Case Completion and Similarity in Case-Based Reasoning

Hans-Dieter Burkhard

Institute of Informatics
Humboldt University, D-10099 Berlin, Germany
hdb@informatik.hu-berlin.de

Abstract. Case Completion investigates cases for complex problem solving tasks using Case-Based Reasoning. Such tasks consist of several steps, and related cases should support each intermediate decision. Related cases are of a constraint like style and need the handling of partial matching for retrieval. Related similarity measures are investigated, and the implementation by Case Retrieval Nets is proposed.

1 Introduction

Case-Based Reasoning (CBR) is a model for acting by experience. Cases describe former episodes of problem solving for later reuse. Important issues are organization of memory (case base and its maintenance), reminding (retrieval), and reuse (adaptation and application).

Case-based techniques are used in practice for many years, e.g. for customer support [11]. CBR is used for cognitive modeling from the very beginning [9], and for control of autonomous systems in dynamic environments [12]. The technology of CBR applications is often described by the R^4 cycle ([1]) with the phases *Retrieve*, *Reuse*, *Revise*, *Retain*. Additional phases for maintenance are discussed in [8].

Cases are usually organized in a rule-like style: If there is a problem, then the case provides a solution. Such cases consist of a problem-part and a solution part. In contrast, the concept of Case Completion ([5]) considers cases as constraints: Given some piece of knowledge of a problem, ask for possibly useful information by experience (case knowledge) from the past.

Retrieval of constraint like cases needs handling of partial information, i.e. of "missing values". An appropriate measure of similarity is needed, where similarity has to approximate usefulness of cases, and acceptance of users, respectively.

The paper introduces Case Completion first. Section 4 discusses similarity and distance measures, based on some results from [6]. There is some common understanding, that both – similarity and distance – are more or less equivalent. Actually, there are some differences which are further discussed in Section 5. Appropriate linked index structures for Case Completion are implementated by Case Retrieval Nets (Section 6).

2 Case Completion

2.1 Cases behind "Problem + Solution"

Case-Based Reasoning is a problem solving method: Given a problem we have to find a suitable solution. In a simple setting, cases are split into a problem and a solution part. Given a new problem we search for cases with related ("similar") problems in the case memory. The solutions of those cases are supposed to be useful (maybe after some adaptation) for the new problem. There is an underlying hypothesis: Similar problems have similar solutions.

However, problem solving usually does not start with a complete problem description sufficient for the identification of a case with a final solution. Instead, the process of completing a task up to the final solution usually depends on a number of decisions and related subsequent steps. In the case of diagnosis and repair we start with some first observations, e.g. the vacuum cleaner is not working. Then we test if the cable is plugged in. If so, we make a first trial for "repair" like switching on and off several times. If it does not work, we may test if there is power at all, e.g. if other devices are working and if the power supply is ok. We find out that the power supply is ok. Something must be wrong with the switch, the cable, the motor or some other part of our vacuum cleaner. We may decide to give it away for repair, to buy a new one, or to continue with further more specific tests and trials for repair.

We have a similar situation in design processes: There we start with a raw specification, then we add more specific design decision and finally end up with a complete specification.

In general, the course of actions in complex problem solving processes is initially open. The intermediate actions and the final result do not depend from the initial problem only, they depend also on intermediate human decisions. According to different possible decisions we end up with different stories and different final solutions. It is difficult to apply the simple scheme of "problem" and "solution" to such processes. Nevertheless, the whole story of problem solving can be useful in future similar situations. The whole story is in fact a case. Such a case provides useful hints at the different steps of problem solving. It describes all actions and results during the whole process of solving a problem. It is useful for reuse at various intermediate steps.

Case Completion is an attempt to organize CBR systems to store and reuse such complex cases. Case Completion describes the completion of a task (of a "new case") in the real world, usually consisting of several steps. The whole episode of problem solving is stored as a single case in the case memory. We do not have the distinction between problem and solution anymore, and hence the retrieval of cases must be more flexible. For later reuse, a case has to be accessible by arbitrary subsets of its indexes.

Such a retrieval is similar to the retrieval from a data base: Given some indexes, a complete relation can be retrieved from a database. But for CBR, we have to deal with similarity matches. Given some indexes (by a query), we look for cases matching the given indexes according to some similarity measure.

The distinction between "problem" and "solution" treats cases as rules: If the "problem" matches the case, then the "solution" can be adapted from that case. In Case Completion, a case is more like a constraint than like a rule. The constraints have been observed by former experiences. Any item in such a case may be part of problem description and part of proposed solution at different steps of problem solving. By past experience we may know a case where the vacuum cleaner did not work. The control light was off. There was no power due to a short cut. The reason for the short cut was a defect of the cable. In a future problem with our vacuum cleaner, we may again have the first symptoms *not working* and *control light off*. Then the item *no power* is a proposal for an intermediate test by the former case. In the next step, if power is indeed missing, *no power* becomes part of the problem description. Now we look for cases with *not working*, *control light off*, *no power*. The former case proposes the solution *cable defect*, while some other case might propose *cable not plugged in*. If both proposed solutions do not apply, some other cases might be found. We will continue the discussion in Section 2.3.

2.2 Completing a Task

We talk about a "task" for more complex problem solving processes consisting of several steps. Initially, a task is specified in some way, but the specification is *incomplete*. This means that the known information is not sufficient for the successful *completion* of the task. For illustration, we will give examples from different domains in the following.

Diagnosis: First symptoms are given, but not enough to identify the malfunction and to propose the appropriate repair steps.

Design: First functional descriptions are given, but not enough to choose the appropriate parts and to fix their layout.

Consulting: First ideas of a customer are given, but not enough to make a clear specification and to propose a suitable offer.

As the completion of the task is going on, more and more information is collected and specified until the (hopefully) successful end. After completion, a new experience was learned and may be reused later for related tasks. The relevant content of this experience is called a *case*. The *process* of elaborating the task under the aspect of collecting case relevant information is the essence of *Case Completion*.

The finally completed case depends not only on the initially given problem. It depends also on the subsequent steps of the problem solving processes. Different processes lead to different cases. Each of them might be valuable as experience for later re-use.

The underlying process usually includes several steps:

Diagnosis: Specify and perform further tests and (possibly experimental) steps of repair. The identification of the malfunction and/or the repair is only the final step in a chain of attempts.

Design: Perform a chain of design decisions (possibly with backtracking) up to a final layout.

Consulting: Discuss and specify the wishes of a customer under different aspects and available alternatives.

After each step, we have collected more information for the completion of the task. This information can be used for the next step (e.g. we may have identified a short circuit as the reason for missing power in our previous step, next we ask for the reason of that short).

2.3 Case Completion Supports Complex Tasks

How can CBR give support for the process of Case Completion? The essence of Case Completion is the collection of further information. This information is obtained by the outcomes of activities in the *real world*. CBR is not the source of this information, but it *can guide* the activity. Possible hypotheses for underlying (but still unknown) circumstances and possible outcomes of possible activities are checked on the base of former experiences, given by the cases: CBR is a matter of proactivity.

The complete cases should be useful in each step (e.g., for the choice of a next test step as well as for a final identification of the diagnosis). This requires the treatment of different amounts of information. Only few *Information Entities* are given at the start of a new task, and each step adds new pieces of information. Thereby, an Information Entity may change its character during the process as follows:

Starting with the first symptom *missing power*, the Information Entity *short circuit* is a possible explanation (which may be proposed by a related case). If it appears to be true in reality, then the Information Entity *short circuit* becomes part of the known information for the considered diagnosis task. Then it may be used to find cases which give hints for reasons of shorts or for repair.

Following the classical approach (case = problem + solution), we would need different cases. For example, special cases with *short circuit* in the solution part, and other ones with *short circuit* in their problem parts. Typically, those cases were obtained during more complex tasks. Why not use the total description of the complex task as cases? In (pure) Prolog, parameters may serve for input or output values: A query specifies values for certain parameters, the answer provides values for the remaining parameters. In a similar way, the query to a case base should specify the known Information Entities of a certain task. The answer of the CBR system should provide further Information Entities which have occurred together with the known ones in the past. As already discussed above, classical CBR cases have rule-form: *IF problem, THEN solution.* Now the cases should have constraint form: Cases are constraints to common occurrences of Information Entities. It connects items (Information Entities) which have appeared together in a problem solving process. Each subset of those items may appear in a query, and the related cases show possible completions (where

existing constraints are satisfied because a case is an experience from reality). A technical framework for retrieval will be given in Section 6.

Case Completion is the collection of relevant information *while completing a task*. The completion of this task has to be done anyway, – in CBR we consider this process under the viewpoint of collecting and using experiences.

> *Case Completion is a general view to the process of CBR problem solving. A new case appears with incomplete and vague information. The process of problem solving is characterized by collecting new information while making decisions and acting in the real work until a satisfactory level is reached. CBR can guide this process by comparison with more complete cases from the past. Completed cases can be stored as new experiences for later reuse.*

CBR systems supporting such processes have to be designed as interactive systems. The users get information about complete cases, and they can choose the information they consider to be relevant from that cases. They can combine information from old cases to get new proposals. Complete cases provide a good base for *argumentation* (e.g., in medicine and in jurisprudence).

The technology of CBR applications is often described by the R^4 cycle ([1]) with the phases *Retrieve – Reuse – Revise – Retain*. It starts with the input of a new "problem" and ends up with the integration of a new case (containing the problem description and its confirmed solution). The solution may have been proposed by the CBR system, and it could have been revised w.r.t. the practical results.

Case Completion differs from this picture since it covers several intermediate problem solving steps. These steps can be supported by cases from the CBR system (*retrieve*, *reuse*) and evaluated by reality (*revise*). After each intermediate step, the process continues for a next step, again with *retrieve*, *reuse*, *revise*. The supporting cases in the next step may be more or less different from the cases of the previous step. The "new case" is completed only if the whole task is completed, and then a single retain step can add this case to the case memory.

The implementation of appropriate CBR systems needs the handling of incomplete information. Cases are composed of Information Entities. They are used for retrieval as indexes – as far as they are known at an intermediate step of Case Completion.

3 Cases and Queries as Sets of Information Entities

3.1 Information Entities

A case is the result of a Case Completion process. Each step of that process adds some new Information Entities. The current situation during the elaboration of a task is described by the Information Entities known at that time point. The final case, as it later may appear in the case memory, is a completed set of Information Entities ("completed" refers to the state of information at the end of the task):

1. The collected Information Entities result from the real world (e.g., as the outcome of a test, a decision in an intermediate design step, or a specified customer requirement in consulting). They are not the direct result of a CBR process – CBR is used to *propose* the next step, e.g., the next test in diagnosis or the next design decision. The test outcome *might*, but need not, be the expected result according to former cases from the case memory, and we *might*, but need not, adapt the design proposal given by a CBR system.
2. The number of Information Entities in a case may vary. It is up to a (human) decision at which time point the task is finished (e.g. the correct functioning of a device after replacing a special part may be sufficient without elaborating a detailed diagnosis based on a complete set of symptoms). It is usually not realistic to think about fixed formats of diagnosis reports in practice.
3. The Information Entities which are later used for retrieval (which appear as indexes in the case memory) may be only a subset of the information collected during Case Completion.

During the process of Case Completion, the current situation is described by the set of known Information Entities. In the following, we restrict the usage of the term "Information Entity" to those which are used as indexes for retrieval. Besides them, further information may be provided for the users (cases might be pointers to more detailed descriptions).

Definition 1 (Cases and Queries Based on Information Entities).
An Information Entity (IE) is an atomic part of a case or a query: Cases and queries are sets of Information Entities.
More formally: The set of all (potential) Information Entities in a given domain is denoted by E.
A case is a set of Information Entities: $c \subseteq E$.
The set of cases (in the case memory) is denoted by C, $C \subseteq 2^E$.
A query is a set of Information Entities: $q \subseteq E$.

Information entities may be of various types. In some applications, Information Entities are simply attribute-value pairs. Information entities may explicitly express certain relations (structures) in the cases. They may be more complex as long as they may serve as indexes with related similarity measures.

Information entities may describe concepts, e.g. for language processing. For textual CBR, the distinction between concepts and strings (appearances of the concepts) has proven to be useful: Concepts are modeled by Information Entities, related strings are simply mapped to them. Therewith, all grammatical forms (e.g. *go*, *goes*, *gone*, ...), and even translations (e.g. *gehen*, *aller*, ...), are mapped to a single Information Entity (e.g. *concept "go"*,). Similarity is then defined between Information Entities (e.g. *sim(concept "go", concept "run") = 0.8*).

3.2 Case Completion using Information Entities

Useful cases from the case memory must be provided for Case Completion. A user asks a query to the system, and the answer should help to solve a problem.

Usefulness is difficult to judge a priori. A somewhat more applicable notion is acceptance of users. But still, for the designer of a CBR system, it is not evident which cases the one or the other use might accept as an answer for his or her query.

The expected user *acceptance* of a case for a query is approximated by corresponding Information Entities. If a case contains Information Entities which correspond to a great extend to the Information Entities of a query, then the user is assumed to accept this case as a candidate for useful information: Useful information is expected from *the cases matching the query by related Information Entities.*

Corresponding Information Entities in the query and the selected cases may concern:

Diagnosis: Certain known symptoms, results of first attempts for repair.
Design: Certain desires for functionality, intermediate design steps and decisions.
Consulting: Certain specifications and further "vague ideas".

Useful hints for future steps are expected from the selected cases. This information may have the form of further Information Entities, or it may appear as additional information:

Diagnosis: Which further tests have been performed in former situations with what results? We can look for discrimination tests (differential diagnosis) if there remain several cases with different final diagnoses.
Design: Which further design steps may lead to what results? Are there risks for later conflicts w.r.t. to some constraints?
Consulting: Which further specifications meet the requests of the customer? Are there alternative offers?

Further Information Entities provided by the cases can be processed by the CBR system for adaptation purposes.

3.3 Usefulness - Acceptance – Similarity

Usefulness of a case in the Case Completion process depends on real world circumstances which are not completely known at retrieval time. Cases are proactively used, as described in Section 2.3, for making proposals. The evaluation of those proposals is possible only after the response from reality to the user's activities (maybe even after completing the whole task). As widely discussed in the literature, usefulness is only an *a posteriori* criterion.

The retrieval from case memory is based on the (vague) matching of certain Information Entities. Usefulness of former cases is not restricted to those cases which are similar to a given query for *all* Information Entities. Cases may contain information entities which have no counterpart in the recent query. It is also possible that some Information Entities of the query are not present in useful cases. Examples are:

Diagnosis: A query may ask only for the recently known symptoms, while the cases may contain complete information about solved tasks including all tests, experiments, knowledge about final diagnosis and repair.
A query may refer to a symptom which was not recorded in a previous case. It is even possible that the query and a useful case are different for some symptoms; e.g., the age of a patient may not be important for the diagnosis, but for the therapy.

Design: A query may ask for the layout of a certain detail, while the cases contain information about complete devices.
A query may contain a specification of the material, but some useful case giving a construction advice might refer to another material.

Consulting: A query may contain only some vague ideas of the customer, while a case contains completely specified offers.
A query may give limitations for the price, but under certain circumstances the customer accepts a more expensive offer.

The cases in the case memory are indexed by all Information Entities which are *potentially* relevant for the retrieval. The features which are important for queries can not be fixed at design time. Moreover, the importance of an Information Entity may change from one query to another (i.e., the sex is important for some, but not for all diseases). *Compromises*, as in the consulting example above, may even relax special features *after* looking for available cases (cf. Section 5.5).

Formal approaches to acceptance are provided by similarity measures or distances. Both notions are often considered as equivalent: The nearest neighbors according to a certain distance are considered as the most similar objects. Actually, there are some differences, which will be discussed in the next sections.

4 Similarity

4.1 Similarity and Distance: Primary Considerations

Similarity itself is not a fixed notion, it depends on the aspects under consideration:

> *Any two things which are from one point of view similar may be dissimilar from another point of view.* (Popper)

or as another quotation:

> *An essay is like a fish.* (Tversky)
> (Why: Both have head, body, tail etc.)

It is a central problem in the design and maintenance of CBR systems to adopt a notion of similarity such that the following assumptions are satisfied:

1. Similarity between a query and a case (hopefully) implies usefulness of the case for the problem to be solved. The user should understand why a case was presented (acceptance).

2. As cases can be more or less useful for problem solving, similarity should provide a quantitative measurement leading to an ordering of cases according to expected usefulness.
3. Similarity must be based on a priori known facts. Complex calculations are less helpful.

4.2 Some Basic Notions

Similarity and distances are considered over some universe U (e.g. $SIM(u, v)$ denotes the similarity value between two objects u and v from U). According to our definition from above, we consider queries q and cases c as sets of Information Entities: E denotes the set of all Information Entities, hence $U = 2^E$.

A case base C is a (usually) finite set of cases. A query may be any subset $q \in E$, while a case is a subset from E which is contained in C: $c \in C \subseteq U = 2^E$.

For sake of simplicity, we will restrict ourselves sometimes to finite sets of feature (attribute) values taken from the reals. Then we consider real valued vectors u over the universe $U := \mathcal{R} \times \mathcal{R} \times ... \times \mathcal{R}$. Cases c and queries q are then vectors from U.

Similarity can be considered as

- Binary relation R_{SIM} between objects from U (where $R_{SIM}(u, v)$ stands for "u is similar to v"):

$$R_{SIM} \subseteq U \times U. \tag{1}$$

  Example: Two things are called similar, if they coincide w.r.t. at least 2 of 3 features:

$$R_{SIM}([x_1, x_2, x_3], [y_1, y_2, y_3]) \Leftrightarrow \exists i, j : 1 \leq i < j \leq 3 : x_i = y_i \wedge x_j = y_j. \tag{2}$$

- Measurement of the degree of similarity by a similarity measure *SIM* (where $SIM(u, v)$ stands for the degree of similarity between u and v). The degree may range over some interval S:

$$SIM : U \times U \rightarrow S. \tag{3}$$

  According to Fuzzy Theory, Stochastics, Certainty Theory etc., the range S is often the real interval $[0, 1]$. Varying the example from above, we can consider the *simple matching coefficient*:

$$SIM([x_1, x_2, x_3], [y_1, y_2, y_3]) := \frac{1}{3} \cdot card(\{\ i \mid x_i = y_i\}). \tag{4}$$

  More sophisticated measures rely on differences $x_i - y_i$ between the feature values (as real numbers) , e.g.:

$$SIM([x_1, x_2, x_3], [y_1, y_2, y_3]) := \frac{1}{3} \cdot \sum_{i=1,2,3} \frac{1}{1 + |x_i - y_i|}. \tag{5}$$

– Neighborhood according to a distance *DIST* (where $DIST(u,v)$ stands for the distance between u and v):

$$DIST : U \times U \to S. \tag{6}$$

An often used distance measure with $S = [0, \infty)$ is the *Manhattan-Distance*:

$$DIST([x_1, x_2, x_3], [y_1, y_2, y_3]) := \sum_{i=1,2,3} |x_i - y_i|. \tag{7}$$

4.3 Relations between the Different Notions

There are some relations between these approaches. At first, binary relations R_{SIM} are equivalent to special similarity measures *SIM* with only the two binary values 0 and 1.

Vice versa, similarity relations R_{SIM} can be defined by the "most similar" objects according to a similarity measure *SIM*. The concept of a *nearest neighbor* is often used for such purposes. We consider a subset $C \subseteq U$ (the case base in CBR). Then $c \in C$ is called a nearest neighbor of an arbitrary object $u \in U$ (a query in CBR) if it satisfies the following definition:

$$NN_C(u,c) :\Leftrightarrow \forall c' \in C : SIM(u,c) \geq SIM(u,c'). \tag{8}$$

Equivalently, for a distance measure we can define the nearest neighbor concept by

$$NN_C(u,c) :\Leftrightarrow \forall c' \in C : DIST(u,c) \leq DIST(u,c'). \tag{9}$$

In a related way, the concept of the "'*k nearest neighbors*"' is defined.

By another common concept, objects u, v are called similar if their degree of similarity exceeds a certain threshold value b:

$$R_{SIM}(u,v) := SIM(u,v) > b. \tag{10}$$

Equivalently, for a distance measure we would use $DIST(u,v) < b$.

Related concepts are basic in CBR: The most (hopefully) useful cases from a case base C are selected according to a query q by such concepts.

Extending the concept of a nearest neighbor, we can consider the ordering between the pairs of objects induced by a similarity/distance measure:

$$ORD_{SIM}(x,y,u,v) :\Leftrightarrow SIM(x,y) \geq SIM(u,v), \tag{11}$$

$$ORD_{DIST}(x,y,u,v) :\Leftrightarrow DIST(x,y) \leq DIST(u,v). \tag{12}$$

We call two (similarity or distance) measures m_1 and m_2 *relationally compatible* iff

$$\forall x,y,u,v \in U : ORD_{m_1}(x,y,u,v) \Leftrightarrow ORD_{m_2}(x,y,u,v). \tag{13}$$

Similarity and distance measures are often considered as equivalent notions. On a first glance, the ordering according to high similarity can be substituted by

the ordering according to low distances. The nearest neighbor concept defines similarity on the base of a distance.

More formally we consider an arbitrary one-one order inverting mapping f of a subset S of the reals to itself. It induces for a similarity measure with range S a relationally compatible distance measure. If $S = [0, 1]$ is the range of SIM then $f(1) = 0$, and $SIM(x, x) = 1$ would be translated to $DIST(x, x) = 0$ for the induced distance measure. Vice versa, a distance measure is transformed into a relationally compatible similarity measure. Measures with different ranges can be mapped by related functions f, too.

In this principle sense distance and similarity are in fact mathematically equivalent. But, certain "common sense" properties have no correspondence. For example, distances should be symmetric: $DIST(x, y) = DIST(y, x)$, while similarity measures need not. Similarity can be used between different levels of abstraction: We can describe a penguin using similarity to a bird, but we would not describe birds by similarity to penguins. We will go more into the details below.

4.4 The Range of the Measures

We assume that the range S of the measures is always a (bounded or unbounded) interval of real numbers. Most of our considerations are related to $S = [0, 1]$ which has two major characteristics:

- There is a maximal value for the similarity, and a minimal value for the distance, respectively.
- There are no negative values.

A second possibility is an unbounded range, say $S = [0, \infty]$. The impact for similarity is some difficulty for defining reflexivity (see below), if there is no maximal similarity value. For the distances the consequence is less dramatic because it only means that there are no maximal distances.

A third possibility is the existence of negative values. For distances this is not intuitive, and it would rise difficulties with reflexivity when $DIST(x, x) = 0$ is not the minimal value. An unwanted consequence could be elements which are not the nearest neighbor of themselves. There are no direct consequences for similarity measures. But for combining global measures from local ones, negative similarity values could be of some interest (cf. Section 5.5).

4.5 The Domain of the Measures, Missing Values

The purpose of similarity is a ranking over the case base $C \subseteq U$ according to usefulness (acceptance) with respect to a given query $q \in U$. Hence similarity (distance) is considered in praxis only over $C \times U$ and not over $U \times U$. We have this already regarded for the nearest neighbor approach.

For the moment, we consider only set theoretic relations between queries q and cases c composed from Information Entities, $U = 2^E$. (We do not consider

similarities between Information Entities). We might have $q = c$, $q \subset c$, $c \subset q$ or none of them. How does it affect similarity?

As a concrete example we consider the query $q = \{b, c\}$ and the cases $c_1 = \{b, c\}$ $c_2 = \{b\}$,$c_3 = \{a, b, c\}$, $c_4 = \{a, b\}$, $c_5 = \{a\}$. What is the appropriate ranking for the cases: Which case provides most useful information for the user? c_5 is not a candidate. But for the remaining ones we have to remind that the Information Entities are used as indexes for retrieval, and that cases may contain further background information.

Then it depends on the concrete application, e.g. from design decisions. There is no general argument how to rank the remaining cases. Missing Information Entities may appear for different reasons, they may be unknown or not relevant. An unknown Information Entity in a case might be irrelevant (like an unmentioned attribute in a rule): In this situation, c_2 might be as good as c_1. An unknown Information Entity in a query might have the same value as in the case (optimistic view): In this situation, c_3 might be as good as c_1. Combining both arguments, c_4 might be as good as well.

We may even find arguments that c_3 is more useful than c_1, because it contains more information, e.g. an additional repair proposal for a diagnosis task.

Related discussions can be performed for cases as feature vectors. There we may have unspecified or unknown values for some attributes.

The consequences of treating unknown and unspecified values are sometimes surprising. Even some "obvious" axioms may be affected, as we will discuss below. It is a difficult task for the designer of a CBR system to find the appropriate similarity assignments leading to useful rankings of cases. Moreover, the designer must try to realize acceptance by the user with a reasonable and easily understandable similarity measure.

4.6 Common Axioms for the Measures

There are various possible axioms for distance measures as well as for similarity measures in the literature. We discuss some of them and assume that the range is $[0, 1]$.

Reflexivity

$$SIM(x, x) = 1 \quad \text{and for distances:} \quad DIST(x, x) = 0. \tag{14}$$

The opposite direction can be demanded, too, – as in the case of a metric:

$$SIM(x, y) = 1 \rightarrow x = y \quad \text{and} \quad DIST(x, y) = 0 \rightarrow x = y. \tag{15}$$

We call this *strong reflexivity.*

Symmetry

$$SIM(x, y) = SIM(y, x) \quad \text{and} \quad DIST(x, y) = DIST(y, x). \tag{16}$$

Triangle inequality

$$DIST(x, z) \leq DIST(x, y) + DIST(y, z). \tag{17}$$

(For similarity: See the discussion below.)

Reflexivity Reflexivity seems to be an obvious axiom. What else should be more similar than the object itself: $SIM(x, x) = 1$? It guarantees that u is always a nearest neighbor of itself. Reflexivity leads to the result, that the best answer to a query q is the query itself (as far as it is contained in the case base).

There might be further objects $y \neq x$ which are acceptable as well, i.e. we might have $SIM(x, y) = 1$ for them. We need not insist in strong reflexivity. Related distances with $DIST(x, y) = 0$ are not metrics.

But, as discussed in Section 4.5, the most useful case might even be a proper superset of the query. Such a case can provide additional Information Entities. According to the example given there, the case $\{a, b, c\}$ can be the *best* answer to the query $\{a, b\}$ if c stands for additional useful information. Then we have $SIM(\{a, b\}, \{a, b\}) < SIM(\{a, b\}, \{a, b, c\}$. In the consequence we have $SIM(\{a, b\}, \{a, b\}) < 1$ – no reflexivity at all.

Symmetry As already mentioned, symmetry is a commonly supposed property of distances (especially of metrics). In contrast, similarity is sometimes not symmetric, e.g. in the case of "directed" comparisons in daily life:

- a daughter is similar to her mother, but not vice versa,
- a penguin is similar to a bird, but not vice versa,
- a query may be similar to a document, while the document might not be similar to the query.

The last point can again be illustrated by our example from above, where we have $SIM(\{a, b\}, \{a, b\}) < SIM(\{a, b\}, \{a, b, c\}$. The case $c = \{a, b, c\})$ is especially useful because it provides additional information to the query $q = \{a, b\}$. Vice versa, a case $c = \{a, b\}$ has no additional for a query $q = \{a, b, c\}$, and the similarity measure should give $SIM(\{a, b, c\}, \{a, b\}) \leq SIM(\{a, b\}, \{a, b\})$. Using both inequalities we obtain $SIM(\{a, b, c\}, \{a, b\}) < SIM(\{a, b\}, \{a, b, c\})$ i.e. symmetry does not hold.

As already mentioned, cases and queries are not symmetrically used. The nearest neighbor relation $NN_C(u, c)$ is defined on $U \times C$ and not on $U \times U$ (which would lead to useless results). In the same way it could be reasonable to define similarity as function

$$SIM : U \times C \to S. \tag{18}$$

instead of $SIM : U \times U \to S$ as in equation (3).

Transitivity, Triangle Inequality Similarity relations R_{SIM} are in general not transitive (as in daily life). In the example (2) we have $R_{SIM}([1, 1, 1], [1, 1, 0])$ and $R_{SIM}[1, 1, 0], [1, 0, 0])$, but not $R_{SIM}([1, 1, 1], [1, 0, 0])$.

As an extension of transitivity we might look for some kind of triangle inequality for similarity measures. The direct translation for a concrete inverting functions f would mean

$$f(SIM(u, w)) \leq f(SIM(u, v)) + f(SIM(v, w)). \tag{19}$$

This shows that the triangle inequality is not easy to express if all inverting functions f have to be regarded. For the concrete function $f(x) = 1 - x$ we simply get $SIM(u,v) + SIM(v,w) \leq 1 + SIM(u,w)$. As another candidate we might consider the inverted triangle inequality of distances:

$$SIM(u,w) \geq SIM(u,v) + SIM(v,w). \tag{20}$$

Together with reflexivity and symmetry, we obtain

$$1 = SIM(u,u) \geq SIM(u,v) + SIM(v,u) = 2SIM(u,v) \tag{21}$$

and therewith $SIM(u,v) \leq 0.5$ for arbitrary u, v. On the other hand,

$$SIM(u,w) \leq SIM(u,v) + SIM(v,w) \tag{22}$$

makes problems, too. It would lead to $SIM(u,v) \geq 0.5$ for arbitrary u, v.

As a concrete illustration we consider

$$SIM([x_1,x_2],[y_1,y_2]) := Min(\{\ 1\ ,\ card(\{\ i \mid x_i = y_i\}\ \}. \tag{23}$$

There we have

$$SIM([0,0],[0,1]) = SIM([0,1],[1,1]) = SIM([0,0],[0,0]) = 1$$

$$SIM([0,0],[1,1]) = SIM([1,1],[0,0]) = 0$$

and hence

$$SIM([0,0],[0,1]) + SIM([0,1],[1,1]) > SIM([0,0],[1,1])$$

$$SIM([0,0],[1,1]) + SIM([1,1],[0,0]) < SIM([0,0],[0,0]).$$

If we replace SIM by $DIST$ using an inverting function f with $f(0) = 1$ and $f(1) = 0$, then we have especially

$$DIST([0,0],[0,1]) + DIST([0,1],[1,1]) < DIST([0,0],[1,1]).$$

That means that there are intuitive distance measures where the triangle inequality does not hold. It turns out that metrics may be too restrictive and cannot be used for implementing nearest neighbor concepts in certain cases. (Remark: We could slightly modify the example such that it satisfies the "strong" reflexivity as in (15), but still misses the triangle inequality).

We feel that this observation is important, since many distance based approaches do have in mind a metric. On the other hand, it is always possible to define a relationally compatible metric to a given similarity function $SIM : U \times U \to [0,1]$ by:

$$DIST(u,v) := \begin{cases} 0 & , \quad if \quad u = v \\ \frac{1}{2} \cdot (2 - SIM(u,v)) & , \quad otherwise. \end{cases} \tag{24}$$

There we have always $\frac{1}{2} \leq DIST(u,v) \leq 1$, and hence the triangle inequality must be satisfied.

5 Composite Measures

5.1 The Local-Global Principle

For this section we restrict ourselves to the description of cases and queries to real valued vectors $u \in U = \mathcal{R}^n$. Each component of a vector corresponds to some attribute A_i. Now we look for appropriate connections between the level of the attributes and the level of the objects.

Definition 2 (Local-Global Principle).
The Local-Global Principle for similarities (and distances) is formulated in the following way:

There are similarity measures sim_i on the domains of the attributes A_i (on the reals in our case) and some composition function $COMP : \mathcal{R}^n \to \mathcal{R}$ such that

$$SIM([q_1, ..., q_n], [u_1, ..., u_n]) = COMP(sim_1(q_1, u_1), ..., sim_n(q_n, u_n)). \quad (25)$$

The measures sim_i are called local measures and SIM is the global measure. A similarity measure SIM is called composite if it can be combined from some local similarity measures as in (25).

Weighted Hamming measures, i.e. weighted linear functions *COMP* are popular measures for similarities and distances:

$$SIM([q_1, ..., q_n], [u_1, ..., u_n]) = \sum_{i=1,...,n} w_i \cdot sim_i(q_i, u_i). \quad (26)$$

5.2 Local Similarity

Local similarity measures $sim_i : \mathcal{R} \times \mathcal{R} \to \mathcal{R}$ determine the similarity $sim(q_i, c_i)$ for the values of a query $q = (q_1, \ldots, q_n)$ and a case $c = (c_1, \ldots, c_n)$ at the Attribute A_i. Local similarity should express acceptance of a user for different values of this Attribute. When asking for "July 1st" we mostly accept cases which are near to this date. Vague queries may ask for a date "during summer", or "during vacation" etc.

We can interpret $sim_i(q_i, x)$ for a fixed query q_i as the characteristic function of a linguistic term *"about July 1st" "during summer"*, or *"during vacation"*, respectively (cf. Figure 1).

Vague notions in queries can be matched with concrete values in the cases. We could also compare concrete values in a query with related vague notions in cases, or even vague notions in both the query and the cases, respectively. In the latter situation, the evaluation of matching could follow operations from fuzzy theory, cf. [6].

Characteristic functions of linguistic terms usually have non-zero values only in a limited region of their domain. In contrast, similarity measures derived from distances, as for example by

$$sim(x, y) := \frac{1}{1 + dist(x, y)}, \quad (27)$$

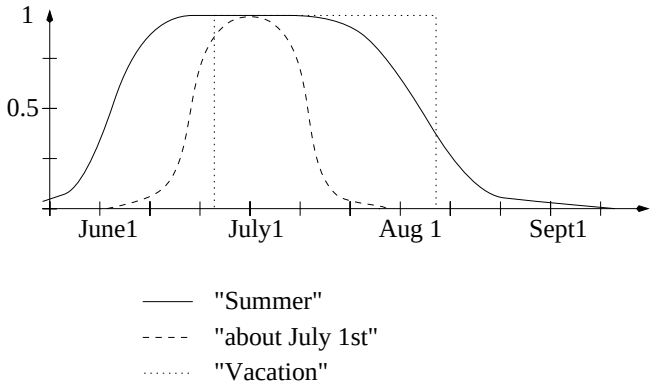


Fig. 1. Local similarity functions (linguistic terms)

may result in non-zero values even for very great differences. Thus, “10” is similar to “1 Million” to a very small, but non-zero amount. This may cause unnecessary effort during retrieval. It is better to ignore such small values.

To express unacceptability we could even use negative ”similarity” values (it would in fact be better to talk about ”acceptance” values as in [5]). An example is given by Figure 2. Using weighted sums as in (26), a negative local value $sim_i(q_i, c_i)$ decreases the global acceptance of a case c for the query q. The negative value for some attribute might be as large that they cannot be compensated by the local similarity values of the other attributes.

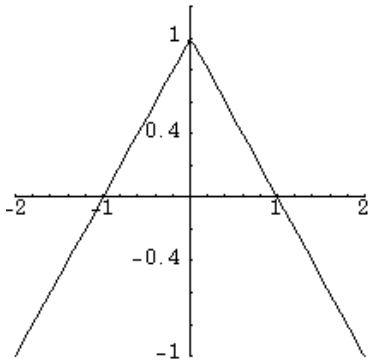


Fig. 2. Local similarity with negative values

5.3 The Global Monotonicity Axiom

The Local-Global Principle connects the representation of queries and cases by vectors with similarity measures. This gives rise to a new axiom for composite measures which we call the *Global Monotonicity Axiom*:

$$SIM(u,v) > SIM(u,w) \rightarrow \exists i \in \{1,...,n\} : sim_i(u_i,v_i) > sim_i(u_i,w_i). \quad (28)$$

The axiom states a necessary condition: A higher global similarity must be supported by at least one higher local similarity.

There are situations where the monotonicity axiom does not hold. As an example we adopt the XOR-problem known from neural nets: We consider two boolean-valued attributes and therefore $U = \{[0,0],[0,1],[1,0],[1,1]\}$. The XOR-problem is represented by $SIM([u_1,u_2],[v_1,v_2]) = 1$ iff $XOR(u_1,u_2) = XOR(v_1,v_2)$, i.e.

$$\begin{aligned} SIM([0,0],[1,1]) &= SIM([0,1],[1,0]) = 1 \\ SIM([0,0],[0,1]) &= SIM([0,0],[1,0]) \\ = SIM([1,1],[1,0]) &= SIM([1,1],[0,1]) = 0. \end{aligned}$$

We have $SIM([0,0],[1,1]) > SIM([0,0],[0,1]$. The Global Monotonicity Axiom ist not satisfied since we have for the local similarity values: $sim_1(0,1) \leq sim_1(0,0) = 1$ (reflexivity) and $sim_2(0,1) = sim_2(0,1)$.

A new attribute (in this case XOR(u,v)) would be needed in order to represent the XOR-problem by monotonous global similarities, i.e. an extension of the language is necessary. It is an important issue to find attributes such that the monotonicity axiom holds. For practical reasons, such new attributes must be easy to compute which limits the search for useful attributes. In particular, it is usually impossible to find an attribute which provides directly the solution. It is, however, fair to say that the representation is not good if the monotonicity axiom does not hold.

There is also a *Local Monotonicity Axiom* which demands decreasing distances/increasing similarities with decreasing absolute differences $|u_i - v_i|$. This axiom is intuitive in a lot of cases, but we could also think of a similarity relation over the reals, where especially the natural numbers are similar to each other:

$$sim(x,y) := \begin{cases} 1 & , \quad if \quad x = y \quad or \quad x,y \in \mathcal{N} \\ 0 & , \quad otherwise. \end{cases} \quad (29)$$

5.4 Transformations for Composite Measures

Relationally equivalent transformations – cf. (13) – can be considered on the local level as well as on the global level.

We have already used the function $\frac{1}{1+x}$ for a transformation on the local level of single attributes in example (5). Thereby, $dist_i(x_i,y_i) = |x_i - y_i|$ are local distances, while $sim_i(x_i,y_i) = \frac{1}{1+|x_i-y_i|}$ are relationally equivalent local similarity measures. They are composed in (5) to the global similarity measure *SIM* using

a (normalized) sum. We obtain a completely different similarity measure SIM' using the transformation by the same function $\frac{1}{1+x}$ on the global level:

$$SIM'([x_1,x_2,x_3],[y_1,y_2,y_3]) := \frac{1}{1+DIST([x_1,x_2,x_3],[y_1,y_2,y_3])} \tag{30}$$

i.e.,

$$SIM'([x_1,x_2,x_3],[y_1,y_2,y_3]) = \frac{1}{1+\sum_{i=1,2,3}|x_i-y_i|}. \tag{31}$$

The example shows that composition and transformation in general do not commute. This is another hint for the fact, that switching between similarity and distances is not as easy as often supposed. Actually, there are even more basic differences as shown below.

5.5 Similarity and Compromises

To be acceptable, it is often sufficient that a case is acceptable for *some*, but not necessarily for all features. We may accept for example

- an expensive offer for more satisfying properties, or
- the color "red" instead of "blue" because the red car is cheaper, or
- a house in a bad state because of its wonderful garden.

Sometimes, offers are formulated in a related style; "we are looking for people who are qualified in at least one of the following disciplines...".

A case $c = (c_1,\ldots,c_n)$ is acceptable for a query $q = (q_1,\ldots,q_n)$ if $SIM(q,c)$ is high. This can be achieved in a weighted sum as in (26) even if $sim_i(q_i,c_i)$ is low for some attributes A_i. This is useful if there is no case with high local similarities for all attributes, i.e. if some compromise between the wanted features and the reality is necessary.

The situation is different for distances: A case c is acceptable for a query q if $DIST(q,c)$ is low. In a weighted sum, a single attribute A_i with a large local distance $dist_i(q,c)$ will give $DIST(q,c)$ a large value. Sometimes it is argued that the problem can be solved using different weights in a weighted sum. It is, in fact, possible to give some features a low priority by low weights. However, these priorities must be fixed *in advance*, while compromises are a matter of relaxing certain features *after* the presentation of cases. We do not know in advance that, e.g., the color is the feature which corresponds to the cheaper offer. Thus we cannot say that the color is an unimportant feature in general – but that is what a low weight in a weighted sum means. We would surely not agree to consider the price as unimportant in general, but we might, in some circumstances, accept a higher price after knowing all alternatives.

The important difference between global similarity and global distance with respect to compromises is illustrated in Figure 3 and 4. We consider the (unnormalized) combination of two identical local measures to a global one by addition.

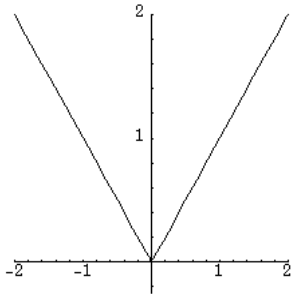
As local distances we consider in Figure 3 the simple distance and its square (such that the composition will be related to the Euclidean distance):

(a) $dist_i(q_i, c_i) = |q_i - c_i|$
(b) $dist_i(q_i, c_i) = (q_i - c_i)^2$.

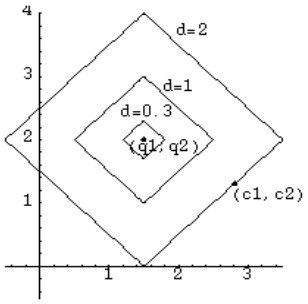
The left side presents the functions for $q_i = 0$, i.e., $dist_i(0, x)$. The corresponding right hand parts of the figure show some "characteristics" of the resulting two-dimensional global distance functions.

(a) $DIST(q, c) = |q_1 - c_1| + |q_2 - c_2|$
(b) $DIST(q, c) = (q_1 - c_1)^2 + (q_2 - c_2)^2$.

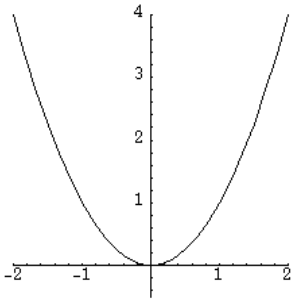
A characteristic (an equi-distance curve) is the line of all cases $c = (c_1, c_2)$ with fixed distance d to a given query $q = (q_1, q_2)$. The characteristics have the form of cubes for (a) (Manhattan distance), and the form of spheres for (b) (similar to the Euclidean distance).



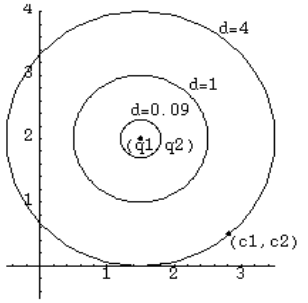
(a) $f(x) = |x|$



$|q_1 - c_1| + |q_2 - c_2| = d$

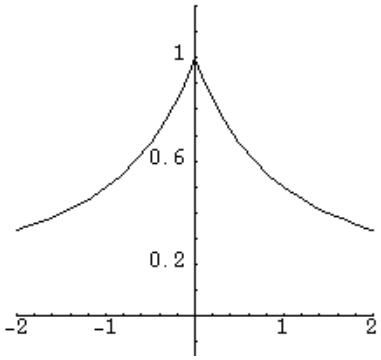


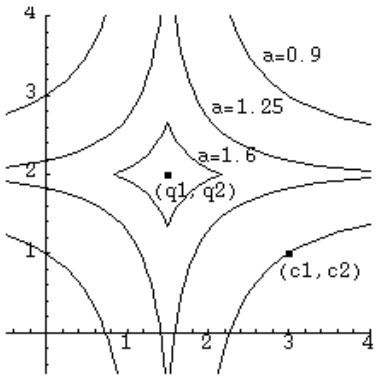
(b) $f(x) = x^2$



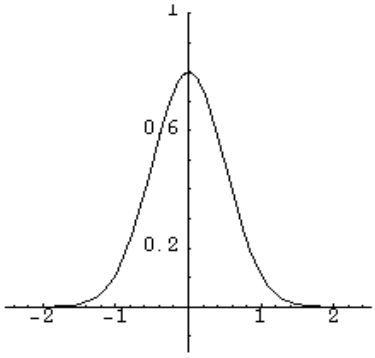
$(q_1 - c_1)^2 + (q_2 - c_2)^2 = d$

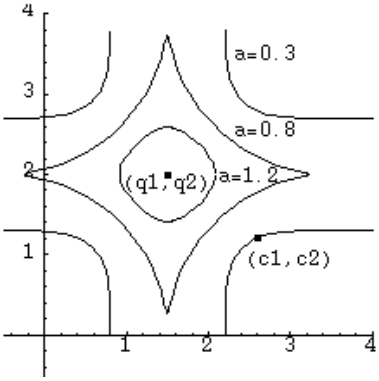
Fig. 3. Characteristics of Composite Distances



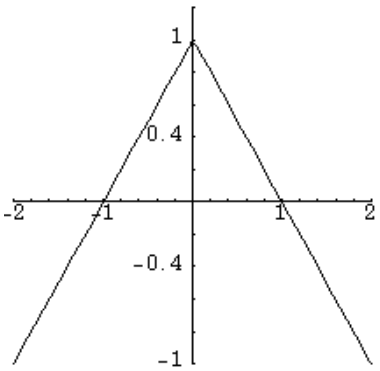


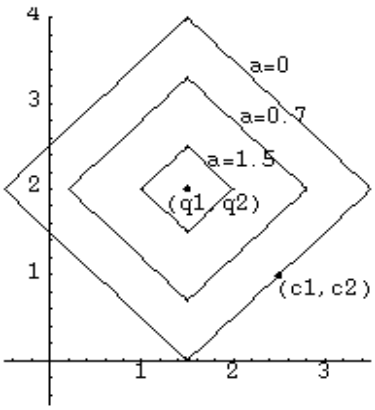
(a) $f(x) = 1/(1+|x|)$ $1/(1+|q_1-c_1|)+1/(1+|q_2-c_2|)=a$





(b) $f(x) = Gauss(x,0,0.5)$ $Gauss(q_1-c_1,0,0.5)+Gauss(q_2-c_2,0,0.5)=a$





(c) $f(x) = 1-|x|$ $(1-|q_1-c_1|)+(1-|q_2-c_2|)=a$

Fig. 4. Characteristics of Composite Similarity Functions

As local similarity functions in Figure 4 we consider an "inverse" of the distance (a) in Figure 3, a Gaussian, and a similarity measure with negative values:

(a) $sim_i(q_i, c_i) = 1/(1 + |q_i - c_i|)$
(b) $sim_i(q_i, c_i) = Gauss(|q_i - c_i|)$
(c) $sim_i(q_i, c_i) = 1 - |q_i - c_i|$.

The left side presents the functions for $q_i = 0$, i.e., $sim_i(0, x)$. The corresponding right hand parts of the figure show some characteristics of the resulting two-dimensional global similarity functions. Global similarity functions in Figure 4 are

(a) $SIM(q, c) = 1/(1 + |q_1 - c_1|) + 1/(1 + |q_2 - c_2|$
(b) $SIM(q, c) = Gauss(|q_1 - c_1|) + Gauss(|q_2 - c_2|)$
(c) $SIM(q, c) = (1 - |q_i - c_i|) + (1 - |q_2 - c_2|)$.

All cases $c = (c_1, c_2)$ on a characteristic (an equi-similarity curve) have the same similarity value a for a given query $q = (q_1, q_2)$.

The characteristics for the distances are closed. This means that there is only limited potential for compromises. A large distance of one attribute cannot be compensated by the other attribute.

In contrast, some characteristics for the similarity measures are open for the lower levels of similarity. This gives potential for compromises: If one attribute has a low similarity value, then it might be compensated by a high value of the other attribute. A high local similarity value for one attribute of the functions (a) and (b) is sufficient to reach a moderate global similarity value independent of the other attribute.

The situation changes if unlimited negative similarity values are allowed as for function (c). Here we have a situation with a similar picture as in Figure 3 (a) with only closed characteristics. Vice versa, we can say that distances correspond to similarity measures which allow negative values.

The difference has consequences for implementations: Closed characteristics allow for pruning strategies. They can be realized as a top down approach. The whole space U of potential cases is divided into different regions of cases which are near to each other. A new query is classified according to the region it belongs to. This can be implemented by a decision tree, and the resulting leaf of that tree can point to the matching cases in the related region. A related compilation process can optimize the regions and the search within a concrete case memory.

The boundaries of the regions (the test criteria in the decision tree) specify values which serve for excluding certain cases. Problems may occur if a query is near to that boundaries. In these situations, the necessary value for pruning is not really reached, and it may be necessary to search in several sub-trees. Unknown values are also critical. Without any hypothesis about the underlying value, all related sub-trees have to be considered. Detailed discussions of this approach including implementation issues can be found in [10].

Open characteristics need another approach. A related bottom up approach is discussed in the next section.

6 Case Retrieval Nets

6.1 Basic Ideas

Information Entities (IEs) have been introduced in Section 2.3 as atomic parts of cases. They are used as indexes, and we consider cases and queries as sets of Information Entities. They may represent any basic knowledge item, such as a particular attribute-value-pair, a key word, or some more complicated structure. They are defined under the view point of *relevant parts of cases to be used in reminding (retrieval).*

Case Retrieval Nets (CRN) combine some ideas of Semantic Networks, Spreading Activation and Neural Networks. They have been first introduced in [2] and [3]. A CRN describes a case base for purposes of retrieval as a net with nodes for the IEs and additional nodes ("case descriptors") denoting the particular cases. The case descriptors may point to additional information (e.g. to a complete textual description of an episode).

IEs are integrated into the CRN as far as they appear in the known cases, or in expected queries, respectively. IE nodes may be connected by *similarity arcs*, and a case node is reachable from its constituting IE nodes via *relevance arcs*. Given this structure, case retrieval is performed by

- *activating* the IEs given by the query,
- *propagating* this activation according to similarity through the net of IEs, and
- *collecting* the achieved activation of the IEs to the case nodes they belong to.

Different degrees of similarity and relevance may be expressed by arcs weights.

The idea is illustrated for the Travel Agency domain in Figure 5: A case is a special travel offer, denoted by a case descriptor, e.g. `<Offer 20219>`. It consists of a set of corresponding IEs giving the specification of that offer, in case of `<Offer 20219>` the IE nodes `<Type:Swimming>`, `<Price:980,->`, `<Place:Matala>`, `<Region:Crete>`, `<Distance to beach:500 m>` are connected with that case node. Asking for an offer in region Crete for swimming and not to far from the beach, the IE nodes `<Type:Swimming>`, `<Distance to beach:200 m>` and `<Region:Crete>` are initially activated. By similarity, the IE nodes `<Region:Malta>` and `<Distance to beach:500 m>` will be activated in the next step, but the amount of activation may depend on arc weights. Finally, the three offers `<Offer 20024>`, `<Offer 20219>`, `<Offer 500122>` will each get some activation. These final case node activations are computed from the incoming activations of IE nodes, which again may be weighted according to the relevance of an IE for case selection. The highest activated cases are proposed to the customer. Here the conflict arises whether the customer accepts a greater distance to the beach for being in Crete or if she changes to Malta. Special preferences may be expressed by initial weights, similarity weights and relevance weights, respectively. A first list of proposals might include both alternatives. Then, if the

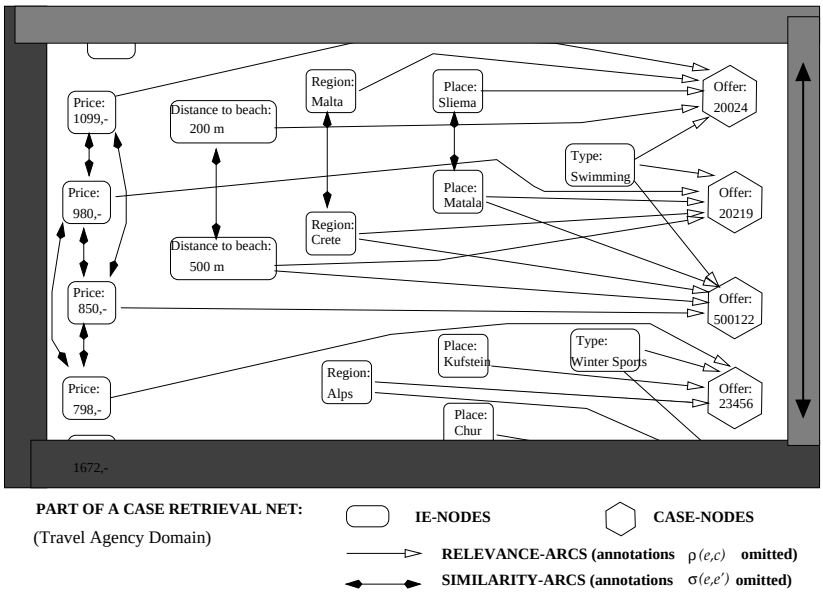


Fig. 5. Example of a CRN in the TRAVEL AGENCY domain.

customer decides for Crete, an appropriate tuning of net parameters can prune other offers in the ongoing process of finding an suitable travel offer.

The example illustrates some features of the retrieval mechanism:

- It can handle *partially specified queries* without loss of efficiency.
- Case retrieval supports *Case Completion*. For any part of a case given as a query, the retrieval algorithm can deliver the remaining part and thus complete the case.
- Insertion of new cases (even with new attributes) can be performed incrementally by injecting related nodes and arcs.

6.2 Basic Case Retrieval Nets

A formal description of Case Retrieval Net in a basic version is given in the following. It can serve as a base for more extended models, and it allows for a detailed investigation of the approach. We recall earlier definitions:

- An *Information Entity* (IE) is an atomic knowledge item in the domain, i.e. an IE represents the lowest granularity of knowledge representation, such as a particular attribute-value-pair. IEs are used as indexes.
- A *case* is a set of IEs. It is denoted by a unique case descriptor.
- A *query* is a set of IEs.

A CRN has links between IE nodes for similarity, and between IE nodes and case nodes for membership of IEs to cases. The links are weighted according to similarity and relevance.

Definition 3.
A Basic Case Retrieval Net (BCRN) is given by $N = [E, C, \sigma, \rho, \Pi]$ *with*

E is the finite set of Information Entities (IE) ("IE nodes"),
C is the finite set of case descriptors ("case nodes"),
σ is the similarity function

$$\sigma : E \times E \to \mathcal{R} \tag{32}$$

which describes the similarity $\sigma(e', e'')$ between IEs e', e'',
ρ is the relevance function

$$\rho : E \times C \to \mathcal{R} \tag{33}$$

which describes the relevance $\rho(e, c)$ of the IE e for the case c.
Π is the set of propagation functions π_n for each node $n \in E \cup C$ with

$$\pi_n : \mathcal{R}^E \to \mathcal{R}. \tag{34}$$

The graphical description (cf. Figure 5) is given by a graph with nodes $E \cup C$ and directed arcs between them. The arc from $e' \in E$ to $e'' \in E$ is labeled by $\sigma(e', e'')$, the arc from $e \in E$ to $c \in C$ is labeled by $\rho(e, c)$. Arcs are omitted if they are labeled by zero. The functions π_n are annotations to the nodes n.

An IE e belongs to a case c if $\rho(e, c) \neq 0$. Its relevance for a case c is given by the value of $\rho(e, c)$ expressing the importance for remembering the case c if the IE e is in the actual "scope of attention". Similarity between IEs e', e'' is measured by $\sigma(e', e'')$. The functions π_n are used to compute the new activation of node n depending on the incoming activations (a simple setting may use the sum of inputs as π_n, – see below).

The *"state"* of a BCRN is defined by the activation of the nodes:

Definition 4.
An activation of a BCRN $N = [E, C, \sigma, \rho, \Pi]$ *is a function*

$$\alpha : E \cup C \to \mathcal{R}. \tag{35}$$

In the graphical notation, an activation α can be represented as a (temporary) annotation to the nodes $n \in E \cup C$. Informally, the activation α describes a *scope of attention*: All activated IEs e ($\alpha(e) \neq 0$) play a role in the process of reminding. The value $\alpha(e)$ of the IE e expresses the importance of that IE for the actual problem. The influence of an IE to the result of the case retrieval depends on its actual importance $\alpha(e)$ and its relevance $\rho(e, c)$ for the cases c (where π_c might express further preferences). Negative values can be used as an indicator for the rejection of cases containing that IE.

6.3 Propagation of Activations

The dynamics of the retrieval process are described as *"state transitions"*, i.e. as changing activations. Formally, the propagation process for the basic model is given by the next definition.

Definition 5.
Consider a BCRN $N = [E, C, \sigma, \rho, \Pi]$ *with* $E = \{e_1, ..., e_s\}$ *and let be* $\alpha_t : E \cup C \to \mathcal{R}$ *the activation at time* t.
The activation of IE nodes $e \in E$ *at time* $t + 1$ *is given by*

$$\alpha_{t+1}(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_t(e_1), ..., \sigma(e_s, e) \cdot \alpha_t(e_s)), \tag{36}$$

and the activation of case nodes $c \in C$ *at time* $t + 1$ *is given by*

$$\alpha_{t+1}(c) = \pi_c(\rho(e_1, c) \cdot \alpha_t(e_1), ..., \rho(e_s, c) \cdot \alpha_t(e_s)). \tag{37}$$

Given an initial activation α_0, then by Definition 5 it is well-defined how the activation α of each node $n \in C \cup E$ has to be computed at any later time point. If we regard similarity and relevance, then the earliest time point to consider the ranking of cases (according to their resulting activations) is after two propagation steps. Hence for the basic model, the propagation process is a three-step process:

Step 1 – Query :
According to the query, the *primary scope of attention* is given by α_0 which is determined for all IE nodes as follows:

$$\alpha_{query}(e) = \begin{cases} 1 & : \quad \text{for IE nodes } e \text{ of the new problem} \\ 0 & : \quad else \end{cases} \tag{38}$$

For more subtle queries, α_0 might assign different weights to special IE nodes, and some *context* may be set as an initial activation for further nodes.

Step 2 – Similarity propagation between IE nodes :
The activation α_0 is propagated to all IE nodes $e \in E$ leading to an *extended scope of attention*:

$$\alpha_1(e) = \pi_e(\sigma(e_1, e) \cdot \alpha_0(e_1), ..., \sigma(e_s, e) \cdot \alpha_0(e_s)), \tag{39}$$

Step 3 – Relevance propagation from IE nodes to case nodes :
The result of step 2 is propagated to the case nodes $c \in C$:

$$\alpha_2(c) = \pi_c(\rho(e_1, c) \cdot \alpha_1(e_1), ..., \rho(e_s, c) \cdot \alpha_1(e_s)). \tag{40}$$

In this computation model all similarity computation has to be performed in Step 2 in a direct manner between adjacent IE nodes.

More subtle computations (including e.g. concepts, micro features, rules) may use *several steps* (e.g. from a "simple IE" to a more general "concept IE", then to another "concept IE" by "firing a rule", and back again to certain "simple IEs", which then leads to the activation of cases – cf. [4])). All these steps can be implemented as activation passing, but some care is necessary to prevent from unwanted effects (circular activation, asynchronous propagation etc.). On the other hand, computations over longer time intervals may be considered as "any time" computations: If the first activations arriving at the case nodes are

sufficient to solve the problem, then the process of reminding may stop. Otherwise, some reminder from more far regions may come into account by further activation steps.

For the 3-Step activation process we obtain the final activation $\alpha_N(c)$ at the case nodes c by combining the formulae from above:

$$\begin{aligned}\alpha_N(c) = \pi_c(\ & \rho(e_1,c)\cdot\pi_{e_1}(\sigma(e_1,e_1)\cdot\alpha_{query}(e_1),...,\sigma(e_s,e_1)\cdot\alpha_{query}(e_s)\ ) \\ & ,... , \\ & \rho(e_s,c)\cdot\pi_{e_s}(\sigma(e_1,e_s)\cdot\alpha_{query}(e_1),...,\sigma(e_s,e_s)\cdot\alpha_{query}(e_s)\ )).\end{aligned} \tag{41}$$

Thus we can summarize the concept of case retrieval in the basic model of CRN:

> Consider a BCRN $N = [E, C, \sigma, \rho, \Pi]$ and the activation functions α_t as defined above. The result of the case retrieval for a given query activation α_0 is the *preference ordering* of cases according to decreasing activations $\alpha_N(c)$ of case nodes $c \in C$.

It is often useful to use the Maximum-function as propagation function π_e at the IE nodes, and the sum as propagation function π_c at the case nodes. Then we obtain for the steps from above:

$$\begin{aligned}&\text{Step 2:} \quad \alpha(e) = MAX_{e'}\ \sigma(e',e)\cdot\alpha_{query}(e') \\ &\text{Step 3:} \quad \alpha(c) = \sum_e \rho(e,c)\cdot\alpha(e)\end{aligned}$$

and the resulting activation in the case nodes for a query α_{query} is given by

$$\alpha_N(c) = \sum_e \rho(e,c)\cdot MAX_{e'}\ \sigma(e',e)\cdot\alpha_{query}(e'). \tag{42}$$

6.4 Implementing Composite Functions by CRNs

Composite similarity measures (and distances in a related way) can be easily implemented by BCRNs. According to the Local-Global Principle, a similarity measure *SIM* is called *composite* (cf. Definition 2) if it can be combined from some local similarity measures sim_i and some *composition function COMP* : $\mathcal{R}^n \to \mathcal{R}$ such that

$$SIM([q_1,...,q_n],[u_1,...,u_n]) = COMP(sim_1(q_1,u_1),...,sim_n(q_n,u_n)). \tag{43}$$

Cases and queries are thereby n-dimensional vectors according to attributes $A_1,...,A_n$. IEs have the form of attribute value pairs: $e = [A_i, value]$, and the set E of IEs is partitioned according to the attributes the IEs belong to.

In principle, a case node $c = ([A_1, value_1],...,[A_n, value_n])$ has a link by a relevance arc from the IE node $[A_i, value]$ for every attribute A_i. Hence the composition function $COMP$ can be implemented directly by the propagation function π_c at the case node c.

For example, weighted Hamming measures – cf. (26) –

$$SIM([q_1, ..., q_n], [u_1, ..., u_n]) = \sum_{i=1,...,n} w_i \cdot sim_i(q_i, u_i) \tag{44}$$

can be implemented by related arcs weights $\rho([A_i, value], c) = w_i$ and $COMP = \sum_{i=1,...,n}$. A closer look to BCRNs show, that the weights can even have individual values for each relevance link.

It is not necessary that queries and cases really do have specified values for each attribute A_i. If a query has unspecified (unknown) values, the initial activation α_{query} will place non-zero activations only for the attributes with known values. If a case has an unspecified value for some attributes, then related relevance links are missing. CRNs can easily handle partially specified queries and cases.

At a first view, a "pessimistic" approach is realized: Attributes with missing values do not contribute to the activation. But using more sophisticated propagation functions π, the effects can be changed, e.g. by some normalization.

6.5 Use of CRNs for Case Completion

The ranking of cases can be exploited in different ways. Some further steps may also use the derived state of the CRN.

We call the highest activated cases (according to α_N) the *answer cases* of the retrieval. Usually, not all IE nodes e of the answer cases have already been in the extended scope of attention after the similarity activation step ($\alpha_1(e) = 0$, computed in Step 2). Such an IE node e was a constituent part of an answer case c, but it was not addressed by the query: That means it was not known for the actual problem. Hence it is a candidate for the completion of the actual case, and we call such IEs the *completion candidates*. A completion candidate could be a proposal for the solution of the actual problem.

It is crucial here to remind that an IE may be any relevant index for a case, especially the "solution parts" of the cases can be integrated as IEs into a CRN (then they could be used in a query, too).

As far as all answer cases provide similar completion candidates of a certain aspect, we have a high evidence that the actual problem would behave in a similar way for that aspect. On the other hand, if there are dissimilar completion candidates, then our actual problem needs further investigation. If for example there are different solutions possible in a diagnosis problem, then we have to test further symptoms. By comparing the answer cases with different solutions, we may find the symptoms which are correlated to the different diagnostics. These symptoms can be found by further propagation steps back from the answer cases to the IEs in the CRN (cf. [7]).

But not all applications must try to find a single solution. In the travel agency example the assistance system should even better make a set of different proposals (which even should be dissimilar to some extend) and let the final choice to the costumer.

7 Conclusions

Case Completion leads to an extension of cases from rule-like style to a constraint oriented model. It can be supported by Case Retrieval Nets. There exist various successful implementations of CRNs ranging from electronic catalogues to document retrieval [11]. CRNs are interesting as some kind of associative memory from technological view point, and from cognitive view point, respectively. There is ongoing work to use CBR techniques for behavior control of autonomous systems in dynamic environments [12], and for repeated negotiations in a sociological environment [13].

Acknowledgements The author wants to thank the colleagues from the AI group of Humboldt University, especially Mirjam Minor, and the colleagues of Empolis, especially Stefan Wess and Mario Lenz, for inspirations and helpful discussions over many years.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI communications, 7(1)1994, 39–59.
2. Burkhard, H.D.: Case retrieval nets. Techn. report. Humboldt University, Berlin, 1995.
3. Burkhard, H.D., Lenz, M.: Case retrieval nets: Basic ideas and extensions. In H.D. Burkhard, M. Lenz (eds.): 4th German Workshop on CBR 1996, Humboldt University Berlin 1996. 103–110.
4. Burkhard, H.D.: Cases, Information, and Agents. In P. Kandzia, M. Klusch (eds.): Cooperative Information Agents. Proc. First Int. Workshop CIA'97. LNAI 1202, Springer-Verlag, 1997, 64–79.
5. Burkhard, H.D.: Extending some Concepts of CBR – Foundations of Case Retrieval Nets. In: M.Lenz, B.Bartsch-Spörl, H.D.Burkhard, S.Wess (eds.): Case-Based Reasoning Technology: From Foundations to Applications. Lecture Notes in Artificial Intelligence 1400. Springer-Verlag 1998, 17–50.
6. Burkhard, H.D., Richter, M.M.: On the Notion of Similarity in Case Based Reasoning and Fuzzy Theory In: Pal, S.K., Dillon, T.S, Yeung, D.S. (eds.): Soft Computing in Case Based Reasoning. Springer-Verlag 2000, 29–46.
7. Lenz, M., Burkhard, H.D., Brückner, S.: Applying Case Retrieval Nets to Diagnostic Tasks in Technical Domains. In: I. Smith, B. Faltings (eds.): Advances in Case-Based Reasoning. Proc. of the Third European Workshop EWCBR-96. Lecture Notes in Artificial Intelligence 1168. Springer-Verlag 1996, 219–233.
8. Roth-Berghofer, T.: Knowledge Maintenance of Case-Based Reasoning Systems The SIAM Methodology. Akademische Verlagsgesellschaft Aka GmbH, DISKI 262, Berlin. Ph.D. thesis. April 2003
9. R. Schank. Dynamic memory: A theory of learning in computers and people. Cambridge Unv. Press, New York, 1982.
10. Wess, S., Althoff,K.-D., Derwand, G.: Using kd-Trees to Improve the Retrieval Step in Case-Based Reasoning. In: S. Wess, K.-D. Althoff, M.M.Richter (eds.): Topics in Case-Based Reasoning. Proceedings of the first European Workshop on Case-Based Reasoning (EWCBR-93), Lecture Notes in Artificial Intelligence 837. Springer-Verlag 1993, 167–181.

11. http://www.empolis.de/en/index.php
12. Project "Architectures and Learning on the Base of Mental Models" of the Research program 1125 "Cooperating teams of mobile robots in dynamic and competitive environments" granted by the German Research Association (DFG). http://www.ais.fraunhofer.de/dfg-robocup/
13. Project "INKA" (http://www.ki.informatik.hu-berlin.de/inka/) of the Research program "Socionics" granted by the German Research Association (DFG). http://www.tu-harburg.de/tbg/Deutsch/SPP/Start_SPP3eng.htm

Hans-Dieter Burkhard, Prof. Dr. sc.,(born 1944) is the leader of the Artificial Intelligence group in the Institute of Informatics at Humboldt University Berlin. He is Fellow of the ECCAI and Vice President of the International RoboCup Federation. He has worked on Automata Theory, Petri Nets, Distributed Systems, VLSI Diagnosis and Knowledge Based Systems. Current interests include Distributed Artificial Intelligence, Agent Oriented Techniques, Case Based Reasoning, Information Systems, Sozionics, AI-applications in Medicine, and Intelligent Robotics. His group has been world champion in the RoboCup in 1997 (Simulation League) and in 2004 (Four Legged Robots League, German Team).