An Approach Transmutation-Based in Case-Based Reasoning

Thouraya Bouabana-Tebibel[1](✉), Stuart H. Rubin[2], Yasmine Hoadjli[1], and Idriss Benaziez[1]

[1] Ecole nationale Supérieure d'Informatique, LCSI Laboratory, Alger, Algeria
{t_tebibel,y_hoadjli,i_benaziez}@esi.dz
[2] Space and Naval Warfare Systems Center Pacific, San Diego, CA 92152-5001, USA
stuart.rubin@navy.mil

Abstract. Case-Based Reasoning (CBR) interests the scientific community, whom are concerned with scalability in knowledge representation and processing. CBR systems scale far better than rule-based systems. Rule-based systems are limited by the need to know the rules of engagement, which is practically unobtainable. The work presented in this paper pertains to knowledge generalization based on randomization. Inductive knowledge is inferred through transmutation rules. A domain specific approach is properly formalized to deal with the transmutation rules. Randomized knowledge is validated based on the domain user expertise. The approach is illustrated with an example and is seen to be properly implemented and tested.

Keywords: Case-Based Reasoning · Contextual search · Randomization · Transmutation

1 Introduction

The performance of a Case-Based Reasoning (CBR) system is highly correlated with the number of cases that it imbues. This is evidenced by the resolution process which seeks the most similar cases found in the case base to solve new (sub-)problems. Each case represents a situational experience already saved by the system. Solutions related to the most similar problems are retrieved for reuse. They are proposed to the user who may revise them according to the current situation. The more similar the experience is to the encountered case (situation), the more likely an adaptation is to be valid. The case base is enriched by the acquisition of revised new cases.

The size of the knowledge base, as source of the primitive knowledge and as a warehouse to the generated cases, is worthy of consideration. On one hand, the more and the richer base, the more accurate the system is with regard to resolution. On the other hand, too large knowledge base slows down the reasoning process. The main purpose served by this paper is to define how to integrate knowledge generation in a CBR system.

Implicit knowledge can be present in the case base through dependencies and similarities between cases. When dependencies are strong, they allow for knowledge derivation by transformation based on equivalence. When they are weak, knowledge derivation is obtained by randomization based on analogy. In this paper, we define an approach to infer knowledge by materializing the implicit knowledge through newly-generated cases. Knowledge inference will be inductive and based on randomization.

Randomization is a new trend in CBR processing with many pioneering works by Rubin et al. [1–7]. These works show that hybridization of CBR and fuzzy methods leads to compromise solutions with respect to accuracy and computational complexity. Randomization is used for case retrieval in CBR.

However, recent work in Case-Based Reasoning [3, 4] has evidenced that while cases provide excellent representations for the capture and replay of plans, the successful generalization of those plans requires far more contextual knowledge than economically sound planning will allow. The approach to randomization, proposed herein, enables the dynamic extraction of transformative cases, which permits the inference of novel cases through the substitution of problem/solution cases along with a check of case validity. It is based on the work proposed in [6] which proposes an approach to derive knowledge by randomization using analogy among the segments of information. The approach proposed in [8] first extends the previous work with new conditions on the randomization process.

In this paper, we propose to formalize the cases semantics, to better handle randomization and validate the randomized cases. We also show how such a randomization may be applied to a case-based reasoning system, and how the latter may benefit from randomization with respect to the problem and solution parts, which define a given case.

An appropriate application illustrates the approach. It consists of a travelling robot. The robot has as its mission to move between two points with the obligation of passing through a number of checkpoints. Such an application finds utility in many areas, like road transportation and office environments. For the latter area, some tasks of the robot could be: mail delivery, document exchange between offices, guest orientation, etc.

The remainder of the paper is structured as follows. Section 2 presents works related to randomization, especially for case-based reasoning systems. Section 3 presents the proposed transmutation approach. Section 4 develops the technical aspect of the approach integration into a CBR system. Section 5 applies the approach to a specific domain and Sect. 6 validates it. Finally, Sect. 7 illustrates the approach through an appropriate application.

2 Related Work

Case-based reasoning has been useful in a wide variety of applications. Health science is one of its major application areas [9]. In addition, financial planning, decision making in general [10, 11], and the design field [12–14] are representative of important application areas for case-based reasoning.

Indeed, many CBR systems have been developed in the past two decades. While increased interest in the application of CBR is observed, the design and development process of the CBR system itself is still the subject of many studies in the literature. The focus is on the theories, techniques, models, algorithms and the functional capabilities of the CBR approach. For example, in the retrieval phase, authors of [15] argue that how to best design a good matching and retrieval mechanism, for CBR systems, is still a controversial issue. They aim to enhance the predictive performance of CBR and suggest a simultaneous optimization of feature weights, instance selection, and the number of neighbors that can be combined using genetic algorithms (GA). The best method to index cases is still an open issue. Fuzzy indexing is one of the approaches, which that have been successfully applied by previous researchers in different applications, such as presented by the works in [16, 17].

Generalization and fuzzification of similar cases comprises the basis for search improvement and case-base size reduction. Randomization is used for case retrieval in CBR. In [18–20] fuzzy similarity measures are applied to range and retrieve similar historical case(s). In [18], a fuzzy comprehensive evaluation is applied for the similarity measure within the CBR approach proposed for reusing and retrieving design information for shoes. In [21], a k-NN clustering algorithm is proposed.

Fuzzy logic is advantageous in knowledge transfer as shown in [22], where uncertainty and vagueness characterize the target domain. An example of the use of fuzzification can be found in [23]. In this work, a framework for fuzzy transfer learning is proposed for predictive modeling in intelligent environments.

Hybridization of multiple techniques is adopted in many applications. In [24] a hybrid decision model is adopted for predicting the financial activity rate. The hybridization is performed using case-based reasoning augmented by genetic algorithms (GAs) and the fuzzy k nearest neighbor (fuzzy k-NN) methods.

Randomization is also used in the reuse phase of CBR to make a decision on whether or not the most similar case should be reused. In [25], a random function is applied on a special category of users that have made similar requests to compute their mean opinion.

Similarly, in [26], decision making on knowledge handling within a CBR system is based on fuzzy rules – thus providing consistent and systematic quality assurance with improvement in customer satisfaction levels and a reduction in the defect rate. In [27], a scoring model based on fuzzy reasoning over identified cases is developed. It is applied to the context of cross-document identification for document summarization.

3 Transmutation Approach

A methodology for the induction of knowledge based on randomization is proposed in [6]. The methodology introduces an approach to derive knowledge by transmutation using analogy among the base cases. We dedicate, in this paper, the approach to an exclusive use in CBR systems. This application is given in the two following subsections, which respectively deal with the problem and solution transmutation.

In the CBR base, a case is represented by a grammatical production in the form of: *Problem Situation* $\rightarrow$ *Solution Action*; where the left-hand side of the production expresses the problem part, and the right-hand side is the solution part. The case base is defined to be domain- specific. The problem is composed of multiple descriptors {A, B, C…} and the solution is composed of multiple components (X, Y, Z…).

Transmutation on the grammatical production is based on analogies between cases and provides new productions, thus enriching the case base. Two types of transmutation may be considered. The first one consists in a problem substitution; whereas the second one is a substitution of the solution part. Both types of transmutation require three specific cases, which we will call transmutation trio or trio for short.

3.1 Problem Substitution

The first type of transmutation is based on a trio composed of: (1) two cases with the same solution part, but different problem parts; they will be called primary cases; (2) a third case on which the substitution can apply; thus a case whose problem part includes the problem part of one of the primary cases. It is called a substitution case. For example, we consider the following trio:

C1 : $(A, B, C) \rightarrow (X)$

C2 : $(A, E) \rightarrow (X)$

C3 : $(A, B, C, D) \rightarrow (X, V)$

We note that C1 and C2 have the same solution (X). Thus (A, B, C) and (A, E) can substitute for each other in the context of (X), since they have the same solution part. We notice that (A, B, C) is included in the problem part of C3 and (X) in its solution part. From these cases we can derive a new case:

C3′ : $(A, E, D) \rightarrow (X, V)$

3.2 Solution Substitution

The second type of transmutation is based on a trio composed of: (1) two cases, called primary cases, with the same problem part, but different solution parts; (2) a third case, called substitution case, on which the substitution can apply; thus a case whose solution part includes the solution part of one of the primary cases. We consider, for example, the following trio:

$$\mathrm{C3}' : (A, E, D) \rightarrow (X, V)$$

$$\mathrm{C4} : (A, E, D) \rightarrow (U)$$

$$\mathrm{C5} : (A, E, D, F) \rightarrow (U, T)$$

We note that C3′ and C4 are non-deterministic; the same problem (A, E, D) is mapped to distinct solutions. Thus (X, V) and (U) can substitute for each other in the context of (A, E, D), since they have the same problem part. The newly generated case C5′ will be:

$$\mathrm{C5}' : (A, E, D, F) \rightarrow (X, V, T)$$

Note that these transformative substitutions are often, but not always valid. This follows because there can be complex latent contextual interactions. For example, it may be that my stove can burn alcohol and simple contextual transformations tell us that it should also burn kerosene. However, here the latent context is that kerosene is more viscous than alcohol, which serves to interfere with the fuel-injection mechanism. Clearly, such transformations require the application of domain-specific knowledge to insure validity.

4 Knowledge Induction in a CBR System

The process of knowledge generation based on the proposed approach serves to amplify the case base with new knowledge, thus enhancing the capability of search in CBR systems for delivering more potentially similar cases. Such knowledge amplification [2] relies on the built transmutation trios. The more the transmutation trios are well-structured the faster will be their retrieval. For this purpose, we propose a case-based segmentation driven by the trios.

4.1 Knowledge Structuration

To speed up retrieval of the transmutation trios, we structure the case base around the trio. We split the knowledge base into segments, where a segment serves to gather cases forming transmutation trios. Trios that belong to the same segment share, necessarily, at least one common case. Each segment is represented by a delegate. The delegate concerns only the problem part. It is a generic description that best represents problems within the same segment. It also can be defined as a generalization yielding the most significant part of a problem. A case is added to the segment if its problem matches the delegate.

For example, let (C1, C2, C3, C4, C5, C6, C7, C8, C9, C10) be the cases of the base. We suppose that we can form the trios {(C1, C2, C3); (C2, C4, C5); (C6, C7, C8)}. We, thus obtain four segments. The first one comprises the trio (C1, C2, C3) and (C2, C4, C5), thus, the cases C1, C2, C3, C4 and C5. Thus, the cases C6, C7 and C8 will have their own segment as they do not share any case with the other segments. The cases C9 and C10 are not included in any randomization trio; thus, each of them will have its own segment.

In order to perform the partitioning, we use a variant of the k-medoids classification algorithm [28]. The k-medoids is a classical partitioning technique that splits a set of n objects into k clusters, each of which is represented by an element. In the classification we propose, k is initially unknown. The segmentation is as follows (Fig. 1):

- The knowledge base will be partitioned in an undefined number of segments.
- Each segment will have a delegate that is the most representative element in the segment.

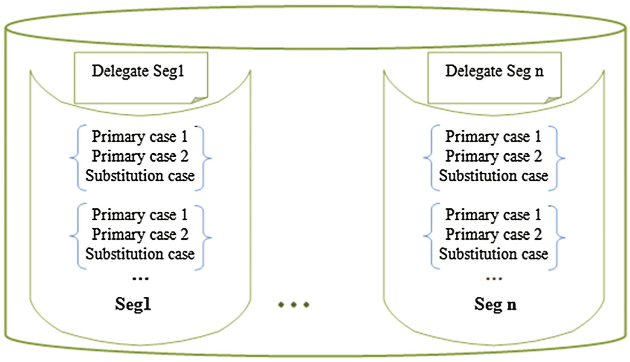


Fig. 1. Knowledge base structure.

- To be included in a segment, a new case is compared with the delegate; if they match, then the case is added; otherwise, it is compared with the other delegates.
- If a case does not match any delegate, it founds a new segment and becomes its delegate.

Figure 1 shows the structure of the knowledge base after segmentation.

We note that the cases are not physically inserted into the segments. They only are indexed. Thus, a case is saved once in the case base; but, it may be indexed several times in a segment.

While structuring the case base into segments, the latter are filled with the transmutation trios. Only trios generating valid cases are saved in the segments. Case validation is domain-specific. It is performed by domain experts. Furthermore, the inducted cases are not stored in the segments. This aims at preventing the explosion of the knowledge base. Thus, the transmutation process will be executed at every problem resolution as an alternative.

The case base segmentation is performed as given by Algorithm 1. No segment exists a priori.

Algorithm 1

```
For each case C
  For each segment S of the case base
    Compare the case C with the delegate of S
    If (comparison is true)
      Add the case to segment and choose
      Browse the segment and for each case C’
        If (C’ is similar to C in either its problem
            or solution part)
              Form a pair of primary cases with C
              And C’ and then save it
        End If
      End For
      Browse the segment a second time and for
        each case C’’
          If (C’’ is a “substitute case” for the
              Primary cases formed above)
                Save C’’ with the associated primary
                cases (C and C’) if a valid trio
          End If
      End For
    End If
  End For
  If no segment can contain C
      Create a new segment S’
      Add C to S’ and define it as its delegate.
  End If
End For
```

4.2 Search for Accurate Knowledge

Figure 2 shows a CBR system integrating the proposed knowledge induction technique, which consists in the trios transmutation.

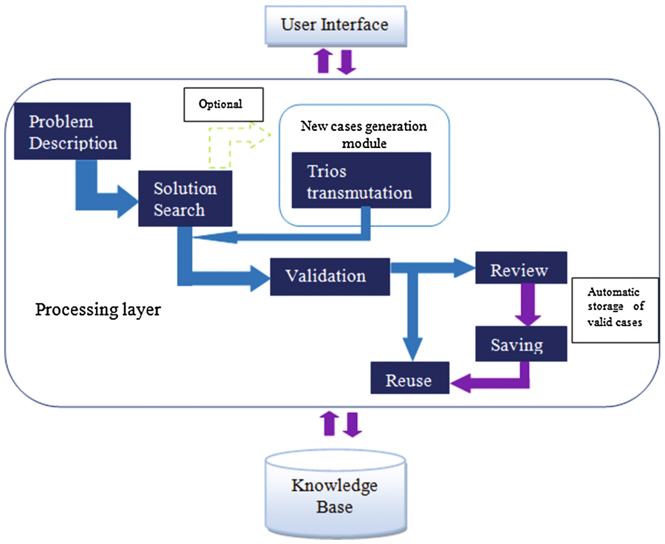


Fig. 2. CBR system architecture integrating knowledge induction

The user interface allows a user to specify the problem part of the searched case. Once the problem resolution is completed, it restores the solution part for the user. It also permits the system to interact with the user by giving him more detailed information about the problem or showing him the problems resolution steps.

The processing layer is based on the classical case-based reasoning process augmented with the knowledge induction process. The case-based reasoning process consists of 3 main phases: (1) description of the problem, (2) search of a solution among the cases relevant to solve the problem, and (3) reuse of a potential solution.

At the search phase, for a given problem resolution, a comparison is first made with the segment delegates. Only the segments whose delegate matches the problem to solve are retained. Next, prior to any search for the most similar case, we first generate more cases by running the transmutation on the trios of the retained segments. This may help finding a solution from the generated cases. It at least improves the chances for obtaining more similar cases than the ones previously entered into the knowledge base. Next, similarity is calculated between the problem to be solved and all cases of the retained segments. Search is given by Algorithm 2.

Algorithm 2

```
For each segment S of the case base
  Compare the case C with the delegate of S
  If (comparison is true)
      Add cases in S to the set SetOfCases
      For each trio T in S
        Generate new cases by transmutation on T
        Add the generated cases to SetOfCases
      End For

  End If
End For
Apply similarity measurement between the
problem of case C and all cases in SetOfCases
```

The reuse phase permits one to build the solution based on the most similar case found in the knowledge base. The more the cases that are similar, the simpler reuse will be, since the modifications will be relatively few. When similarity among the cases is not complete, there is need for adapting the retrieved solution so that it best fits the corresponding problem. Thus, validity of the retrieved solution is checked first. Next, the solution is possibly revised and the new case is saved to be reused in future problem resolutions.

We note that while the inducted cases, resulting from the transmutation process, are not saved in the case base, the revised cases are. Indeed, while the inducted cases can be re-derived, the revised cases will be lost if not stored.

5 Illustration of the Transmutation Approach

In order to best highlight the approach proposed in this paper, an appropriate domain of application is required. The application must have an intuitive solution in order to help to properly evaluate the approach. The degree of similarity between our intuitive knowledge about the results and the results obtained will reflect upon the validity of the approach.

As stated in the introductory section, we will apply our approach to a travelling robot that moves from point A to point B by going through a number of checkpoints. The problem to resolve will be the route described by the user. It consists of the start and end points plus all of the required checkpoints. The solution part must be a route including all checkpoints specified by the user. It is represented by a set of lines: Vertical Line (VL), Horizontal Line (HL) and Diagonal Line (DL).

Thus, each case will be represented as follows:

(Problem Situation) → (Solution Action); where:
Problem Situation: (start point (S), {checkpoints R1, R2, … Rn}, end point (F))
Solution Action: ({HL(S,A), VL (A,B), DL (B,R1), …,})

The segment delegates must be designed to best represent the segment cases. Thus, they should contain the pertinent points allowing for a good compromise between the number of segments and their size. A delegate composed of the start and end points as well as the checkpoints will provide a huge number of segments; whereas a delegate composed of only one point, for example the start point, will generate a large segment size. As a compromise, we suggest to compose the delegate of the start and end points.

In what follows, we note prb_i and sol_i, respectively, the problem and solution parts of Case_i.

5.1 Domain Analysis

Trios construction relies on similarity between the cases, especially those composing pairs of primary cases; whereas similarity depends on the domain we consider, i.e. the problem/solution semantics. An adequate semantics-based similarity function is usually more significant than a syntactic-based one, yielding the sought analogy between cases. A syntactic evaluation, through an integral comparison, may not provide accurate results. Herein, analogy depends on the chosen problem/solution granularity, expressed in terms of points to go through.

More specifically, in the retained domain, the problem part provides the situation, i.e. the points that must be crossed. Given the nature of the application, it means that at least the mentioned points must be visited. This is a hard constraint that must be satisfied by the solution; whereas, the solution action may be released by including more lines. We can intuitively understand that a route contains other points than those specifically indicated. As a result, we can affirm that any route (solution action) including all points mentioned in the problem is a solution for this problem. On the other hand, any situation whose points form concatenated stretches of a route (solution action) may be a problem for this solution. However, in this case, observance of strong sequencing is required.

A substitution case must include, within its problem (resp. solution) part, the problem (resp. solution) part of one case from the primary pair. Inclusion is released with regard to sequencing outside the checkpoints for the problem and solution parts. Points/lines may be added before and/or after the checkpoints, but never inside these latter.

In what follows, we will explore and discuss in detail how a new case is inferred.

5.2 Domain Semantics

Based on the domain analysis developed for the considered application, it follows that a precise semantics need be defined for the problem and solution parts. This semantics must support the necessary elements to express aspects related to route designation and route construction. Routes are traced by rectilinear movements. Hence, they respond to properties handling sequencing actions and/or states. We distinguish, hereafter, between three sequencing types.

Definition 1. Free sequencing
The free sequencing property is defined as a disordered sequencing of the elements of a set. It is expressed by means of the structural operator ";" to separate the elements of the set.

Definition 2. Weak sequencing
The weak sequencing property is defined as an ordered, but not necessarily joint sequencing of the elements of a set. Extra-set elements may be inserted between the elements of the set. This sequencing is expressed by means of the structural operator "|".

Definition 3. Strong sequencing
The strong sequencing property is defined as an ordered and joint sequencing of the elements of a set. None extra-set elements may be inserted between the elements of the set. This sequencing is expressed by means of the structural operator "||".

Definition 4. Sequencing substitution
Free sequencing substitution, weak sequencing substitution and strong sequencing substitution are substitutions where the substitute respectively satisfies free sequencing, weak sequencing, and strong sequencing.

Definition 5. Substitute
The substitute is the set of checkpoints, respectively lines, of a primary case, which replace the set of checkpoints, respectively lines, of the other primary case within a substitution case; e.g., in Sect. 3.1, the substitute is (A, E).

In terms of connection, the operator "||" is stronger than the operator "|" which is stronger than the operator ";".

For example, in S1 = (A; B; C | D | E || F || G), the sequence E||F||G is strongly connected, the sequence C|D|(E||F||G) may be separated by extra-set elements, and the elements A, B and (C|D|(E||F||G)) may appear in any order.

5.3 Transmutation Based on Problem Substitution

Let us consider the following cases:

Case1: (S1 || R1 || R2 || R3 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL(R3, F1))

Case2: (S1 || **R1 | R3** || F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL(R3,F1))

Case1 and Case2 compose a pair of primary cases that can be involved in transmutation by problem substitution as defined in Sect. 3.1. They have the same solution action for two different problem situations.

An appropriate substitution case would be Case3. Its problem situation includes all points specified by Case2. It is structured as follows.

Case3: (S1 || **R1 | R3** || R4 || R5 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL (R3,R4), HL(R4,R5), HL(R5,F1))

Transmutation of the trio Case1, Case2 and Case3 yields the case Case3′.

Case3′: (S1 || R1 || R2 || R3 || R4 || R5 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2, R3), HL(R3,R4), HL(R4,R5), HL(R5,F1))

Now, let us consider the following cases:

Case1: (S1 || **R1** || **R2** || **R3** || F1) → (HL(S1, R1), DL(R1, R2), DL(R2, R3), HL (R3, F1))
Case2′: (S1 | R1 | R3 | F1) → (HL(S1, R1), DL(R1, P), DL(P, R3), HL(R3, F1))

Case1 and Case2′ do present integral similarity neither for the problem nor for the solution parts. However, this does not mean that no similarity exists between the two cases. We notice that sol1 is a solution of prb1, but also a solution of prb2′. It provides a route that crosses all the points declared by prb2′, namely, the start and end points S1 and F1 as well as all the checkpoints (R1 and R3). Thus, prb2′ may be substituted for prb1. However, sol2′ does not resolve prb1. Thus, prb1 cannot be substituted for prb2′.

Thereby, Case1 and Case2′ compose a pair of primary cases, which can be involved in a transmutation by problem substitution. Case3 cannot constitute an appropriate substitution case for that pair of primary cases, since prb1 cannot be substituted for prb2′. Case4 does; its problem situation includes all points specified by prb1. It is structured as follows:

Case4: (S1 || **R1** || **R2** || **R3** || R4 || R5 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2, R3), HL(R3,R4), HL(R4,R5), HL(R5,F1))

Transmutation of this new trio Case1, Case2′ and Case4 yields Case4′.

Case4′: (S1 || R1 | R3 || R4 || R5 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL (R3,R4), HL(R4,R5), HL(R5,F1))

We note that R2 is no more a checkpoint in the generated case, but only a point included in the route.

5.4 Transmutation Based on Solution Substitution

Now, let us suppose that we have the following cases:

Case2: (S1 || R1 | R3 || F1) → (HL(S1, R1), **DL(R1, R2), DL(R2, R3)**, HL(R3, F1))
Case2′: (S1 || R1 | R3 || F1) → (HL(S1, R1), DL(R1, P), DL(P, R3), HL(R3, F1))

Case2 and Case2′ compose a pair of primary cases that can be involved in a transmutation by solution substitution as defined in Sect. 3.2. They have the same problem situation for two different solution actions.

An appropriate substitution case for this pair of primary cases would be Case5. Its solution action includes all stretches specified by Case2. It is structured as follows:

Case5: (S1 || R1 | R3 || R4 || F1) → (HL(S1, R1), **DL(R1, R2), DL(R2, R3),** HL (R3, R4), HL(R4, F1))

Transmutation of the trio Case2, Case2′ and Case5 yields the new case Case5′. It is equal to:

Case5′: (S1 || R1 | R3 || R4 || F1) → (HL(S1, R1), DL(R1, P), DL(P, R3), HL(R3, R4), HL(R4, F1))

Next, let us consider similarity between Case1 and Case2′ with regard to problem situations. We notice that both sol1 and sol2′ are solutions for prb2′. They both pass through R1 and R3 checkpoints and have the same start and end points S1 and F1. This implies that sol1 may be substituted for sol2′. However prb1 is not resolved by sol2′. Thus, sol2′ cannot be substituted for sol1.

Case1 and Case2′ thus compose a pair of primary cases, which can be involved in a transmutation by solution substitution. Case5 cannot be an appropriate substitution case for that pair of primary cases, since sol2′ cannot be substituted for sol1. But Case6 does; its solution action includes all lines specified by Case2′. It is given by:

Case6: (S1 || R1 | R3 || R4 || F1) → (HL(S1, R1), **DL(R1, P), DL(P, R3),** HL(R3, R4), HL(R4, F1))

Transmutation of this new trio Case1, Case2′ and Case6 yields Case6′.

Case6′: (S1 || R1 | R3 || R4 || F1) → (HL(S1, R1), **DL(R1, R2), DL(R2, R3),** HL (R3, R4), HL(R4, F1))

6 Validation of the Randomized Knowledge

6.1 Validation Rules

Consistency of the generated cases must be verified, since these latter cases have been generated using weak dependencies that cannot assure valid resulting cases. Hence, each generated case must be checked for solution consistency, problem consistency and case consistency. If consistency of a generated case is proved, then the latter can be used by the system. This means that its related trio is saved in the case base.

- **Solution consistency,** i.e. is the generated solution syntactically and semantically correct with regard to the domain of interest? For example, a non-continuous path from a source to a destination is an invalid solution.
- **Problem consistency,** i.e. does the generated problem contain an incoherence with regard to the domain of interest? For example, the problem has two start points.
- **Case consistency,** i.e. does the solution correspond to the associated problem? Does it achieve the goals targeted by the problem and satisfy its constraints? The problem and solution must have been previously validated.

6.2 Application to Strong Sequencing

Validation of the problem part consists in checking that: (i) the start and end points are the same as those of the delegate; (ii) substitute points are linked with the connector ||.

This is true with prb3′, since it includes the delegate's start and end points, namely S and F; and its sustitute points are linked with ||.

Validation of the solution part consists in checking that: (i) the first point of the first line is the start point of the delegate, and the last point of the last line is the end of the delegate; (ii) all lines are connected, thus forming a continuous route.

This is true with sol5′, since the first point of the first line is S, the start point of the delegate, and the last point of the last line is F, the end of the delegate; besides, all lines are connected, thus forming a continuous route.

Validation of a case consists in checking the following: (i) all points of the problem part must appear in the lines of the route composing the solution part; (ii) only those points must apear in the lines of the route; (iii) the order of these points must be kept the same in the lines composing the route as that of the problem part.

This is true with Case3′ and Case5′, since all points of their problem part appear in the lines of their routes; only those points apear in the lines of the routes; the order of these points is kept the same in the lines composing the route as that of the problem part.

6.3 Application to Weak Sequencing

Validation of the problem part consists in checking that: (i) the start and end points are the same as those of the delegate; (ii) all points are linked with the connector |.

This is true with prb4′, since it includes the delegate's start and end points, namely S and F; and its points are linked with |.

Validation of the solution part consists in checking that: (i) the first point of the first line is the start point of the delegate, and the last point of the last line is the end of the delegate; (ii) all lines are connected, thus forming a continuous route.

This is true with sol6′, since the first point of the first line is S, the start point of the delegate, and the last point of the last line is F, the end of the delegate; besides, all lines are connected, thus forming a continuous route.

Validation of a case consists in checking the following: (i) all points of the problem part must appear in the lines of the route composing the solution part.

This is true with Case4′ and Case6′, since all points of their problem part appear in the lines of their routes; and the order of these points is kept the same in the lines composing the route as that of the problem part.

6.4 Application to Free Sequencing

Validation of the problem part consists in checking that: (i) the start and end points are the same as those of the delegate; (ii) all points are linked with the connector ";".

This situation does not appear in our application.

Validation of the solution part consists in checking that: (i) the first point of the first line is the start point of the delegate, and the last point of the last line is the end of the delegate; (ii) all lines are connected, thus forming a continuous route.

Validation of a case consists in checking the following: (i) all points of the problem part must appear in the lines of the route composing the solution part; (ii) the order of these points may be different in the lines composing the route from that of the problem part.

7 Transmutation in Case Based-Reasoning

In this section, we will show how randomization is applied to case-based reasoning. It should increase the probability to retrieve the searched case by improving retrieval of new problems and/or retrieval of the pertinent solutions. We will show how knowledge induction either by problem transmutation and solution transmutation benefit, each one on its side, to case-based reasoning.

7.1 Problem Transmutation in CBR

Knowledge generation by problem transmutation provides new cases with novel problems and their corresponding solutions. We are, for example, interested in searching a route including the following points: (S1||R1||R2||R3||R4||R5||F1). This succession of points composes the problem part of the case, which we note prb. To retrieve prb in the case base, we proceed as follows.

First, the segment whose delegate is given by S1-F1 is retrieved. We note that the problem prb does not match with any problem part of the cases belonging to the segment. Transmutation is performed on every trio within the segment to generate new cases with a problem that would match the searched problem prb. Especially, the following trio provides a new case case3′, whose problem matches prb:

Case1: (S1 **|| R1 || R2 || R3 ||** F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL(R3, F1))
Case2: (S1 || R1 | R3 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL(R3,F1))
Case3: (S1 **|| R1 | R3 ||** R4 || R5 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL (R3,R4), HL(R4,R5), HL(R5,F1))
Case3′: (S1 || R1 || R2 || R3 || R4 || R5 || F1) → (HL(S1,R1), DL(R1,R2), DL(R2, R3), HL(R3,R4), HL(R4,R5), HL(R5,F1))

Since the solution of case3′ has already been validated as a good solution in the phase of randomization, it will be retained for the searched problem prb.

7.2 Solution Transmutation in CBR

Knowledge induction by solution transmutation cannot help retrieving a searched problem, as problem transmutation does. Rather, it may help improving retrieval of the good solution by providing more than one solution for the searched problem. The user will be offered to choose the most convenient one to his problem.

We are, for example, interested in searching a route including the following points: (S1||R1|R3||R4||F1). This problem search involves a solution transmutation on the following trio:

Case2: (S1 ‖ R1 | R3 ‖ F1) → (HL(S1,R1), DL(R1,R2), DL(R2,R3), HL(R3,F1))
Case2′: (S1 ‖ R1 | R3 ‖ F1) → (HL(S1,R1), **DL(R1,P), DL(P,R3),** HL(R3,F1))
Case5: (S1 ‖ R1 | R3 ‖ R4 ‖ F1) → (HL(S1,R1), **DL(R1,R2), DL(R2,R3),** HL(R3, R4), HL(R4,F1))

This results in a new case5′, whose problem is the same as that of case5; but, their solutions are different.

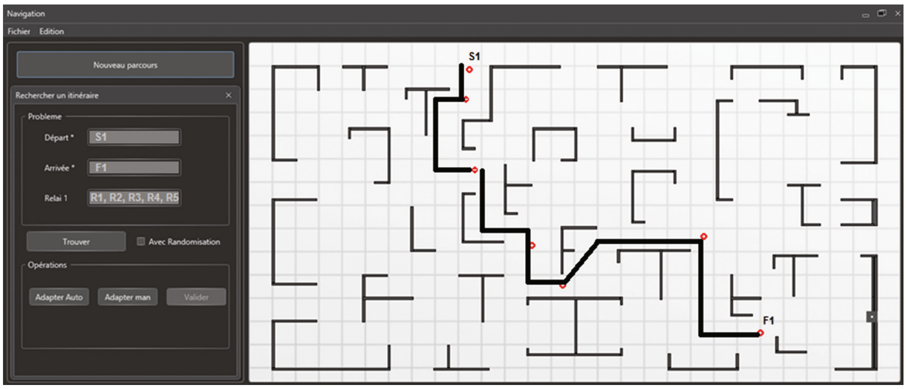
Case5′: (S1 ‖ R1 | R3 ‖ R4 ‖ F1) → (HL(S1,R1), DL(R1,P), DL(P,R3), HL(R3, R4), HL(R4,F1))

It follows that the user will be provided with a double solution for the searched problem.

8 Implementation

Figure 3 shows the user interface and the result for a transmutation by problem substitution of the trio Case7-Case8-Case9, thus inducting Case9’. Here, the comma repre-sents a weak sequencing. The trio is structured as follows:

Case7: (S1, R1, R2, R3, F1) → (VL(S1,R1), HL(R1,A), VL(A,B), HL(B,R2), VL (R2,C), HL(C,R3), VL(R3,D), HL(D,G), VL(G,H), HL(H,F1))
Case8: (S1, **R1, R3,** F1) → (VL(S1,R1), HL(R1,A), VL(A,B), HL(B,R2), VL(R2, C), HL(C,R3), VL(R3,D), HL(D,G), VL(G,H), HL(H,F1))
Case9: (S1, **R1, R3,** R4, R5, F1) → (VL(S1,R1), HL(R1,A), VL(A,B), HL(B,R2), VL(R2,C), HL(C,R3), VL(R3,D), HL(D,R4), DL(R4,E), HL(E,R5), VL(R5,F), HL (F,F1))
Case9′: (S1, R1, R2, R3, R4, R5, F1) → (VL(S1,R1), HL(R1,A), VL(A,B), HL(B, R2), VL(R2,C), HL(C,R3), VL(R3,D), HL(D,R4), DL(R4,E), HL(E,R5), VL(R5, F), HL(F,F1))



Départ * S1
Arrivée * F1
Relai 1 R1, R2, R3, R4, R5

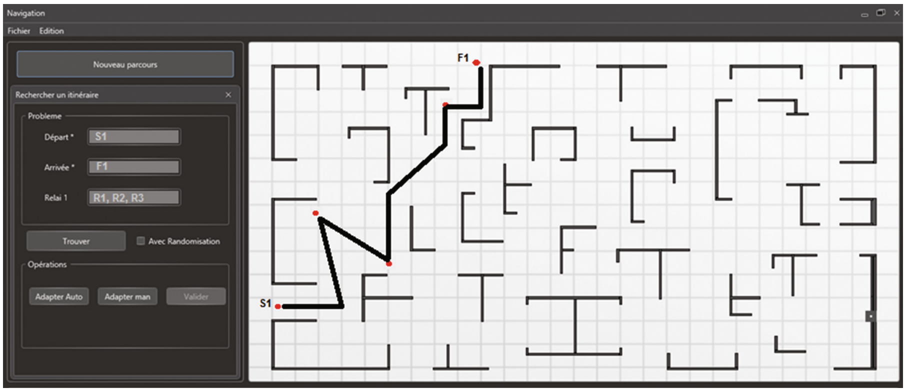
Fig. 3. Transmutation by problem substitution

Induction of Case9′ allows for providing a solution for the searched problem (S1, R1, R2, R3, R4, R5, F1). It generates a new case, with an additional checkpoint R2, in the path between S1 and F1, through the known checkpoints R1, R3, R4, R5.

Figure 4 shows the result for a transmutation by solution substitution of the trio Case10-Case11-Case12, thus inducting Case12′. The trio is structured as follows:

Case10: (S1, R1, R2, F1) → (HL(S1,A), DL(A,R1), DL(R1,R2), VL(R2,B), DL (B,F1), VL(R3,F1))
Case11: (S1, R1, R2, F1) → (HL(S1,A), DL(A,R1), DL(R1,R2), **VL(R2,C), DL (C,F1)**, VL(R3,F1))
Case12: (S1, R1, R2, R3, F1) → (HL(S1,A), DL(A,R1), DL(R1,R2), VL(R2,B), DL(B,D), VL(D,R3)), HL(R3,E), VL(E,F1))
Case12′: (S1, R1, R2, R3, F1) → (HL(S1,A), DL(A,R1), DL(R1,R2), **VL(R2,C), DL(C,D)**, VL(D,R3)), HL(R3,E), VL(E,F1))

Induction of Case10′ allows for providing a new route passing through the three checkpoints R1, R2 and R3.



Départ * S1
Arrivée * F1
Relai 1 R1, R2, R3

Fig. 4. Transmutation by solution substitution.

9 Conclusion

The approach proposed in this article allows for an enhancement of problem resolution in case-based reasoning systems by amplifying knowledge while saving space and speeding up search. Knowledge amplification is based on a technique of trios transmutation. Space saving is deduced from the key idea around case transmutation. Implicit knowledge within the case base is materialized, every time it is necessary - without the need to be saved. Only the relevant information, namely the valid transmutation trios, allowing for this materialization is saved. Knowledge materialization is obtained by transmutation, which scales well with CBR systems. Time gains are obtained through the proposed segmentation strategy based on delegates generalizing the problem description.

However, the inducted knowledge is not necessarily valid. It necessitates validation. This validation is domain- specific. We proposed to formalize the semantics of a domain specific application and validate the randomized knowledge.

References

1. Rubin, S.H.: Computing with words. IEEE Trans. Syst. Man Cybern. Part B **29**(4), 518–524 (1999)
2. Rubin, S.H., Murthy, S.N.J., Smith, M.H., Trajkovic, L.: KASER: knowledge amplification by structured expert randomization. IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**(6), 2317–2329 (2004)
3. Rubin, S.H.: Case-Based Generalization (CBG) for Increasing the Applicability and Ease of Access to Case-Based Knowledge for Predicting COAs. NC No. 101366 (2011)
4. Rubin, S.H.: Multilevel Constraint-Based Randomization Adapting Case-Based Learning to Fuse Sensor Data for Autonomous Predictive Analysis. NC 101614 (2012)
5. Rubin, S.H., Bouabana-Tebibel, T.: NNCS: randomization and informed search for novel naval cyber strategies. In: Recent Advances in Computational Intelligence in Defense and Security. Studies in Computational Intelligence, vol. 621, pp. 193–223. Springer, Cham, December 2015
6. Rubin, S.H., Bouabana-Tebibel, T.: Naval intelligent authentication and support through randomization and transformative search. In: New Approaches in Intelligent Control and Image Analysis. Intelligent Systems Reference Library, vol. 107, pp. 73–108. Springer (2016)
7. Rubin, S.H.: Learning in the large: case-based software systems design. In: IEEE International Conference on Systems, Man, and Cybernetics, Decision Aiding for Complex Systems, Conference Proceedings (1991)
8. Bouabana-Tebibel, T., Rubin, S.H., Chebba, A., Bédiar, S., Iskounen, S.: Knowledge induction based on randomization in case-based reasoning. In: The 17th IEEE International Conference on Information Reuse and Integration – IEEE IRI 2016, Pittsburgh, USA, 28–30 July 2016
9. Begum, S., Ahmed, M.U., Funk, P., Xiong, N., Folke, M.: Case-based reasoning systems in the health sciences: a survey on recent trends and developments. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **41**(4), 421–434 (2011)
10. Bridge, D., Healy, P.: GhostWriter-2.0 case-based reasoning system for making content suggestions to the authors of product reviews. Knowl. Based Syst. **29**, 93–103 (2012)
11. Wang, C., Yang, H.: A recommender mechanism based on case-based reasoning. Expert Syst. Appl. **39**(4), 4335–4343 (2012)
12. Hu, J., Guo, Y., Peng, Y.: A CBR system for injection mould design based on ontology: a case study. Comput. Aided Des. **44**, 496–508 (2012)
13. Yang, H.Z., Chen, J.F., Ma, N., Wang, D.Y.: Implementation of knowledge-based engineering methodology in ship structural design. Comput. Aided Des. **44**, 196–202 (2012)
14. Xie, X., Lin, L., Zhong, S.: Handling missing values and unmatched features in a CBR system for hydro-generator design. Comput. Aided Des. **45**, 963–976 (2013)
15. Ahn, H., Kim, K.: Global optimization of case-based reasoning for breast cytology diagnosis. Expert Syst. Appl. **36**, 724–734 (2009)

16. Naderpajouh, N., Afshar, A.: A case-based reasoning approach to application of value engineering methodology in the construction industry. Constr. Manag. Econ. **26**(4), 363–372 (2008)
17. Zarandi, F.M., Razaee, Z.S., Karbasian, M.: A fuzzy case based reasoning approach to value engineering. Expert Syst. Appl. **38**(8), 9334–9339 (2011)
18. Shi, L.-X., Peng, W.-L., Zhang, W.-N.: Research on reusable design information of shoes based on CBR. In: 2012 International Conference on Solid State Devices and Materials Science. Physics Procedia, vol. 25, pp. 2089–2095. Elsevier (2012)
19. Fan, Z.-P., Li, Y.-H., Wang, X., Liu, Y.: Hybrid similarity measure for case retrieval in CBR and its application to emergency response towards gas explosion. Expert Syst. Appl. **41**, 2526–2534 (2014). Elsevier
20. Khanuma, A., Muftib, M., Javeda, M.Y., Shafiqa, M.Z.: Fuzzy case-based reasoning for facial expression recognition. Fuzzy Sets Syst. **160**, 231–250 (2009)
21. Mittal, A., Sharma, K.K., Dalal, S.: Applying clustering algorithm in case retrieval phase of the case-based reasoning. Int. J. Res. Aspects Eng. Manag. **1**(2), 14–16 (2014)
22. Behbood, V., Lu, J., Zhang, G.: Fuzzy refinement domain adaptation for long term prediction in banking ecosystem. IEEE Trans. Ind. Inf. **10**(2), 1637–1646 (2014)
23. Shell, J., Coupland, S.: Fuzzy transfer learning: methodology and application. Inf. Sci. **293**, 59–79 (2015)
24. Li, S., Ho, H.: Predicting financial activity with evolutionary fuzzy case-based reasoning. Expert Syst. Appl. **36**(1), 411–422 (2009)
25. Sampaio, L.N., Tedesco, P.C.A.R., Monteiro, J.A.S., Cunha, P.R.F.: A knowledge and collaboration-based CBR process to improve network performance-related support activities. Expert Syst. Appl. **41**, 5466–5482 (2014)
26. Lao, S.I., Choy, K.L., Ho, G.T.S., Yam, R.C.M., Tsim, Y.C., Poon, T.C.: Achieving quality assurance functionality in the food industry using a hybrid case-based reasoning and fuzzy logic approach. Expert Syst. Appl. **39**, 5251–5261 (2012)
27. Kumar, Y.J., Salim, N., Abuobieda, A., Albaham, A.T.: Multi document summarization based on news components using fuzzy cross-document relations. Appl. Soft Comput. **21**, 265–279 (2014)
28. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. (CSUR) **31**, 264–323 (1999)