Similarity Assessment for Generalizied Cases by Optimization Methods

Babak Mougouie[1] and Ralph Bergmann[2]

[1] Max-Planck Institut für Informatik, Saarbrücken, Germany
mbabak@mpi-sb.mpg.de
[2] University of Hildesheim, Data- and Knowledge Management Group
PO-Box 101363, D-31113 Hildesheim, Germany
bergmann@dwm.uni-hildesheim.de

Abstract. Generalized cases are cases that cover a subspace rather than a point in the problem-solution space. Generalized cases can be represented by a set of constraints over the case attributes. 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 find the distance (or the related similarity) between the point query and the closest point of the area covered by the generalized cases, with respect to some given similarity measure. We formulate this problem as a mathematical optimization problem and we propose a new cutting plane method which enables us to rank generalized cases according to their distance to the query.

1 Introduction

The traditional concept of a cases is that of a point in the problem-solution space, i. e., a problem-solution pair that assigns a single solution to a single problem. During case-based problem solving, cases are retrieved from a case base using a similarity function, which compares the case description with the current query. Driven by examination of several new applications, we proposed the concept of *generalized cases* [7, 6, 5]. A generalized case covers not only a point of the problem-solution space but a whole subspace of it. A single generalized case immediately provides solutions to a set of closely related problems rather than to a single problem only. The solutions that 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 viewed as an implicit representation of a (possibly infinite) set of traditional "point cases".

1.1 Motivation for Generalized Cases

Generalized cases naturally occur in certain CBR applications, such as for the recommendation of parameterized products within electronic commerce or brokerage services. The selection of reusable Intellectual Properties (IPs) of electronic designs is a very concrete example application. Such an IP is a design

object whose major value comes from the skill of its producer [14], and a redesign of it would consume significant time. 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. More details of this application area for generalized cases is given in section 2.

We also want to make clear, that the idea of generalizing cases is not a radically new concept. It was already implicitly present since the very beginning of CBR and instance-based learning research [13, 2, 18]. For the purpose of adaptation, recent CBR systems in the area of design and planning [12, 17, 8] use complex case representations that realize generalized cases. However, in this paper we explore general and more formal view on the concept of generalized cases, which also partially covers the above mentioned related work.

In general, we expect several significant advantages of representing experience in the form of generalized cases [7], which can be summarized as follows:

- experience that naturally covers some space rather than a point can be represented more adequately,
- generalized cases can be integrated with traditional point-cases in a single case base,
- several (possibly an infinite number of) point-cases can be substituted by a single generalized case which helps to reduce the size of the case base,
- case-specific adaptation knowledge can be represented, rather that general case-independent adaptation knowledge.

1.2 The Research Problem: Similarity Assessment and Retrieval

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

In this paper we address the problem of similarity computation for generalized cases defined over an n-dimensional Real valued vector space through sets of constraints. This extends our previous work that was restricted to linear constraints [6] In particular, we propose two methods for similarity assessment based on numeric optimization methods.[3] We distinguish between convex and non-convex representations of the constraints that define the generalized cases. We first propose the Topkis-Veinott method, which is a feasible direction method, to solve similarity computation problem for convex constraints. However, if the generalized cases includes also non-concvex constraints, this method can still be used on a relaxation of the similarity problem to compute an upper

[3] We must assume some basic familiarization of the reader with optimizations methods to allow a complete understanding of the methods proposed here. For a comprehensive textbook on optimization see [3].

bound on the similarity. Second, we provide a cutting plane algorithm to compute *approximations* for the similarity of a generalized case. Instead of finding the exact similarity value, the algorithm allows to incrementally find increasing lower and decreasing upper bounds for the similarity of generalized cases. This algorithm works for non-convex and convex constraints. However, its performance decreases for non-convex constraints due to the inherent complexity of the problem of solving a feasibility problem. This algorithm is the foundation for developing retrieval algorithms for ranking generalized cases w.r.t. their similarity to the query similar to the fish and shrink algorithm proposed in [19].

2 Representing Electronic Design IPs as Generalized Cases

Increasingly, electronics companies use IPs ("Intellectual Properties") from third parties inside their complex electronic systems. An IP is a design object whose major value comes from the skill of its producer [14], 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, first, the public-domain documentation of IPs is very restricted and second 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 new application. This is one objective of the current project *IPQ: IP Qualification for Efficient Design Reuse* [4] funded by the German Ministry of Education and Research (BMBF) and the related European Medea project *ToolIP: Tools and Methods for IP* [5].

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.

We developed a representative IP to study issues of representation, retrieval, and parameterization. This IP implements an algorithm for the *discrete cosine transform (DCT)* and its inverse operation *(IDCT)* which is needed as an important part of decoders and encoders of the widely known MPEG-2 video compression algorithm. While the complete electronics design in the VHDL language is described in [21], we now focus only on the variable parameters of the IP, which are shown in Table 1.

[4] 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

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

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.

These parameters heavily depend on each other: increasing the accuracy w of the DCT/IDCT increases the chip area consumption a and decreases the maximum clock frequency f in a particular way, shown in Fig. 1. These relationships define a very specific design space that is spanned by the four mentioned parameters. The knowledge about this design space is very central for the selection of the IP during the process of design reuse. When case-based reasoning is applied for IP reuse, the question must be solved how to represent and how to reason with cases that encode design spaces rather than design points.

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

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

Fig. 1. Dependencies between the parameters of an example IP

3 Representing, Similarity Assessment, and Retrieval

To lay the foundation for the remainder of the paper, we now briefly review the basic formal concept of generalized cases and the related similarity assessment described in detail in [6].

Let $\mathcal{S}$ be the (possibly infinite) representation space for cases. In the IP application this representation space would include the parameters f, a, w, and s. One could assume that this representation space is subdivided into a problem

space and a solution space, but we drop this assumption here since it is less appropriate in design domains. A traditional case c or *point case* is a point in the representation space $\mathcal{S}$, i. e., $c \in \mathcal{S}$. A *generalized case*, can now be extensionally defined as follows:

Definition 1. (Generalized Case) A *generalized case* gc is a subset of the representation space, i. e., $gc \subseteq \mathcal{S}$.

Hence, a generalized case stands for a set of point cases. However, a generalized case should not represent an arbitrary set. The idea is that a generalized case is an abbreviation for a set of closely related point cases that naturally occur as one entity in the real world (e. g. a single IP).

For retrieving generalized cases, the similarity between a query and a generalized case must be determined. As in traditional CBR, we assume that the query is a point in the representation space that may be only partially described. We further assume that a traditional similarity measure $\mathrm{sim}(q, c)$ is given which assesses the similarity between a query q and a point case c. Such a similarity measure can be extended in a canonical way to assess the similarity $\mathrm{sim}^*(q, gc)$ between a query q and a generalized case gc:

Definition 2. (Canonical Extension of Similarity Measures for Generalized Cases) The similarity measure $\mathrm{sim}^*(q, gc) := sup\{\mathrm{sim}(q, c) \mid c \in gc\}$ is called the canonical extension of the similarity measure sim to generalized cases.

Applying sim^* ensures that those generalized cases are retrieved that contain the point cases which are most similar to the query. This property is captured in the following lemma.

Lemma 1. Given a case base CB of point cases and a similarity measure sim, then for any case base CB^* of generalized cases such that $CB = \bigcup_{gc \in CB^*} gc$ holds: if c_{ret} is a case from CB which is most similar to a query q w. r. t. sim then there is a generalized case gc_{ret} from CB^* which is most similar to q w. r. t. sim^* such that $c_{ret} \in gc$.

This states that if we introduce generalized cases together with sim^*, the same cases are retrieved during problem solving, independent from the clustering of point cases into generalized cases. This provides a clear semantics of generalized cases defined in terms of traditional point cases and similarity measures.

For building CBR systems that reason with generalized cases, efficient representations for generalized cases and efficient approaches for similarity assessment must be developed. Obviously, a straight forward approach by applying the definitions from the previous section (e. g. computing the similarity for each point case covered by a generalized case and determining the maximum of the similarity values) can be quite inefficient if the generalized case covers a large subspace and is impossible of it covers an infinite space. In the following we assume a representation for generalized cases that uses constraints for representing dependencies between attributes of the representation space. Hence, a generalized case is represented by a set of constraints $\{C_1, \ldots, C_k\}$. Such a generalized case represents the set of point cases whose attribute values fulfill all constraints it consists of.

4 Rephrasing Similarity Assessment

In the remainder of the paper we focus on the problem of similarity assessment and retrieval w.r.t. the canonical extension of a given similarity measure. This problem is quite hard if generalized cases are represented by sets of constraints such as shown in Fig. 1. In the following we restrict ourselves to case representation spaces that are n-dimensional Real valued vector spaces, i.e. $\mathcal{S} = I\!R^n$ and $gc \subseteq I\!R^n$. Further, we assume that the generalized case is intensionally defined by a set of constraints over $I\!R^n$, i.e. a point case is within the generalized cases if for this point all constraints are fulfilled. According to our current analysis of the domain of IP reuse, we consider this as a reasonable restriction for IP representation as well as a reasonable restriction for other domains in which generalized cases occur.

4.1 Similarity Assessment as Optimization Problem

We now rephrase the similarity assessment problem for generalized cases as an optimization problem. For this purpose we make use of the well known correspondence between similarity measures and distance measures as discussed for example in [10]. Given a bijective, strictly monotonously decreasing mapping $f : [0, 1] \to [0, 1]$ such that $f(0) = 1$, $f(1) = 0$ and $\mathrm{sim}(x, y) = f(dist(x, y))$ then *dist* is a distance measure. Then we say sim and *dist* are compatible. Given such a compatible distance measure, the similarity assessment problem for generalized cases can now be formulated as an optimization problem:

$$\textbf{(OP)} \qquad \begin{array}{l} \textbf{min}\ dist(q, x) \\ \text{s.t.} h(x) \leq 0, \end{array}$$

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

4.2 Convex Minimization

Certain important classes of optimization problems have the property that every local minimum is a global minimum. A well known example is *convex minimization* where the objective function (here the distance measure) is a convex function and where the feasible set (here the constraints of the generalized case) is a convex set. To be more precise, let's consider some basic definitions.

Definition 3. A set $\mathcal{C} \subset I\!R^n$ is called *convex* if for each two points x, y in $\mathcal{C}$ the line segment $[x, y]$ connecting x and y is completely inside $\mathcal{C}$.

Definition 4. Let $\mathcal{C} \subset I\!R^n, \mathcal{C} \neq \emptyset, \mathcal{C}$ convex. A function $f : \mathcal{C} \to I\!R$ is called *convex* if for all $x, y \in \mathcal{C}$ and all $\alpha \in (0, 1)$

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$$

Geometrically, the line segment connecting $[x, y]$ is above the graph of f.

If both, the objective function *dist* is a convex function and *gc* is a convex set, then using nonlinear programming methods, we can retrieve a global optimal solution. For this important cases we can exactly determine the similarity sim^* as we will show in section 5.

4.3 Relaxed Optimization to Find a Lower Bound

We now relax the restriction to convex constraints. If *gc* includes constraints which are also non-convex, we can at least relax the original optimization problem **(OP)** Therefore, we can decompose **(OP)** as follows:

$$\textbf{(OP)} \qquad \begin{aligned} \textbf{min} \quad & dist(q, x) \\ \text{s.t.} \quad & h_1(x) \leq 0, \\ & h_2(x) \leq 0, \end{aligned}$$

where h_1 is a set of m_1 convex functions and h_2 is a set of m_2 non-convex functions and $m_1 + m_2 = m$. As a result, the following problem is again a convex minimization problem:

$$\textbf{(CP)} \qquad \begin{aligned} \textbf{min} \quad & dist(q, x) \\ \text{s.t.} \quad & h_1(x) \leq 0. \end{aligned}$$

It's clear that (**CP**) is a relaxation of (**OP**), therefore its optimal solution is a lower bound for (**OP**).

5 The Topkis-Veinott Method

In order to compute the optimal solution to the convex minimization problem, one can apply the well known Topkis-Veinott algorithm [3]. This nonlinear optimization method is basically a feasible direction method that finds the global optimum under the assumption that the objective function *dist* is convex and all constraints h_1 are convex. We have analyzed several similarity and distance measure that typically occur in CBR applications (particularly including the IP reuse domain) and it turned out that all of them are in fact convex. Hence, this algorithm is of some practical importance.

The idea of the Topkis-Veinott method is to improve the value of objective function by moving from one feasible point (a point which fulfills all constraints) to another one. Let's consider the following definition.

Definition 5. Consider the problem $\mathbf{min} f(x)$ subject to $x \in S$, where $f : \mathbb{R}^n \to \mathbb{R}$ and S is a nonempty set in $\mathbb{R}^n$. A nonzero vector d is called a *feasible direction* at $x \in S$ if there exist a $\delta > 0$ such that $x + \lambda d \in S$ for all $\lambda \in (0, \delta)$. Furthermore, d is called an *improving fesible direction* at $x \in S$ if there exists a $\delta > 0$ such that $f(x + \lambda d) < f(x)$ and $x + \lambda d \in S$ for all $\lambda \in (0, \delta)$.

Suppose that x is a given feasible point. If an improving feasible direction d at x exists, we move along d with a step size λ reaching a new feasible point and we do the same iteration for the new point. If such an improving direction does not exist, we conclude that the current point is the optimal solution. Applying this method on (**CP**) we receive the global optimal solution. We now formalize the algorithm as follows:

For the sake of simplicity set $h_1 = h$.

Input: $h = h_i(x) \leq 0$ for $i = 1, .., m_1$,
$f(x) = dist(q, x)$,
an initial feasible solution x_1 with $h_i(x_1) \leq 0$.
Output: the optimal solution of (**CP**).

Set $k = 1$ and go to step 1.

1. If (z_k, d_k) be an optimal solution of:

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & \nabla f(x_k)^t d - z \leq 0 \\ & \nabla h_i(x_k)^t d - z \leq -h_i(x_k) \text{ for } i = 1, ..., m_1 \\ & -1 \leq d_j \leq 1 \qquad\quad \text{for } j = 1, ..., n \end{aligned}$$

   If $z_k = 0$, stop; x_k is the optimal solution of (**CP**). If $z_k < 0$, go to step 2.
2. Let λ_k be an optimal solution to:

$$\begin{aligned} \min \quad & f(x_k + \lambda d_k) \\ \text{s.t.} \quad & 0 \leq \lambda \leq \lambda_{max} \end{aligned}$$

   where $\lambda_{max} = sup\{\lambda : h_i(x_k + \lambda d_k) \leq 0 \text{ for } i = 1, ..., m_1\}$. Let $x_{k+1} = x_k + \lambda_k d_k$, replace k by $k + 1$ and return to step 1.

Remark 1. As we said, the optimal solution of (**CP**) is a lower bound for (**OP**). To make this bound tighter, we can replace the objective function of (**CP**) with its *Lagrangian relaxation* $dist(q, x) + uh_2(x)$ where $u = (u_1, ..., u_{m_2}) \geq 0$ is the *Lagrangian multiplier* associated with $h_2(x) \leq 0$.

Remark 2. In case of non-differentiability of any of the above functions, we can apply the sub-gradient of the function instead of its gradient. It's obvious that Topkis-Veinott algorithm is not the only choice we have. We can apply any other algorithm like sub-gradient algorithm as well.

6 A Cutting Plane Method

Retrieving the most similar case to the query is our ultimate goal. This can be achieved if we are able to rank all gc's such that there is one gc where its upper bound is smaller than the lower bound of all others. Hence it is not necessary to determine the similarity between the query and all generalized cases exactly. This principle is also underlying the fish and shrink retrieval algorithm [19].

6.1 The Basic Approach

We assume that for each gc, we are given a feasible point, i.e. one point for which we know that all constraints are fulfilled. This feasible point gives us an upper bound for the corresponding gc and the Topkis-Veinott Algorithm on the relaxed problem will give a lower bound for it. This means we can rank gc's using these lower and upper bounds and probably exclude some of them which have lower bounds greater than the upper bound of at least one gc. The problem arises when there is no gc such that its upper bound is smaller than the lower bound of all other gc's. To solve this problem we need to tighten the bounds. Therefore, we propose a cutting plane method which does the task:

Input: (**OP**) corresponding to a gc,
an upper bound ub and a lower bound lb for (**OP**).
Output: a better lower or upper bound for (**OP**).

1. Consider the line segment connecting ub and lb. Retrieve the perpendicular bisector of this line segment and its corresponding half plane including lb. Let's represent this half plane by $H(x) \leq 0$.
2. We want to find a feasible point for the region $\mathcal{R}$ constructed by the following constraints:

$$\begin{aligned} &h(x) \leq 0, \\ &H(x) \leq 0, \\ &dist(q,x) - dist(q,ub) \leq 0. \end{aligned}$$

   Let's call these $m+2$ constraints $g_i(x) \leq 0$ for $i = 1, ..., m+2$.
3. Suppose $g_i(ub) \leq 0$ for $i \in I$ and $g_i(ub) > 0$ for $i \notin I$. Now consider the following problem which is a feasibility problem:

$$\begin{aligned} & & \textbf{min} \ \textstyle\sum_{i \notin I} z_i & & \\ \textbf{(FP)} \qquad & \text{s.t.} & g_i(x) & \leq 0 & \text{for } i \in I \\ & & g_i(x) & -z_i \leq 0 & \text{for } i \notin I \\ & & & z_i \geq 0 & \text{for } i \notin I \end{aligned}$$

   If the optimal objective value of the above problem is zero, then there exists a feasible solution of $\mathcal{R}$, since $z_i = 0$ for $i \notin I$, therefore $g_i(x) \leq 0$ for $i = 1, ..., m+2$. We solve the above problem starting from $(ub, \hat{z})$, where $\hat{z}_i = g_i(ub)$ for $i \notin I$.

4. Assume at termination of **(FP)** the optimal objective value is zero. Hence we obtain a feasible solution of $\mathcal{R}$. This means we have recognized a point valid in gc which is closer to q than ub. Therefore, we have a new upper bound for **(OP)**.
5. If **(FP)** has the optimal objective value bigger than zero, we conclude there is no better solution than ub in $H(x) \leq 0$. Therefore, we have to look for the better solution in $H(x) > 0$. We add this constraint to the set of constraints of **(CP)** and solve the new problem. The new region does not include lb since lb is inside $H(x) \leq 0$ due to the construction of it. So the solution of the new **(CP)** is a new lower bound for **(OP)**.

With the above method, we tighten the bounds of **(OP)**. Therefore applying it iteratively will not be exhaustive. The only problem is that in each iteration we need to solve two optimization problems which is time consuming. To avoid this, we consider some more intelligent methods.

6.2 Gauges, Unit Balls and Distances

Definition 6. Let $\mathcal{B}$ be a compact set in $\mathbb{R}^n$ containing the origin in its interior. The *Minkowski-functional* or *gauge* with respect to $\mathcal{B}$ is defined as

$$\gamma_{\mathcal{B}}(x) = inf\{\lambda \geq 0 : x \in \lambda\mathcal{B}\}, \qquad x \in \mathbb{R}^n,$$

and $\mathcal{B}$ is called the *unit ball* of the gauge.

The idea to find the distance from x to y is to put the origin of the unit ball on x and enlarge or shrink the unit ball till its boundary touches y,

$$dist(x, y) = \gamma_{\mathcal{B}}(x - y)$$

As an example for Euclidean distance, the unit ball is a circle.

Remark 3. The unit balls should be subset of $\mathbb{R}^n$ but not subset of $\mathbb{R}^{n-1}$ (otherwise we have infinite or undefined distances in some dimensions). Then, we simply say they are *affine unit balls*.

Definition 7. Let the affine polyhedron $\mathcal{B}$ define a gauge in $\mathbb{R}^n$. Then ext($\mathcal{B}$) $= \{e_1, ..., e_\sigma\}$ is the set of *extreme points* of $\mathcal{B}$.
Each face of $\mathcal{B}$ is an affine polyhedron in $\mathbb{R}^{n-1}$ with some extreme points in ext($\mathcal{B}$). The half-lines starting at the origin and passing through the extreme points of $\mathcal{B}$ are called *fundamental directions* of the gauge, denoted by $d_1, ..., d_\sigma$. The fundamental directions, formed by these points and the origin, produce affine cones in $\mathbb{R}^n$. These cones are called *fundamental cones in* $\mathbb{R}^n$.

Remark 4. For the sake of simplicity, we assume the unit ball corresponding to the distant function of **(OP)** is convex. For the remedy of such a problem and more information about unit balls, you can refer to [15], [16] and [20].

Lemma 2. The optimal solution of **(OP)** lies on the boundary of gc if q is not located inside gc.

Proof. Suppose x', inside gc, is the optimal solution such that for all $x \in gc, dist(q,x) \geq dist(q,x')$. Then, there exists $\delta > 0$ such that the ball $B(x',\delta)$ is located completely inside gc. The line segment passing through q and x' cuts the boundary of $B(x',\delta)$ at y and y'. Without loss of generality, assume q, y' and x' are located in this line such that y' is between q and x'.

On the other hand, since q is located out of gc, the unit ball with the origin on q can be chosen small enough such that y' and x' are out it. Then if we enlarge the unit ball, it will first touch y' and then x', since y' is located between x' and q on the line segment passing through them. Therefore

$$dist(q,x') > dist(q,y').$$

This is a contradiction, since y' is feasible in gc is and gives a better objective value than x'. Therefore, the optimal solution lies on the boundary of gc.

Lemma 3. Let x' be a point on the boundary of gc. The unit ball $\mathcal{B}$ with the origin on q will touch x' at one of its faces $\mathcal{F}$. Let H be the hyperplane containing $\mathcal{F}$. Without loss of generality assume q is located in $H(x) \leq 0$. Prove that all the points inside $H(x) > 0$ have a greater distance to q than x'.

Proof. The proof is very trivial. H is the separating hyperplane of $\mathcal{B}$ and the points inside $H(x) > 0$. We know $dist(x',q) \geq dist(x,q)$ for all $x \in B$ and $dist(x',q) < dist(x,q)$ for all $x \notin \mathcal{B}$.

Since $\mathcal{B}$ and $H(x) > 0$ are completely separated therefore $dist(x',q) < dist(x,q)$ for any x in $H(x) > 0$.

These two lemmas will lead us to choose a more intelligent cutting hyperplane.

6.3 The Algorithm

Bellow, we establish an algorithm using this more intelligent cutting hyperplane.

Input: (**OP**) corresponding to a gc and a feasible point of gc called ub.

1. We first check wether q is inside the feasible region of **(OP)**, i.e. q is inside the generalized case. Therefore, check whether $h_j(q) > 0$ for some j. Let $J := \{j \mid h_j(q) > 0\}$. If J is empty stop, q is inside the generalized case and the distance is 0.
2. Solve the corresponding (**CP**) with Topkis-Veinott algorithm. Let lb be the determined lower bound.
3. Now, we check whether l_b is also a solution of **(OP)**. For this purpose, we check whether $h_i(lb) \leq 0$ for all $i = 1, ..., m_2$ (remember, these are the non-convex constraints). If this is the case we stop and lb gives the exact distance between q and gc.

4. Now, we move from u_b to the boundary of the feasible region of **(OP)**. Therefore, we intersect the line passing through ub and q with $h_j(x) = 0$ for all $j \in J$. Let x' be the closest intersection point to ub for $j \in J$. Set $ub = x'$.
5. Consider the line segment connecting ub and q. Let mid be a point on this line segment such that

$$dist(q, mid) = (dist(q, ub) + dist(q, lb))/2. \tag{1}$$

6. Retrieve the unit ball $\mathcal{B}$ with origin on q touching mid on a face $\mathcal{F}$. This can be simply done by finding the fundamental cone including mid and moving the face of the unit ball of this cone to mid. Let H be the corresponding hyperplane containing $\mathcal{F}$. Suppose q is located in $H(x) \leq 0$.
7. Consider the following constraints:

$$\begin{aligned} &h(x) \leq 0, \\ &H(x) \leq 0, \\ &dist(q, x) - dist(q, mid) \leq 0. \end{aligned}$$

   Let's call these $m + 2$ constraints $g_i(x) \leq 0$ for $i = 1, ..., m + 2$.
8. Suppose $g_i(mid) \leq 0$ for $i \in I$ and $g_i(mid) > 0$ for $i \notin I$. Solve the corresponding (**FP**) of $g_i(x)$.
9. Consider (x^*, z^*) where z^* is the optimal objective value of (**FP**).
   If $z^* = 0$ set $ub = x^*$ and add $H \leq 0$ to (**OP**),
   else set $lb = mid$ and add $H > 0$ to (**OP**).
10. If a stopping criterion is not satisfied, go to step 4.

The steps 5 to 10 are also shown in Figure 2. The point mid, the unit ball and its face touching mid and H are represented in figure. Here, $H \leq 0$ means the half space to the right hand side of H. The feasible region of **(OP)** or $h(x) \leq 0$ is also represented by its original name gc. If we intersect these spaces together with $dist(q, x) - dist(q, mid) \leq 0$ we get $\mathcal{R}$. If $\mathcal{R}$ is not empty as it is the case in this figure, by solving **(FP)** we will find a point which is the new upper bound. If $\mathcal{R}$ is empty, then mid is the new lower bound.

Now we apply the algorithm as a black box for tightening the bounds of the (**OP**) for each gc separately and in parallel. We can stop further investigation for a gc whenever its lb is bigger than the ub of another gc. This is going to be our stopping criterion.

Remark 5. It's obvious that mid need not to be exactly as in (1). It can be chosen according to the lower and upper bounds of other gc's.

6.4 Convergence and Complexity

Lemma 4. Prove that the algorithm is not exhaustive.

Proof. Obviously $lb \neq ub$, otherwise the stopping criterion is satisfied. Due to the construction of mid:

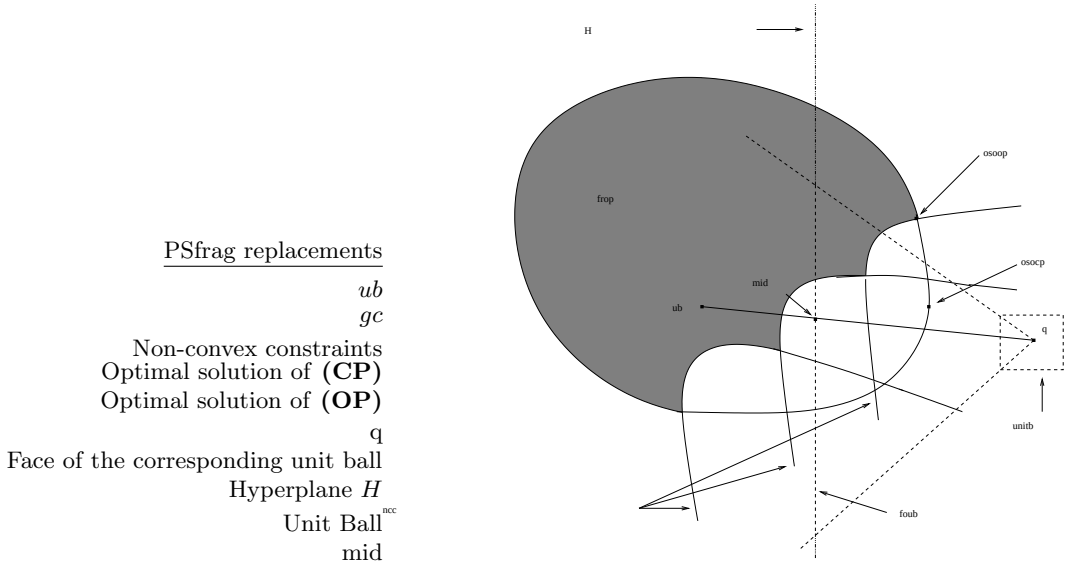

Fig. 2. An instant of the algorithm

$$dist(q, lb) < dist(q, mid) < dist(q, ub).$$

where $dist(q, mid)$ is exactly in the middle of the interval $[dist(q, lb), dist(q, ub)]$. Suppose $z^* = 0$, then $dist(q, x^*) \leq dist(q, mid)$ and since we set $ub = x^*$, then the new interval has at most half of the length of the old one.
If $z^* > 0$, then $lb = mid$ which again the new interval is exactly half of the old one. Therefore, we have a sequence of intervals getting smaller and smaller. So the algorithm is not exhaustive.

The algorithm is quite flexible in the sense that the first 4 steps or some of them can be omitted. These 4 steps are some heuristical methods that depending on the behavior of the problem might accelerate the procedure of tightening the bounds. To get more insight about the behavior of them, we refer the reader to [15].

Let ub_1 and lb_1 be the initial points giving upper and lower bounds respectively. ub_1 is the initial feasible point and lb_1 might be q itself or the solution of (**CP**). If we set $l = dist(q, ub_1))$ and , $T = O((\mathbf{FP}))$, then the asymptotic running time of the algorithm is $O(T\log(l))$.

6.5 Feasibility Problem

Efficiency of our algorithm is crucially dependent on the feasibility problem. There exist several references considering different methods to solve this problem. In [1] and [3], the standard line search method has been developed. The problem

of finding a solution for a nonlinear system of equations is investigated in [9]. Finally some global optimization methods in [11] which solve (**FP**) without considering the nature of it.

We propose to use a line search algorithm for the feaibility problem. This can be done by searching inside H starting from mid. It's enough to search only inside H, since gc is a connective region and if H cuts it somewhere, then there exists a point $\bar{x}$ where $H(\bar{x}) = 0$ and $\bar{x} \in \mathcal{R}$. Note that $H(mid) = 0$, therefore mid is a good starting point.

7 Summary and Conclusion

In this paper, we have analyzed the problem of similarity assessment for generalized cases that are represented through constraints over an n-dimensional Real-valued vector space. We have shown that the difficulty of this problem depends on whether the constraints include just convex constraints or also non-convex constraints. For the convex constraints and if also the similarity measure is convex, the Topkis-Veinott method can be easily applied to determine exactly the similarity between a point query and generalized cases. If the similarity measure is not convex or the generalized case contains also non-convex constraints, the problem is more difficult. For this situation, we proposed a new algorithm that allows to incrementally compute sequences of upper and lower bounds for the similarity and we show the convergence of the algorithm. This algorithm can be applied to construct a retrieval algorithm for generalized cases similar to the fish and shrink retrieval algorithm [19]. It allows to rank generalized cases without the need to exactly compute all similarity values.

In general this paper is an indication for the fact that optimization theory provides a variety of methods that are useful for similarity assessment for generalized cases. Although, this paper is a very important step towards solving the similarity problem for generalized cases, there are still many open problems to be addressed. First, the feasibility problem that has to be solved as part of the algorithm (see section 6.5) is still a difficult problem in optimization and it has to be analyzed in more details in the context it occurs in our algorithm. Second, the proposed method is limited to Real-valued case representations only. The problem of similarity assessment for generalized cases must also be analyzed for case-representations including discrete types and constraints and also for mixed representations.

Currently, practical experiences with this algorithm concerning efficiency in real domains, such as the IP reuse domain, is still missing and it is an important part of our future work as well.

Acknowledgements

This work has been partially funded by the BMBF project IPQ. The authors want to thank Ivo Vollrath and Michael M. Richter for many discussions on the topic of this paper.

References

1. M. Avriel and B. Golany. *Mathematical Programming for Industrial Engineers.* Marcel Dekker, INC., 1996.
2. Ray Bareiss. *Exemplar-Based Knowledge Acquisition: A unified Approach to Concept Representation, Classification and Learning.* Academic Press, 1989.
3. M. S. Bazaraa, H.D. Sherali, and Shetty C.M. *NonLinear Programming, Theorey and Algorithms.* 2nd Edition, Wiley, 1993.
4. R. Bergmann. *Experience Management: Foundations, Development Methodology, and Internet-based Applications.* Springer, forthcomming, 2002.
5. 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.
6. 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.
7. Ralph Bergmann. *Effizientes Problemlösen durch flexible Wiederverwendung von Fällen auf verschiedenen Abstraktionsebenen.* DISKI 138. infix, 1996.
8. L. Blum, F. Cucker, M. Shub, and Smale S. *Complexity and Real Computation.* Springer, 1997.
9. H.-D. Burkhard and M.M. Richter. Similarity in case-based reasoning and fuzzy theory. In S.K. Pal, T.S. Dillon, and D.S. Yeung, editors, *Soft Computing in Case-Based Reasoning*, chapter 2. Springer, 2000.
10. R. Horst and H. Tuy. *Global Otimization: Deterministic Approaches.* Springer, 1993.
11. Kefeng Hua, Ian Smith, and Boi Faltings. Integrated case-based building design. In Stefan Wess, Klaus-Dieter Althoff, and Michael M Richter, editors, *Topics in Case-Based Reasoning. Proc. Of the First European Workshop on Case-Based Reasoning (EWCBR-93)*, Lecture Notes in Artificial Intelligence, 837, pages 436–445. Springer Verlag, 1993.
12. Janet L Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory.* PhD thesis, Yale University, 1980.
13. Jeff Lewis. Intellectual property (IP) components. Artisan Components, Inc., [web page], http://www.artisan.com/ip.html, 1997. [Accessed 28 Oct 1998].
14. B. Mougouie. Optimization of distance/similarity functions under linear and nonlinear constraints with application in case-based reasoning. Diplomarbeit, Max-Planck Institut für Informatik, Saarbrücken, Germany., 2001.
15. S. Nickel. Convex analysis. Technical report, Department of Mathematics, University of Kaiserslautern, Kaiserslautern, Germany, 1998.
16. Lisa Purvis and Pearl Pu. Adaptation using constraint satisfaction techniques. Lecture Notes in Artificial Intelligence, 1010, pages 289–300. Springer Verlag, 1995.
17. S Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:277–309, 1991.
18. 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.
19. A. Schoebel. Lecture notes in location theory. Technical report, Department of Mathematics, University of Kaiserslautern, Kaiserslautern, Germany, 2000.
20. Thomas Wahlmann. Implementierung einer skalierbaren diskreten Kosinustransformation in VHDL. Diploma thesis, University of Siegen, 1999.