Similarity Assessment and Retrieval of Generalized Cases

Alexander Tartakovski and Rainer Maximini
University of Hildesheim
Institute for Mathematics and Applied Computer Science
Data and Knowledge Management Group
31113, Hildesheim, Germany
{tartakov|r_maximi}@dwm.uni-hildesheim.de

Abstract

This paper addresses the similarity assessment and the retrieval problems in Case-Based Reasoning for case bases consisting of traditional and generalized cases. Previous work focussed on similarity assessment for generalized cases with continuous domains. The similarity assessment problem was formulated as Nonlinear Programm (NLP), which is well known in mathematical optimization. In several real world applications (e.g. in our project) generalized cases have mixed discrete and continuous domains. This paper provides a formulation of similarity assessment problem for such generalized cases as a Mixed Integer Nonlinear Optimization Programm (MINLP). Furthermore this paper proposes two different index-based methods solving the retrieval problem.

1 Introduction

The smallest experience item in Case-Based Reasoning is called *case*. It is a point in case space, i.e. in the Cartesian product of problem space $\mathbb{P}$ and solution $\mathbb{S}$ space. A case assigns a single solution to a single characterization of some problem.
Several applications, e.g. selection of parameterized products within electronic commerce, lead to an extension of this concept. A case that covers not only a point of the case space but a subspace of it is called *generalized case* [Bergmann *et al.*, 1999], [Bergmann, 2002]. A single generalized case provides a set of solutions to a set of closely related problems. It can be viewed as an implicit representation of a (possibly infinite) set of traditional point cases, formally: $GC \subseteq \mathbb{P} \times \mathbb{S}$.
The extension of the case concept implies the extension of a similarity measure. A reasonable possibility is to define the similarity between a query and a generalized case as the similarity between the query and the most similar point-case that the generalized case includes, formally: $sim^*(q, GC) := max\{sim(q, c)|c \in GC\}$.
This problem was regarded in previous work with a focus on generalized cases with continuous domains [Bergmann and Vollrath, 1999], [Mougouie and Bergmann, 1999]. In several real world applications (e.g. in our project) the domains are mixed, i.e., some attributes of a case space are continuous and some of them are discrete. In chapter 2 we introduce an optimization-based approach to solve this problem.
Because of a high computational complexity of the similarity assessment problem it is not satisfiable to use a linear retrieval method. It is important to investigate an index-based retrieval in order not to solve all hard assessment problems if a query arises. In chapter 3 we present two index-based approaches to retrieve from case bases consisting of generalized cases and point cases with mixed integer 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 treat the similarity assessment problem for generalized cases with continuous domains. The main idea of this approach is to formulate the assessment problem as *nonlinear optimization problem (NLP)*, which can be solved with common optimization software. Afterwards we present a new approach to treat the similarity assessment problem for generalized cases with mixed discrete and continuous domains. Here we formulate the assessment problem as the specific optimization problem called *mixed integer nonlinear optimization problem(MINLP)* as well. In contrast to NLP formulation MINLP formulation is much more complex. We discuss this approach on a concrete example from our project.

2.1 Similarity Assessment as Optimization Problem

For retrieval of generalized cases, the similarity between a problem and a generalized case must be determined. A natural way is to extend a traditional similarity measure as follows [Bergmann and Vollrath, 1999]:

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

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 containing in generalized case GC.
Due to the fact that the similarity assessment problem can be viewed as a specific optimization problem, we regard first the last one. An optimization problem in mathematics is maximizing or minimizing of some objective function, often under restriction given through equalities and inequalities. A general form can be defined as follows:

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & x \in F \end{array} \qquad (2)$$

with f *an objective function* and F a set of feasible solutions *(feasible set)*, implicit defined through constraints.

There is a direct relation between the similarity assessment problem and the optimization problem [Bergmann and Vollrath, 1999]. By defining an objective function $f(x) := sim(q, x)$ and $F := GC$ cp. (1) we transform a similarity assessment problem to a specific optimization problem.
In mathematics several classes of optimization problems are known. They differs in computational complexity, methods of treating and in a problem formulation. On account of this it is important to find out the class and formulation of optimization problem by deriving it from similarity assessment problem. These classes and formulations differ for generalized cases with continuous domains and for generalized cases with mixed discrete and continuous domains. This is a content of the following two subsections.

2.2 Similarity Assessment for Continuous Domains

In this subsection we regard optimization based similarity assessment for generalized cases with continuous domain. This kind of cases is restricted to be connected sets in case space spawned by continuous attributes. A single generalized case can be represented through equality and inequality constraints. The general form is:

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

The constraint functions c_i are not restricted to be linear, they can also be nonlinear.
Now we are going to formulate an optimization problem. For this 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. A direct consequence is nonlinearity of the global similarity function sim. This property and the property of generalized cases determine the class of optimization problem we are going to derive. It is a nonlinear optimization problem (NLP) [Bazaraa, M.S. *et al.*, 1993] having a general form as follows:

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & c_i(x) \geq 0, \quad i \in I, \\ & c_i(x) = 0, \quad i \in E, \\ & x \in \mathbb{R}^n \end{array} \quad (4)$$

This optimization problem has a nonlinear objective function and nonlinear constraints.
The formulation of the optimization problem is quite similar to the formulation in the general case. The objective function f is equal to $sim(q, x)$ (with constant query q) and the constraints set can be directly taken over.

2.3 Similarity Assessment for Mixed Domains

By mixed domains the formulation of an optimization problem is much more complex. The most difficult issue is handling of discrete attributes. In this subsection we describe an example generalized case from the IPQ project with such properties, we explain mixed integer nonlinear optimization problem (MINLP) and present a formulation of similarity assessment for example generalized case as MINLP problem.

Example from the IPQ Project
The growing complexity of today's electronic design forces to reuse existing design objects, called IPs *(Intellectual Properties)* [Lewis, 1997]. IP is a description of flexible design that have to be synthesized to hardware. It usually spans a design space. The behavior of the final hardware depends on parameters of the design description. Experience management approaches can be used to support the selecting process of existing IPs. It is one of the goals of the current Project *IPQ: IP Qualification for Efficient Design Reuse*[1] founded by the German Ministry of Education and Research (BMBF).
For electronic design IP discrete cosine transformation (DCT) is a typical design object. It implements an algorithm for (DCT) and its inverse operation which is needed as an important part of decoders and encoders of the widely known MPEG-2 video compression algorithm. The variable parameters of the IP are shown in a following table:

Table 1. Selected parameters of the example IP.

parameter		**description**
frequency	f	The clock frequency that can be applied to the IP.
area	a	The chip area the synthesized IP will fit on.
width	w	Number of bits per input/output word. Determines the accuracy of the DCT. Allowed values are 6, 7, ..., 16.
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.

Now we present dependencies between the parameters of an example IP:

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

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

This IP can be viewed as a single generalized case with parameterized attributes f, a, w, s and with not listed fixed attribute values. The dependencies (5) between the parameterized attributes are represented according to format for generalized cases introduced in [Maximini, 2002].

[1] 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

Mixed Integer Nonlinear Optimization Problem

The formulation of the assessment problem for generalized cases with mixed discrete and continuous domain as NLP can't succeed. 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) [Leyffer, 1993], which covers nonlinear and integer programming. A general formulation of this problem follows:

$$\begin{array}{lll} \min\limits_{x,y} & f(x,y) & \\ s.t. & c_i(x,y) \geq 0, & i \in I, \\ & c_i(x,y) = 0, & i \in E, \\ & x \in \mathbb{R}^n, y \in \mathbb{Z}^n & \end{array} \tag{6}$$

The main difference to NLP is that the objective function f has an argument consisting of an continuous part x and an integer part y.

This problem is harder as NLP since it has in addition a combinatorial character. The handling of this problem is one of the actual research interests in mathematical optimization. However since few years there are several industrial solvers handling MINLP.

Similarity Assessment for Generalized Cases with mixed integer and continuous domains as MINLP

Now we are going to explain a formulation of MINLP on example of the DCT-IP. As in general case 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 should be transferred:

$$\begin{array}{rrrr} f \leq & -0.66w + 115 & if & s = 1 \\ f \leq & -1.94w + 118 & if & s = 2 \\ & \vdots & & \end{array} \tag{7}$$

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:

$$\begin{array}{rcl} s_1 & \geq & 0 \\ s_1 & \leq & 1 \\ s_2 & \geq & 0 \\ s_2 & \leq & 1 \\ \vdots & & \end{array} \tag{8}$$

Each new variable stays for 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 to the same time, a new additional constraint should be defined:

$$s_1 + s_2 + s_4 + s_8 + s_{no-pipe} = 1 \tag{9}$$

Now 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:

$$s = 4 \Leftrightarrow \begin{pmatrix} s_1 \\ s_2 \\ s_4 \\ s_8 \\ s_{no-pipe} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

Now it is simply possible to formulate the dependencies (7) as inequalities:

$$\begin{array}{rcl} s_1(-0.66w + 115 - f) & \geq & 0 \\ s_2(-1.94w + 118 - f) & \geq & 0 \\ & \vdots & \end{array} \tag{11}$$

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 (11), (8), (9) and additional through not listed before constraints with variable a inside:

$$\begin{array}{rcl} s_1(-0.66w + 115 - f) & \geq & 0 \\ s_2(-1.94w + 118 - f) & \geq & 0 \\ \vdots & & \\ s_1(-1081w^2 - 2885w - 10064 + a) & \geq & 0 \\ s_2(-692w^2 - 2436w - 4367 + a) & \geq & 0 \\ \vdots & & \\ s_1 & \geq & 0 \\ -s_1 & \geq & -1 \\ s_2 & \geq & 0 \\ -s_2 & \geq & -1 \\ \vdots & & \\ s_1 + s_2 + s_4 + s_8 & = & 1 \end{array} \tag{12}$$

$f, w, a \in \mathbb{R}^n$ and $s_1, s_2, s_4, s_8 \in \mathbb{Z}^n$

Now we proceed with an 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 regard the similarity function given through local similarities and an aggregation function Φ:

$$sim(q,c) := \Phi(sim_f(q_f, c_f), sim_w(q_w, c_w),$$
$$sim_a(q_a, c_a), sim_s(q_s, c_s),$$
$$sim_{r_1}(q_{r_1}, c_{r_1}), \ldots, sim_{r_n}(q_{r_n}, c_{r_n}))$$

The symbols $r_1, \ldots, r_n$ refer to not listed before and not parameterized attributes.

The objective function can be formulated in the following way:

$$f_q(f, w, a, s_1, \ldots, s_{no-pipe}, r_1, \ldots, r_n) := \quad (13)$$
$$\Phi\Big(sim_f(q_f, f), sim_w(q_w, w), sim_a(q_a, a),$$
$$\big(s_1 sim_s(q_s, 1) + s_2 sim_s(q_s, 2) + s_4 sim_s(q_s, 4)$$
$$+ s_8 sim_s(q_s, 8) + s_{no-pipe} sim_s(q_s, no-pipe)\big),$$
$$sim_{r_1}(q_{r_1}, r_1), \ldots sim_{r_n}(q_{r_n}, r_n)\Big)$$

The trick by this formulation is based on the following fact:

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

Regard the example (10) again. Assume that $s_1 = 0, s_2 = 0, s_4 = 1, s_8 = 0$ and $s_{no-pipe} = 0$ is part of some valid assignment accordantly to the constraint set (12). Then the term:

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

is equal to:

$$\big(0 sim_s(q_s, 1) + 0 sim_s(q_s, 2) + 1 sim_s(q_s, 4)$$
$$+ 0 sim_s(q_s, 8) + 0 sim_s(q_s, no-pipe)\big) =$$
$$sim_s(q_s, 4)$$

The objective function (13) together with the feasible set (12) define a wanted MINLP, which solution provides the similarity between the query q and the example generalized case.

It is to mention that MINPL is a very hard problem, e.g. it is harder than NLP and integer program IP. Normally this problem is not solved exactly, but approximately.

There are several commercial solver on the market treating this problem. Among of them Xpress-SLP, MINLP, GAMS and so on.

3 Retrieval

Because of the high calculation complexity of the assessment problem for generalized cases it is very important to investigate an index-based retrieval. We developed two new methods which reduce the response time, through building and using of index-structures. In the first subsection we introduce a similarity based method, which takes the similarity function into account to build a high efficient retrieval tree. In the second subsection we introduce a kd-Tree based method, which builds the index structure independent from the similarity measure.

3.1 Similarity Based Retrieval Method

Because of the high complexity of the assessment problem for generalized cases we decided to develop a retrieval method that takes into account a fix similarity measure for building of 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 partition is a hyperrectangle that has faces parallel to the coordinate planes. Figure 1 shows a problem space consisting of two continuous attributes that is partitioned into hyperrectangles.

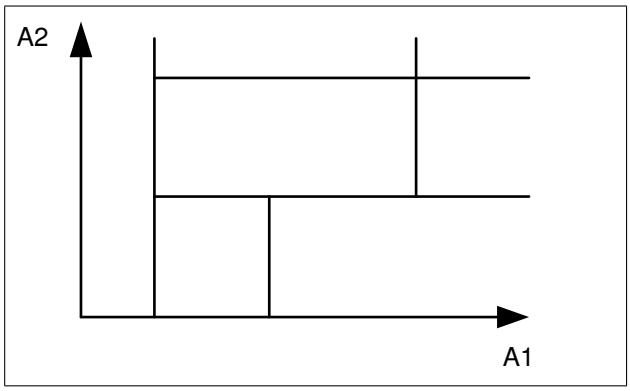

Figure 1: Partition with hyperrectangles

Every query arising in the online phase hits exactly one of this subspaces, but it is unknown which subspace and where in the subspace.

Furthermore, we define for a partition Par and a generalized case GC two similarity bounds:

$$Similarity_{min}(Par, GC) := \quad (14)$$
$$\min_{s \in Par} sim^*(s, GC) = \min_{s \in Par} \max_{g \in GC} sim(s, g)$$

and

$$Similarity_{max}(Par, GC) := \quad (15)$$
$$\max_{s \in Par} sim^*(s, GC) = \max_{s \in Par} \max_{g \in GC} sim(s, g)$$

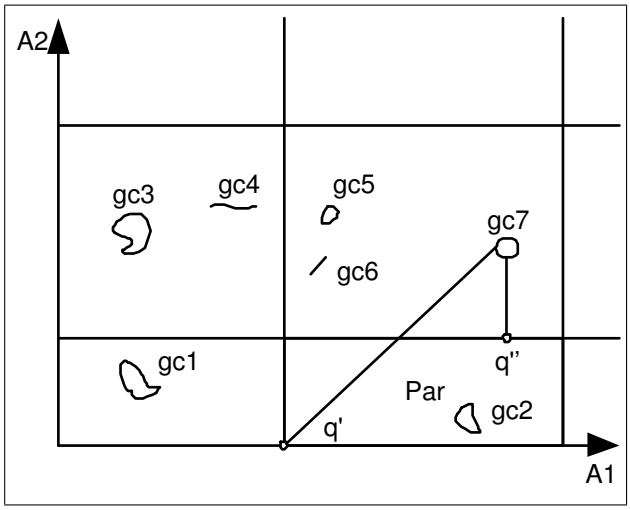

Figure 2: Similarity bounds

Consider the partition Par and the generalize case $gc7$ in figure 2. The query q' hitting the partition Par has a lowest similarity to generalized case $gc7$ from all queries hitting this partition. Exactly this similarity value is provided by the function $Similarity_{min}(Par, gc7)$. The query q'' hitting the partition Par has a highest similarity to generalized case $gc7$ from all queries hitting this partition. Exactly this similarity value is provided by the function $Similarity_{max}(Par, gc7)$. If these bounds are known and some query hits the partition Par in the online phase we can guarantee that its similarity to the generalized case $gc7$ lies between the bounds $Similarity_{min}(Par, gc7)$ and $Similarity_{max}(Par, gc7)$.

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 partitions and all generalized cases. Furthermore, it is necessary to derive a partial order on generalized cases in terms of similarity for every single partition and all generalized cases. The calculation of the partial order is based on calculated bounds. Figure

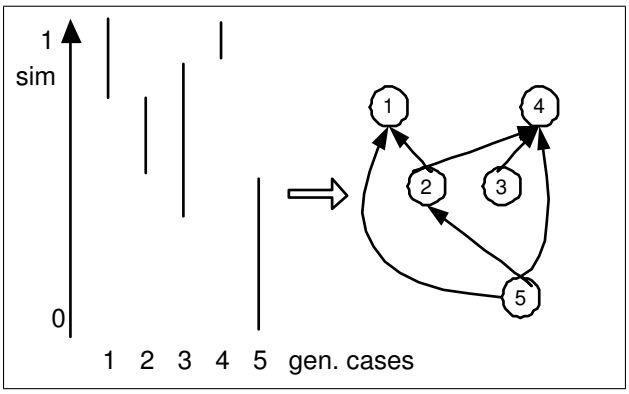

Figure 3: Similarity intervals and partial order

3 demonstrates uncertainty intervals given by similarity bounds for some single partition and generalized cases $gc1 - gc5$. Figure 3 demonstrates also a partial order on generalized cases $gc1 - gc5$ induced through similarity intervals. If a query arrises in the online phase, we only need to check which partition it hitts. Then we can get 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 predecessors in partial order. Finally we have to perform linear retrieval on the rest of cases.
This method can be significant improved if we bound the size of the retrieval set before building the index structure. The customer is normally interested on at most $10 - 20$ cases in a retrieval set. Consequently for a single partition we have only to remember all generalized cases having a lower number of 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 partition. 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 lower than the size of the case base.

A further improvement can be achieved by using the

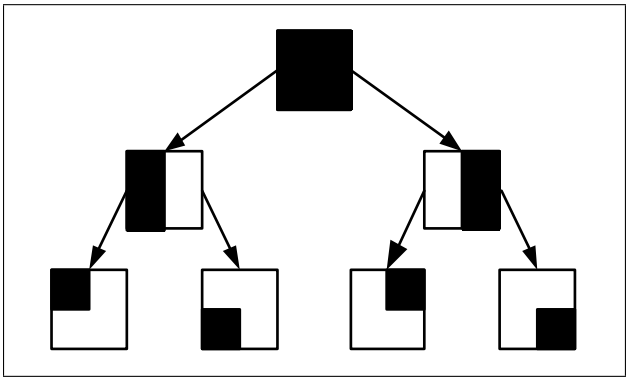

Figure 4: Partition with a decision tree

technic of decision trees. The problem space can be partitioned recursively by choosing attributes and attribute values and building new partitions with a border on the chosen value. For every gained partition (cf. figure 4.) similarity bounds to all generalized cases and a partial order can be calculated. Based on this data the termination criterion can be developed (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 of index structure is as follows:

INPUT: Case Base CB, similarity measure sim

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 partition Par
   and assigned subset $SubCB \subseteq CB$ of cases,
   STOP if termination criterion is valid.
3. select some attribute A of the regarded partition Par.
4. select a cutting point p on selected attribute A of the selected partition SD.
   Condition: there is some point $t \in Par$
   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 $Par_1 = \{t \in Par | t_A \leq p\}$ to L_1
   and $Par_2 = \{t \in Par | t_A > p\}$ to L_2.
7. calculate $\forall GC \in SubCB$:
   $Similarity_{min}(Par_1, GC)$,
   $Similarity_{max}(Par_1, GC)$,
   $Similarity_{min}(Par_2, GC)$,
   $Similarity_{max}(Par_2, GC)$.
8. based on the similarity bounds calculate the partial order
   O_1 and O_2 on generalized cases with respect to Par_1 and Par_2.
9. for Par_1: drop all cases having equal or more predecessors as the maximum size of a retrieval set, call the rest of generalized cases as $SubCB_1$,
   for Par_2: drop all cases having equal or more predecessors as the maximum size of a retrieval set, call the rest of generalized cases as $SubCB_2$.
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.
12. GOTO 2

The result of this algorithm is a tree with leaves having assigned significant cases and partial orders on them.
If a query arises in the online phase the partition which the query hits can be effective acquired. We have to start with

a root node and then follow the path of partitions including the query. The rest is an execution of linear retrieval of cases assigned to an arrived leaf.

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 the treatment of max/max problem is quite simple the treatment of min/max problem is complex. We start with the simple case first.
Remind of definition of upper similarity bound (15), the max/max problem was given as follows:

$$\max_{s \in Par} \max_{g \in GC} sim(s, g) \tag{16}$$

A partition Par and a generalized case GC are both located in the problem space P, i.e. $Par \subseteq P$ and $GC \subseteq P$. Regard a space $P \times P$. Furthermore imagine that the partition Par 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 (16):

$$\begin{array}{ll} \max\limits_{x} & sim((x_1, \ldots, x_n), (x_{n+1}, \ldots, x_{2n})) \\ s.t. & (x_1, \ldots, x_n) \in Par, \\ & (x_{n+1}, \ldots, x_{2n}) \in GC, \\ & x \in \mathbb{P}^2 \end{array} \tag{17}$$

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 in majority of cases only approximately. In our approach we choose an upper approximation. The reason of this commitment is explained afterwards.
The treatment of the min/max problem is much more complex. There is no literature found by authors handling this problem in general. Although there is some work on handling special min/max problems (e.g. in [Horst and Tuy, 1993]) this is not matching our case.
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.

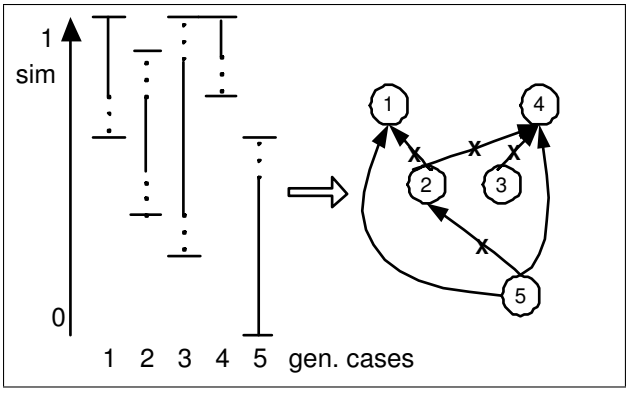

Figure 5: Relaxation of uncertainty intervals

Figure 5 shows relaxed intervals of the uncertainty. We can see that some intervals that didn't overlap before are overlapping now. This causes that the ordering between the corresponding generalized cases is not valid more. 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:

$$\begin{array}{l} \forall Par, GC \subseteq P : \\ \min\limits_{s \in Par} \min\limits_{g \in GC} sim(s, g) < \min\limits_{s \in Par} \max\limits_{g \in GC} sim(s, g) \end{array} \tag{18}$$

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:

$$\min_{s \in Par} sim(s, c) \tag{19}$$

Also here it holds:

$$\begin{array}{l} \forall Par, GC \subseteq P, c \in GC : \\ \min\limits_{s \in Par} sim(s, c) < \min\limits_{s \in Par} \max\limits_{g \in GC} sim(s, g) \end{array} \tag{20}$$

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

3.2 Kd-Tree Based Retrieval Method

In this subsection we describe, explain and discuss a novel k-d tree-based retrieval techniques for case bases consisting of traditional and generalized cases. The key characteristic of this method is building the index structure independent from the similarity measure.
The traditional k-d tree-based retrieval consists of two major parts - building a k-d tree in the offline phase and searching similar generalized cases using k-d tree in the online phase. For the retrieval of generalized cases these parts will be adapted and extended.

Building a k-d Tree
The algorithm in Table 1 [Wess *et al.*, 1993] builds the k-d tree variant, named inreca tree, for common case bases.
The main algorithm for case bases consisting of common and generalized cases stays the same. The difference occurs in the function *Partition* in decision if a case belongs to some partition. This check is quite simple for point cases, but not for generalized cases.
A generalized case belongs to some partition if and only if their intersection is not empty. For generalized case GC represented through $GC = \bigcup_{i=1}^{n} g_i$ with g_i closed connected sets (generalized cases with mixed integer and continuous domain) should be checked if there is some $i \in [1, n]$ with $g_i \cap Partition \neq \emptyset$.
The feasibility problem, that is, finding a point in the intersection of finitely many closed convex sets in Euclidian spaces, arises in various areas of mathematics and physical sciences. It is well discovered and efficient to solve.
If some set g_i is not convex it can be relaxed to a convex set. The function *Partition* can assign by this relaxation some generalized case to a partition without really having an intersection point. This inexactness doesn't lead to failure by retrieval.

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

Table 1: Construction of the Inreca tree

Searching Similar Generalized Cases Using a k-d Tree
There are no great changes in 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 stay the same.

4 Related Work

The idea of generalized cases was already implicitly presented since the very beginning of CBR and instance-based learning research [Kolodner, 1980]. A more formal and systematic view on generalized cases using constraints to express the dependencies between several attributes was presented in [Bergmann *et al.*, 1999].
The first online retrieval approach for generalized cases with continuous domain was presented in [Bergmann and Vollrath, 1999]. The idea of this algorithm is to perform a linear retrieval without solving an assessment problem exactly. Upper and lower bounds are calculated for a query and all cases from a case base. Based on similarity bounds similar cases to the query can be found, but not the most similar cases in general.
In [Mougouie and Bergmann, 1999] an extended online retrieval approach for generalized cases with continuous domain is presented. This approach is based on techniques of mathematical optimization. Similar to the first approach the idea is to estimate the bounds. In contrast these bounds can be refined in order to gain the exact set of the best neighbours.
We introduced in [Maximini *et al.*, 2003] the first index based method for generalized cases with mixed continuous and discrete domain. The idea of this approach is to transform generalized cases through sampling into point cases to use well researched and fast retrieval engines.

5 Summary and Outlook

The both facts that on the one hand the similarity assessment problem for generalized cases can be formulated as an optimization problem and on the other hand mathematical optimization problems belong to a research field since many years in mathematics lead to an investigation optimization-based assessment and retrieval methods.
In this paper we introduced for the first time the formulation of a similarity assessment problem for generalized cases with mixed continuous and discrete domains as a special optimization problem MINLP. Furthermore, we introduced two optimization-based retrieval methods building the index structure in the offline phase. The first retrieval method takes into account the similarity measure by building an index structure. We hope that this method will work efficient in the online phase. The second one is more flexible, since it is independent from similarity measure. We expect a lower efficiency. The next research point is a realization and evaluation of these concepts.

References

[Bazaraa, M.S. *et al.*, 1993] Bazaraa M. S., Sherali H.D., and Shetty C.M. *NonLinear Programming, Theory and Algorithms*. Second Edition. Wiley, 1993.

[Bergmann, 2002] Ralph Bergmann. *Experience Management*. Springer-Verlag Berlin Heidelberg New York, 2002.

[Bergmann *et al.*, 1999] Bergmann, R., Vollrath, I., and Wahlmann, T. *Generalized cases and their application to electronic design.* In E. Melis (Hrsg.) 7th German Workshop on Case-Based Reasoning, 1999

[Bergmann and Vollrath, 1999] Bergmann, R., and Vollrath, I. *Generalized cases: Representation and Steps Towards Efficient Similarity Assessment.* KI-99.

[Horst and Tuy, 1993] Horst R. and Tuy H. *Global Optimization: Deterministic Approaches.* 2nd rev. Edition, Springer, Berlin, Germany, 1993.

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

[Lewis, 1997] Lewis, J. *Intellectual property (IP) components.* Artisan Components, Inc., [web page], http://www.artisan.com(1997)[Accessed 28 Oct 1998].

[Leyffer, 1993] Leyffer S. *Deterministic Methods for Mixed Integer Nonlinear Programming.* PhD Thesis, Department of Mathematics and Computer Science, University of Dundee, 1993

[Maximini, 2002] Maximini R. *Concepts for the Representation of Parametrized IP by Constraints in the Sense of Artificial Intelligence.* technical report, University of Hildesheim, 2002

[Maximini *et al.*, 2003] Maximini R., Tartakovski A., and 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.

[Mougouie and Bergmann, 1999] Mougouie, B. and 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

[Wess *et al.*, 1993] Wess S., Althoff K. D., and Derwand G. *Using k-d Trees to Improve the Retrieval Step in Case-Based Reasoning* University of Kaiserslautern, 1993