR'99).*, 1999.

[Kel99] A. Keller. *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis*. Shaker Verlag, Aachen, 1999.

[Kol80] Janet L Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory*. PhD thesis, Yale University, 1980.

[Lew97] Jeff Lewis. Intellectual Property (IP) Components. Artisan Components, Inc., [web page], http://www.artisan.com/ip.html, 1997. [Accessed 28 Oct 1998].

[MB02] B. Mougouie and R. Bergmann. Similarity Assessment for Generalizied Cases by Optimization Methods. In *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-02)*. Springer., 2002.

[Sal91] S Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309, 1991.

[Sch96] Jörg W. Schaaf. Fish and Shrink: a next step towards efficient case retrieval in large scaled case bases. In Ian Smith and Boi Faltings, editors, *Advances in Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, 1186, pages 362–376. Springer Verlag, 1996.