MINLP Based Retrieval of Generalized Cases

Alexander Tartakovski[1], Martin Schaaf[2], Rainer Maximini[1], and Ralph Bergmann[1]

[1] University of Trier,
Department of Business Information Systems II
54286 Trier, Germany
{tartakov|rmaximini|bergmann}@wi2.uni-trier.de
[2] University of Hildesheim,
Institute for Mathematics and Applied Computer Science,
Data and Knowledge Management Group,
31113 Hildesheim, Germany
schaaf@dwm.uni-hildesheim.de

Abstract. The concept of generalized cases has been proven useful when searching for configurable and flexible products like reusable components in the area of electronic design automation. This paper addresses the similarity assessment and retrieval problem for case bases consisting of traditional and generalized cases. While approaches presented earlier were restricted to continuous domains, this paper addresses generalized cases defined over mixed, continuous and discrete, domains. It extends the view on the similarity assessment as a nonlinear optimization problem (NLP) towards a mixed integer nonlinear optimization problem (MINLP), which is an actual research topic in mathematical optimization. This is an important step because most real world applications require mixed domains for the case description. Furthermore, we introduce two optimization-based retrieval methods that operate on a previously created index structure, which restricts the retrieval response time significantly.

Keywords: *generalized cases, mixed integer nonlinear programming, kd-trees*

1 Introduction

The smallest experience item in Case-Based Reasoning (CBR) is called a *case*. When applying the structural CBR approach, each case is described by a finite and structured set of attribute-value pairs that characterize the problem and the solution. Hence, a single case can be considered as a point in the space defined by the Cartesian product of the problem space $\mathbb{P}$ and solution space $\mathbb{S}$. Newer applications of CBR motivated a new concept called *generalized cases* [1], [2]. In contrast to a traditional case, a generalized case does not cover only a point of the case space but a whole subspace. This allows the representation of complex and configurable products, for instance, reusable electronic design components, in a very natural and efficient way. Generalized cases provide a set of solutions to

a set of closely related problems and can be viewed as an implicit representation of a (possibly infinite) set of traditional point cases.
The concept of generalized cases implies the extension of similarity measures as well. In [2] the similarity between a query and a generalized case has been defined as the similarity between the query and the most similar point-case contained in the generalized case.
Previous work, e.g. [3], focuses on generalized cases defined over continuous attribute domains by considering the similarity assessment problem as a kind of Nonlinear Programm (NLP), which is well known in mathematical optimization. In contrast to these approaches, the work presented here addresses the similarity assessment and retrieval problem for generalized cases defined over mixed, discrete and continuous, attribute domains, which are typical for most real world applications.
The concepts presented here were developed in the context of the project *IPQ: IP Qualification for Efficient Design Reuse*[3] founded by the German Ministry of Education and Research (BMBF), which aims at the improvement of electronic design processes by reusing already existing design components called *Intellectual Properties* (IPs) [4]. Nowadays, IPs are offered by specialized vendors via the Internet. For such large IP assets, we developed a CBR-based retrieval solution [5]. Physically, IPs are descriptions that can be later synthesized to hardware. They are usually configurable to some degree, e.g. IPs with flexible bus width or frequency, and recommend themselves to be represented as generalized cases.
In section 2 we introduce an optimization-based approach to solve the similarity assessment problem. In section 3 we present two index-based approaches to improve the retrieval for generalized cases and point cases defined over mixed, continuous and discrete, domains.

2 Optimization Based Similarity Assessment

In this section we characterize a relationship between the similarity assessment problem for generalized cases and the optimization problem in mathematics. Furthermore, we describe how to solve the similarity assessment problem for generalized cases defined over continuous domains. Afterwards, we present a new approach to solve the similarity assessment problem for generalized cases defined over mixed domains.

2.1 Similarity Assessment as Optimization Problem

For the retrieval of generalized cases, the similarity between a problem and a generalized case must be determined. A natural way is to extend a traditional

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

similarity measure as follows [6]:

$$sim^*(q, GC) := max\{sim(q, c)|c \in GC\} \tag{1}$$

According to this definition, the value of the extended similarity function $sim^*(q, GC)$ is equal to the similarity $sim(q, c)$ between a query q and the most similar point case c contained in the generalized case GC.
Due to the fact that the similarity assessment problem can be viewed as a specific optimization problem, we describe the relationship between both problems as in [6]. An optimization problem is the maximization or minimization of some objective function, often under restrictions given through equalities and inequalities. In general, optimization problems are defined as follows:

$$\begin{aligned} &\max_x f(x) \\ &s.t. \quad x \in F \end{aligned} \tag{2}$$

with f *an objective function* and F a set of feasible solutions *(feasible set)*, implicit defined through constraints.
By defining an objective function $f(x) := sim(q, x)$ and the feasible set $F := GC$ we transform a similarity assessment problem to a specific optimization problem. In mathematical optimization several classes of optimization problems are known. They differ in computational complexity, problem solution methods and problem formulation. Therefore, it is important to find out the class and formulation of an optimization problem by deriving it from a similarity assessment problem. These classes and formulations differ for generalized cases defined over continuous domains and for generalized cases defined over mixed domains. This will be further elaborated in the following two sections.

2.2 Similarity Assessment for Continuous Domains

Generalized cases defined over continuous domains are restricted to connected sets in the case space spanned by continuous attributes. A single generalized case can be represented through equality and inequality constraints. The general form is:

$$\begin{aligned} GC = \{x \in \mathbb{R}^n | &c_1(x) \geq 0 \wedge \ldots \wedge c_k(x) \geq 0 \\ &\wedge c_{k+1}(x) = 0 \wedge \ldots \wedge c_l(x) = 0\} \end{aligned} \tag{3}$$

The constraint functions c_i are not restricted to be linear, they can also be nonlinear.
For refining the similarity assessment as an optimization problem, we regard a similarity function *sim*. Although the aggregation function is commonly a weighted average, which is a linear function, the local similarities are mostly nonlinear. Consequently, the global similarity function *sim* is nonlinear as well. Nonlinearity of the similarity function together with the nonlinearity of the generalized cases determine the class of optimization problems we are going to derive. It is a nonlinear optimization problem (NLP) [7] having a general form as follows:

$$\begin{aligned} \max_{x} \; & f(x) \\ s.t. \; & c_1(x) \geq 0, \\ & \cdots \\ & c_k(x) \geq 0, \\ & c_{k+1}(x) = 0, \\ & \cdots \\ & c_l(x) = 0, \\ & x \in \mathbb{R}^n \end{aligned} \tag{4}$$

This optimization problem has a nonlinear objective function and nonlinear constraints. By replacing the objective function f with $sim(q, x)$ (with constant q), we receive the desired representation of the similarity assessment problem as optimization problem. The constraint set can be taken directly from the specification of the generalized case itself.

2.3 Similarity Assessment for Mixed Domains

For mixed domains, the formulation of an optimization problem is much more complex. The most difficult issue is handling discrete attributes. In this section we explain mixed integer nonlinear optimization problem (MINLP) and present a formulation of similarity assessment as MINLP problem.

Example from the IPQ Project For illustrating the concepts presented in this paper, we will use an example from the design of electronic circuits (see above). The discrete cosine transformation IP (DCT IP) is a frequently reused design component because it implements an algorithm widely used for MPEG-2 encoders/decoders. The parameters of this IP are clock frequency, chip area, bus width, and subword size. There are dependencies between these parameters defining the feasible design space. For simplification and without loss of generality, we can restrict the description of DCT IPs to the attributes shown in the following table:

Table 1. Selected parameters of the example IP.

parameter		**description**
frequency	f	The clock frequency that can be applied to the IP. (continuous)
area	a	The chip area the synthesized IP will fit on. (continuous)
width	w	Number of bits per input/output word. Determines the accuracy of the DCT. Allowed values are 6, 7, ..., 16. (discrete)
subword	s	Number of bits calculated per clock tick. Changing this design space parameter may have a positive influence on one quality of the design while having a negative impact on another. Allowed values are 1, 2, 4, 8 and no-pipe. (discrete)

The dependencies between the parameters follow:

$$f \leq \begin{cases} -0.66w + 115 \; if \; s = 1 \\ -1.94w + 118 \; if \; s = 2 \\ -1.74w + 88 \quad if \; s = 4 \\ -0.96w + 54 \quad if \; s = 8 \\ -2.76w + 57 \quad if \; s = no - pipe \end{cases} \tag{5}$$

$$a \geq \begin{cases} 1081w^2 + 2885w + 10064 \; if \; s = 1 \\ 692w^2 + 2436w + 4367 \quad if \; s = 2 \\ 532w^2 + 1676w + 2794 \quad if \; s = 4 \\ 416w^2 + 1594w + 2413 \quad if \; s = 8 \\ 194w^2 + 2076w + 278 \quad if \; s = no - pipe \end{cases} \tag{6}$$

This IP can be viewed as a single generalized case with parameterized attributes f, a, w, and s.

Mixed Integer Nonlinear Optimization Problem The formulation of the assessment problem for generalized cases defined over mixed discrete and continuous domain is beyond the scope of NLP. The reason is a combinatorial character of the assessment problem which is not covered through an NLP. Therefore we need to use a generalization of NLP called mixed integer nonlinear program (MINLP) [8], which covers nonlinear and integer programming. A general formulation of this problem follows:

$$\begin{aligned} \min_{x,y} \; & f(x,y) \\ s.t. \; & c_1(x,y) \geq 0, \\ & \cdots \\ & c_k(x,y) \geq 0, \\ & c_{k+1}(x,y) = 0, \\ & \cdots \\ & c_l(x,y) = 0, \\ & x \in \mathbb{R}^m, y \in \mathbb{Z}^n \end{aligned} \tag{7}$$

The main difference to NLP is that for the objective function f and the constraints a continuous part x and an integer part y is distinguished.
MINLP is harder than NLP since it has, additionally, a combinatorial character. The handling of this problem is one of the actual research topics in mathematical optimization [9]. However, since few years there are several industrial solvers available handling MINLPs.

Similarity Assessment for Generalized Cases with mixed integer and continuous domains as MINLP Now we are going to explain a formulation

of MINLP for the example of the DCT-IP. In general, the formulation consists of two parts: from modelling of a feasible set and from modelling of the objective function. We start with the first one.
Since a feasible set of MINLP is defined through equalities and inequalities the following dependencies must be transformed:

$$\begin{array}{c} f \leq -0.66w + 115 \; if \; s = 1 \\ f \leq -1.94w + 118 \; if \; s = 2 \\ \vdots \end{array} \tag{8}$$

We define in place of the variable s with a domain $T(s) = \{1, 2, 4, 8, no - pipe\}$ $|T(s)|$ new variables:

$$s_1, s_2, s_4, s_8, s_{no-pipe} \in \mathbb{Z}$$

and a set of constraints:

$$s_1 \geq 0, s_1 \leq 1, s_2 \geq 0, s_2 \leq 1, \ldots \tag{9}$$

Each new variable represents a single attribute value of the variable s. So, if some new variable $s_v = 1$ it implies that $s = v$ and contrary if $s = v$ then $s_v = 1$. Since the variable s can have only one value at a given time, a new additional constraint should be defined:

$$s_1 + s_2 + s_4 + s_8 + s_{no-pipe} = 1 \tag{10}$$

Every valid assignment of variables $s_1 ... s_{no-pipe}$ implies a single value for the variable s and every assignment of variable s implies a valid assignment of variables $s_1 ... s_{no-pipe}$. Example:

$$s = 4 \Leftrightarrow \begin{pmatrix} s_1 \\ s_2 \\ s_4 \\ s_8 \\ s_{no-pipe} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{11}$$

Now, it is simple to formulate the dependencies (8) as inequalities:

$$\begin{array}{c} s_1(-0.66w + 115 - f) \geq 0 \\ s_2(-1.94w + 118 - f) \geq 0 \\ \vdots \end{array} \tag{12}$$

For $s_1 = 1$ the first inequality is "switched on" and the other inequalities are "switched off", since $s_2 = s_4 = s_8 = 0$.
The feasible set of MINLP is given through the set of constraints (12), (9), (10) and additional constraints depending on the attribute a (chip area):

$$\begin{array}{ll}
s_1(-0.66w + 115 - f) & \geq 0 \\
s_2(-1.94w + 118 - f) & \geq 0 \\
\vdots & \\
s_1(-1081w^2 - 2885w - 10064 + a) & \geq 0 \\
s_2(-692w^2 - 2436w - 4367 + a) & \geq 0 \\
\vdots & \\
s_1 & \geq 0 \\
-s_1 & \geq -1 \\
s_2 & \geq 0 \\
-s_2 & \geq -1 \\
\vdots & \\
s_1 + s_2 + s_4 + s_8 + s_{no-pipe} & = 1 \\
w & \geq 6 \\
w & \leq 16
\end{array} \tag{13}$$

$$f, a \in \mathbb{R} \text{ and } w, s_1, s_2, s_4, s_8, s_{no-pipe} \in \mathbb{Z}$$

We proceed with the objective function f. Because of the introduction of new binary variables the formulation of the function f becomes more complex. To define the objective function we first define the similarity function given by local similarities and an aggregation function Φ:

$$sim(q, c) := \Phi(sim_f(q_f, c_f), sim_w(q_w, c_w), sim_a(q_a, c_a), sim_s(q_s, c_s))$$

Consequently, the objective function can be defined as follows:

$$\begin{aligned}
f_q(f, w, a, s_1, \ldots, s_{no-pipe}) := \Phi\Big(& sim_f(q_f, f), sim_w(q_w, w), \\
& sim_a(q_a, a), \big(s_1 sim_s(q_s, 1) + s_2 sim_s(q_s, 2) + s_4 sim_s(q_s, 4) \\
& + s_8 sim_s(q_s, 8) + s_{no-pipe} sim_s(q_s, no - pipe)\big)\Big)
\end{aligned} \tag{14}$$

The idea, here, is based on the following fact:

$$\begin{aligned}
sim_s(q_s, c_s) = \big(& s_1 sim_s(q_s, 1) + s_2 sim_s(q_s, 2) + s_4 sim_s(q_s, 4) \\
& + s_8 sim_s(q_s, 8) + s_{no-pipe} sim_s(q_s, no - pipe)\big)
\end{aligned}$$

Revisiting the example (11) again and assuming that $s_1 = 0, s_2 = 0, s_4 = 1, s_8 = 0$ and $s_{no-pipe} = 0$ is part of some valid assignment according to the constraint set (13):

$$\big(s_1 sim_s(q_s,1) + s_2 sim_s(q_s,2) + s_4 sim_s(q_s,4)$$
$$+ s_8 sim_s(q_s,8) + s_{no-pipe} sim_s(q_s, no-pipe)\big) =$$
$$\big(0 sim_s(q_s,1) + 0 sim_s(q_s,2) + 1 sim_s(q_s,4)$$
$$+ 0 sim_s(q_s,8) + 0 sim_s(q_s, no-pipe)\big) = sim_s(q_s,4)$$

The objective function (14) together with the feasible set (13) define the MINLP, its solution provides the similarity between the query q and the example generalized case.
Normally, MINLP problems are not solved exactly but approximately.
There are several commercial solver on the market for MINLP problems, e.g. GAMS/Baron [9], Xpress-SLP, and MINLP.

3 Retrieval

Because of the high calculation complexity of the assessment problem for generalized cases it is very important to develop index-based retrieval approaches. The overall strategy is to build an index structure in advance, which is later used for improving the response time of the retrieval. The step when building the index structure will be denoted as *offline phase*, the retrieval step as *online phase*. Hence, most of the calculation complexity is shifted from the online to the offline phase. For building and integrating index structures, we developed two new methods, a similarity based method presented in section 3.1 and a kd-tree based method, which will be introduced in section 3.2.

3.1 Similarity Based Retrieval Method

Because of the high complexity of the assessment problem for generalized cases we developed a retrieval method that is based on a fix similarity measure for building an index structure.
A main step of this approach, is a partition of a problem space $\mathbb{P}$ into some simple subspaces. An example of a such simple subspace is a hyperrectangle that has faces parallel to the coordinate planes. Queries are points of exactly one of the subspaces, but it is unknown which one and where exactly.
Furthermore, we define for a subspace Sub and a generalized case GC two similarity bounds:

$$Similarity_{min}(Sub, GC) := \min_{s \in Sub} sim^*(s, GC) = \min_{s \in Sub} \max_{g \in GC} sim(s, g) \tag{15}$$

and

$$Similarity_{max}(Sub, GC) := \max_{s \in Sub} sim^*(s, GC) = \max_{s \in Sub} \max_{g \in GC} sim(s, g) \tag{16}$$

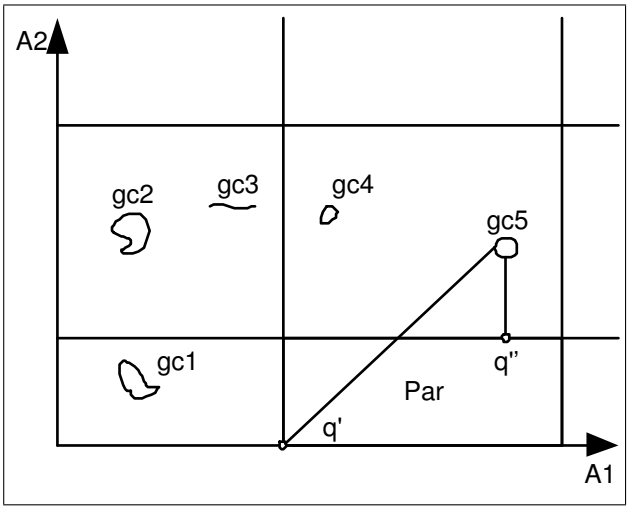

Fig. 1. Similarity bounds

Consider the subspace Sub and the generalize case $gc5$ in figure 1. The query q' belonging to the subspace Sub has a lowest similarity to generalized case $gc5$ from all queries of this subspace. Exactly this similarity value is provided by the function $Similarity_{min}(Sub, gc5)$. The query q'' belonging to the subspace Sub has a highest similarity to generalized case $gc5$ from all queries of this subspace. This similarity value is provided by the function $Similarity_{max}(Sub, gc57)$. If these bounds are known and some query is inside the subspace Sub in the online phase we can guarantee that its similarity to the generalized case $gc5$ lies within the bounds $Similarity_{min}(Sub, gc5)$ and $Similarity_{max}(Sub, gc5)$.
Based on this fact, a retrieval approach can be simply constructed. The first idea is to calculate, in the offline phase, similarity bounds for all subspaces and all generalized cases. Furthermore, it is necessary to derive a partial order on generalized cases in terms of similarity for every single subspace and all generalized cases. The partial order is defined as follows:

$$\bigwedge_{\substack{gc_1, gc_2 \in CB, \\ Sub \in \mathbb{P}}} gc_1 < gc_2 \Leftrightarrow Similarity_{max}(Sub, gc_1) < Similarity_{min}(Sub, gc_2)$$

Figure 2 illustrates the calculated similarity bounds for a single subspace and the generalized cases $gc1$ to $gc5$. Furthermore, Figure 2 shows the resulting partial order. For every query of the online phase, we only need to check to which subspace it belongs to. After this check we know immediately the partial order and the similarity bounds for all generalized cases. When searching for the n best nearest neighbours, we can exclude all cases having n or more successors because it is guaranteed that at least n cases are more similar. Finally we have to perform linear retrieval on the rest of the cases.

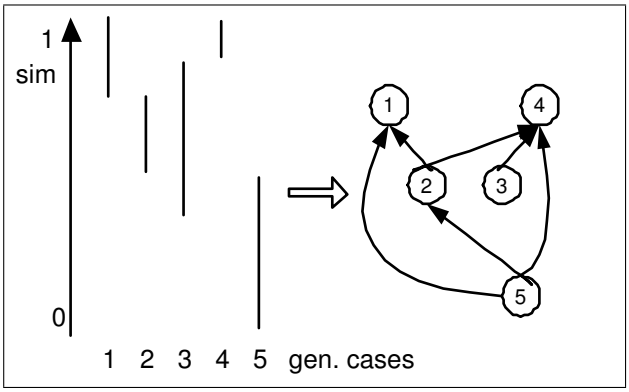

Fig. 2. Similarity intervals and partial order

This method can be significantly improved if we bound the size of the retrieval set before building the index structure. The customer is normally interested in at most $10-20$ cases in a retrieval set. Consequently, for a single subspace we have only to remember all generalized cases with fewer predecessors than the desired maximum size of the retrieval set. This improvement reduces significantly the size of information, which should be saved with every single subspace. Therefore, we can produce a much more detailed partition of the problem space and reduce the online complexity. The reason for this significant improvement is the fact that the cardinality-bound of a retrieval set is usually much lower than the size of the case base.
A further improvement can be achieved by using the technique of decision trees. The problem space can be partitioned recursively by choosing attributes and attribute values and building new subspaces with borders on the chosen values. For every gained subspace similarity bounds to all generalized cases and a partial order can be calculated. Based on this data the termination criterion can be specified (e.g. size of non dropped cases, degree of deviation of partial order from linear order and so forth). The main algorithm schema for building an index structure is as follows:

INPUT: Case Base CB, similarity measure sim,
maximum size of a retrieval set

OUTPUT: A Retrieval Tree

1. create a root node R,
   assign a whole problem space P to it,
   assign all general cases CB to it.
2. select a leaf L with assigned subspace Sub
   and assigned subset $SubCB \subseteq CB$ of cases,
   STOP if termination criterion is valid.

3. select some attribute A of the subspace Sub.
4. determine a cutting point p on selected attribute A
   Condition: there is some point $t \in Sub$
   having attribute value $t_A = p$.
5. create two constraints $A \leq p$ and $A > p$.
6. create two new leafs L_1 and L_2,
   assign $Sub_1 = \{t \in Sub | t_A \leq p\}$ to L_1
   and $Sub_2 = \{t \in Sub | t_A > p\}$ to L_2.
7. calculate $\forall GC \in SubCB$:
   $Similarity_{min}(Sub_1, GC)$,
   $Similarity_{max}(Sub_1, GC)$,
   $Similarity_{min}(Sub_2, GC)$,
   $Similarity_{max}(Sub_2, GC)$.
8. based on the similarity bounds calculate the partial order
   O_1 and O_2 on generalized cases with respect to
   Sub_1 and Sub_2.
9. for Sub_1: drop all cases having equal or more
   successors as the maximum size of a retrieval set,
   the rest of generalized cases becomes $SubCB_1$,
   for Sub_2: drop all cases having equal or more
   successors as the maximum size of a retrieval set,
   the rest of generalized cases becomes $SubCB_2$.
10. assign $SubCB_1$, O_1 to L_1 and $SubCB_2$, O_2 to L_2, delete the assignment
    of $SubCB$ to L.
11. set a node L as a predecessor of L_1 and L_2.
    (L is no longer a leaf, now).
12. GOTO 2

The result of this algorithm is a tree with leaves having assigned significant cases and partial orders on them.
For every query in the online phase the subspace where the query belongs to, can be effectively determined. We have to start with a root node and then follow the path of subspaces including the query. The rest is an execution of linear retrieval of cases assigned to the leaf determined.

Computation of MAX/MAX and MIN/MAX-Problems In the description of this approach we didn't discuss a computation of max/max and min/max problems. Although solving max/max problems is quite simple, the min/max problems are complex. We start with the simple case first.
Consider the definition of upper similarity bound (16), the max/max problem was given as follows:

$$\max_{s \in Sub} \max_{g \in GC} sim(s, g) \tag{17}$$

A subspace Sub and a generalized case GC are both located in the problem space P, i.e. $Sub \subseteq P$ and $GC \subseteq P$. Regard a space $P \times P$. Furthermore, imagine that

the subspace Sub is located in the first space of a Cartesian product and the generalized case GC in the second space of a Cartesian product. The following optimization problem in general form is then equivalent to the max/max problem (17):

$$\begin{aligned} &\max_{x} \; sim((x_1,\ldots,x_n),(x_{n+1},\ldots,x_{2n})) \\ &s.t. \quad (x_1,\ldots,x_n) \in Sub, \\ &\qquad (x_{n+1},\ldots,x_{2n}) \in GC, \\ &\qquad x \in \mathbb{P}^2 \end{aligned} \tag{18}$$

The consequence is that the max/max problem can be formulated as a common max problem in double dimensioned space.
Since the max/max problem can be formulated as NLP or MINLP, we mention again that this kind of problems can be solved for the majority of cases only approximately. In our approach we have chosen an upper approximation.
The treatment of the min/max problem is much more complex. We were not able to find publications tackling this problem in general. Although there is some work on handling special min/max problems (e.g. in [10]) these approaches are not applicable here.
Our idea is not to solve this problem exactly but approximately by estimating a lower bound of the objective function. By estimating upper bound for max/max problem and lower bound for min/max problem the index structure stays consistent, i.e. no cases are excluded that belong to exact retrieval set. Figure 3

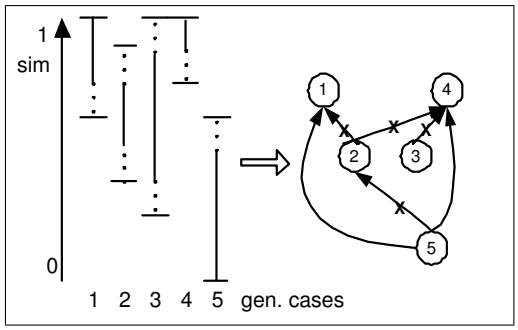

Fig. 3. Relaxation of Similarity Bounds

shows the relaxed similarity bounds. The figure shows that some intervals that didn't overlap before are overlapping now. Consequently, the ordering between the corresponding generalized cases is not valid anymore. Furthermore, this relaxation doesn't lead to new ordering relationships, so no cases can be excluded, that wouldn't be excluded by exact calculation of bounds.
The simplest way to estimate a lower bound for min/max problem is to solve min/min problem, since:

$$\forall Sub, GC \subseteq P : \\ \min_{s \in Sub} \min_{g \in GC} sim(s,g) < \min_{s \in Sub} \max_{g \in GC} sim(s,g) \tag{19}$$

The handling of the min/min problem is exactly the same as the handling of the max/max problem. It can be formulated as a common min problem in double dimensioned space.
The other possibility to estimate a lower bound for the min/max problem by a known feasible point c in the generalized case is to calculate:

$$\min_{s \in Sub} sim(s,c) \tag{20}$$

Also, here it holds:

$$\forall Sub, GC \subseteq P, c \in GC : \\ \min_{s \in Sub} sim(s,c) < \min_{s \in Sub} \max_{g \in GC} sim(s,g) \tag{21}$$

In both cases the min problem has to be approximated through the lower bound.

3.2 Kd-Tree Based Retrieval Method

Another method to reduce the computational complexity of the retrieval adopts the idea of the kd-trees [11]. The key characteristic of this method is building the index structure independent from the similarity measure.
The traditional kd-tree based retrieval consists of two major steps - building a kd-tree in the offline phase and using this tree for searching during the online phase. For the retrieval of generalized cases these parts will be adapted and extended.

Building a kd-tree The following algorithm [11] builds the kd-tree variant, named inreca tree, for common case bases.
While there are no changes in the main flow of the algorithm necessary to create kd-trees for case bases consisting of generalized cases, two methods have to be extended. The first extension is made in the function *Partition*. This function contains a test checking if a given case belongs to a given subspace. This check is quite simple for point cases, but not for generalized cases.
A generalized case belongs to some subspace if and only if their intersection is not empty. E.g. for generalized case GC represented through $GC = \bigcup_{i=1}^{n} g_i$ with g_i closed connected sets it should be checked if there is some $i \in [1,n]$ with $g_i \cap Subspace \neq \emptyset$.
The feasibility problem, that is, finding a point in the intersection of finitely many sets, is discussed in various areas of sciences. It is well researched and for many cases efficient to solve. Therefore, the test in the function *Partition* should be extended to solve a feasibility problem for the given generalized case and the given subspace.

INPUT: Case Base CB

Output: An Inreca Tree

```
 1. IF NOT Split?(CB) THEN RETURN MakeBucket(CB)
 2. ELSE
 3.         Discriminator := SelectAttribute(CB)
 4.         IF OrderedValueRange(Discriminator) THEN
 5.            Value := SelectValue(CB, Discriminator)
 6.            RETURN MakeInternalOrderedNode(Discriminator, Value,
                   CreateTree (Partition_<(Discriminator, Value, CB)),
                   CreateTree (Partition_>(Discriminator, Value, CB)),
                   CreateTree (Partition_=(Discriminator, Value, CB)),
                   CreateTree (Partition_unknown(Discriminator, Value, CB)))
 7.         ELSE
 8.            RETURN MakeInternalUnorderedNode(Discriminator,
                   CreateTree (Partition_1(Discriminator, CB)),...,
                   CreateTree (Partition_m(Discriminator, CB)),
                   CreateTree (Partition_unknown(Discriminator, CB)))
 9.         ENDIF
10. ENDIF
```

The function *split* in the traditional algorithm interrupts the split process if a current subspace includes fewer cases as a given limit. Since several generalized cases can overlap, it's not always possible to achieve the limit, therefore the split process should be stopped if, after several attempts, no successful split occurs.

Searching Similar Generalized Cases Using a k-d Tree There are no major changes necessary for the online phase. Since a case base contains generalized and point cases, sim^* should be calculated instead of sim in this part of the method. The search algorithm, BOB and BWB tests remain the same.

4 Related Work and Summary

The idea of generalized cases is not new and has been already discussed in the area of instance-based learning and in earlier works on CBR [12]. Although not explicitly mentioned, some CBR-based applications adopt the idea of generalizing cases and provide proprietary solutions, often restricted to the particular application domain. For an overview, see [13]. The term generalized case has been introduced in [1], which provides a formal and systematic view using constraints to express the dependencies between several attributes. Based in this, we developed the first index based method for generalized cases defined over mixed, continuous and discrete, domains [14]. The idea of this approach is to transform generalized cases by sampling them into point cases and using fast traditional retrieval engines.

In this paper we presented the formulation of the similarity assessment problem for generalized cases with mixed domains as a special optimization problem MINLP. Because of the computational complexity, we introduced two optimization-based retrieval methods that operate on a previously created index structure. The first retrieval method takes the similarity measure into account, while the second one is based on kd-trees. Both optimization techniques improve the response time of CBR-based applications with generalized cases significantly. However, generating the required index-structures can be time consuming but this is done only once for static case bases.

References

1. Bergmann, R., Vollrath, I., Wahlmann, T.: Generalized cases and their application to electronic design. In E. Melis (Hrsg.) 7th German Workshop on Case-Based Reasoning (1999)
2. Bergmann, R.: Experience management. Springer-Verlag Berlin Heidelberg New York (2002)
3. Mougouie, B., Bergmann, R.: Similarity assessment for generalized cases by optimization methods. In S.Craw and A.Preece (Hrsg.) European Conference on Case-Based Reasoning (ECCBR'02).Lecture Notes in Artificial Intelligence, Springer (2002)
4. Lewis, J.: Intellectual property (ip) components. Artisan Components, Inc., web page, http://www.artisan.com (1997, Accessed 28 Oct 1998)
5. Schaaf, M., Maximini, R., Bergmann, R., Tautz, C., Traphöner, R.: Supporting electronic design reuse by integrating quality-criteria into cbr-based ip selection. Proceedings 6th European Conference on Case Based Reasoning (September 2002)
6. Bergmann, R., Vollrath, I.: Generalized cases: Representation and steps towards efficient similarity assessment. KI-99 (1999)
7. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear programming, theory and algorithms. Wiley (1993)
8. Leyffer, S.: Deterministic methods for mixed integer nonlinear programming. PhD Thesis, Department of Mathematics and Computer Science, University of Dundee (1993)
9. Tawarmalani, M., Sahinidis, N.: Convexification and global optimization in continuous and mixed-integer nonlinear programming: Theory, algorithms, software, and applications. Kluwer Academic Publishers, Boston MA (2002)
10. Horst, R., Tuy, H.: Global optimization: Deterministic approaches. 2nd rev. Edition, Springer, Berlin, Germany (1993)
11. Wess, S., Althoff, K.D., Derwand, G.: Using k-d trees to improve the retrieval step in case-based reasoning. University of Kaiserslautern (1993)
12. Kolodner, J.L.: Retrieval and organizational strategies in conceptual memory. PhD thesis, Yale University (1980)
13. Maximini, K., Maximini, R., Bergmann, R.: An investigation of generalized cases: Theory, algorithms, software, and applications. Proceedings of 5th International Conference on Case Base Reasoning (ICCBR'03), June 2003 in Trondheim, Norway. Editors: Kevin D. Ashley, Derek G. Briddge (2003)
14. Maximini, R., Tartakovski, A., Bergmann, R.: Investigating different methods for efficient retrieval of generalized cases. In Reimer U., Abecker A., Staab S., Stumme

G.(Hrsg). WM2003: Professionelles Wissensmanagement-Erfahrungen und Visionen (2003)