Journal of Computer Science 1 (3): 419-423, 2005
ISSN 1549-3636
© Science Publications, 2005

Applying Evolutionary Computing in Case Retrieval Stage

Nabila Nouaouria-Amri and Med Tayeb Laskri
Department of Computer Science, Laboratory of Computer Science (LRI), Research Group on Artificial Intelligence, University Badji Mokhtar of Annaba, B.P. 12, Annaba (23000), Algeria

Abstract: Case Based Reasoning is a problem solving paradigm which is able to retrieve and reuse solutions that have worked for similar situations in the past. Past situations and their solutions are stored in a memory called case base. To find the good experiment in memory is the key of success in the reasoning. To identify adequate experiment in memory constitutes the process of recall. The study presents an associative memory model used for a Case-Based Reasoner. The search algorithm is funded on an evolutionary approach to compute neighbourhood of a new problem then a direct access is performed.

Keywords: Artificial Intelligence, Case Based Reasoning, Evolutionary Computing, Case Retrieval

INTRODUCTION

"Reasoning is Remembering" is the slogan of most researchers in Case Based Reasoning field. Case Based Reasoning (CBR) is a problem solving methodology founded on reusing old solution that have worked for similar situations in the past to solve new problem. Past situations and their solutions are stored in a memory called case base.

To find the good experiment in memory is the key of success in the reasoning. The good experiment is the one that can perform the best inferences. To identify adequate experiment in memory is the recall process. Recall is highly influenced by memory organization and by retrieve strategies. The accuracy (in the sense of exhaustiveness) and speed of recall task constitute two important parameters in the performance evaluation of a CBR system.

Case based reasoning is an Artificial Intelligence paradigm that can be synergistically combined with other approaches to facilitate a broad array of tasks [1].

Among those possible combinations, we present in the following, an approach to perform a quick and complete recall, in an associative memory, using evolutionary computing.

The main idea is to compute the neighbourhood of a new problem by an evolutionary algorithm. This draws up the boundaries of the search space in the case base. And then, attains directly this neighbourhood via a network in an associative memory style.

For a best understanding of the study, we start with a fast skimming of CBR paradigm, followed by the memory model proposed. Then the general framework is depicted. The conclusion section goes over the main points, it also presents related works and future direction

CASE BASED REASONING PARADIGM

The idea of CBR is intuitively pleasing because it is similar to human problem-solving behaviour. People sketch on past experience while solving new problems and this approach is both convenient and effective and it often reduces the burden of depth analysis of the problem domain [2]. This leads to the advantage that CBR can be based on shallow knowledge and does not require significant effort in knowledge engineering when compared with other approaches (e.g., rule-based).

Problem solving with CBR proceeds as follows: a new problem is posed and is described as the problem part of a new case, sometimes also called the query. Then, old cases containing problems that are similar to the new problem are retrieved and the most suitable solution among retrieved solutions is suggested to become the solution of the new problem. This solution is then tested in reality and may lead to a revised solution worth to be Stored as a new case. This last step is a form of incremental learning that enables CBR systems to adapt to changing environments rather smoothly.

In theory, the basic cycle of CBR is in three phases: «retrieve, reuse and store». The system looks for a similar case to the input case, reuse the recovered solution and finally, store the current case for a future utilization.

This cycle can be extended to five stages [2, 3]:

Presentation or Specification: A description of the problem is provided at the entrance of the system. This description must be suitable to the comparison between the case in entrance and cases stored in memory (uniformity of the representation). One of the key points of the CBR is the research of applicable cases. It is what justifies the importance of the process that is going to label cases with indexes so that they could be recalled at the appropriate moment. This indexing leans mainly on the extraction of the most characteristic descriptors of the case.

Retrieval: The system looks for cases that are best unified to this description (closest matching cases). These cases are stored in a case base or case memory (i.e.: data base of cases). If the case base is organized according to a particular structure, an algorithm of research describes then a path in this structure. A phase of filtering or selection is often performed in order to eliminate a subset of worst cases. A measure of similarity can be then used to refine the resemblance measure between the current case and selected cases. Then returns ordered cases.

Adaptation: The system uses the current problem and the matching case to generate a solution to this problem. The adaptation constitutes the second difficult point (after the indexing) when conceiving a CBR system. It is necessary to decide what type of knowledge it is interesting to transfer from the best case remembered. We can do a transformational analogy, consisting in transforming the solution of retrieved case to adapt it to the current case. Or to proceed by derivation when adapting the method of solution generation. Otherwise, the possibility to adapt several cases to solve a problem, in a simultaneous way or operating several remembering and simple adaptation to the different stages of the resolution, has been judged more creative [5].

Validation: This phase includes the possibility of an assessment of the solution proposed while testing it in an either simulated or real environment. The information returned guides a repair process, in case of failure of the proposed solution.

Storage: The validated solution is added to the case base for a future utilization. We can have systems which store cases systematically in memory. A more selective memorization is however possible and would use some specific criteria to judge if the new case is useful to learn according to the current case memory. Generally, a case is useful to learn when it can reach a point of the solution space that was inaccessible before the arrival of this new case.

THE MEMORY MODEL PROPOSED

In order to function correctly, the case based reasoning uses cases stored in a case base. This one is supposed to be representative of the whole problems encountered in the field. The more it contains cases, the more the case selected for the reasoning will be similar to the new case. The elaborate solution will be thus better. But more the base increases, more prohibitive will be the calculating cost. This is why techniques of memory organization and search algorithms are particularly important in this reasoning mode.
There are several memory organizations according to which search algorithms exist [2, 3]. The most frequently used models of memory relying on a Top-Down search, present some common features [4]:

* They support a structuring of data by regrouping together related objects.
* They support an efficient retrieval by utilizing traditional tree search algorithms.
* Traversing a Top-Down memory structure is performed by answering questions in the internal nodes in order to choose which path to follow. This requires a specific order in the answers. In the case of incomplete information, it could mislead the utilization of an erroneous path.
* Once a certain cluster of cases has been reached in the leaf of a tree, it is hard to access neighbouring clusters containing similar cases.

For those reasons, we will expose another vision of retrieval problem based on the construction of problem neighbourhood.
The case memory is indeed, a flat structure on which we construct a nested structure. There are two types of node: value node and case node.
Each value node represents a particular value of a problem attribute (Fig. 1). It is linked to all case nodes where it occurs.
The case node point out to the case base where the whole case is stored.
The particularity of this structure is that we reach the case by its contents (the principle of associative memories).
Another particularity is that the structure could be easily and automatically build by simply scanning the case memory.

The General Framework: The retrieval of applicable cases can be formulated in how to extract from the search space a sub-space of cases that are similar to the problem to resolve. This sub-space is what we call neighbourhood of the target problem. It is classically obtained by a search strategy.
The main idea is to compute the neighbourhood of a new problem by an evolutionary algorithm (Fig. 2). And then, access directly to this neighbourhood via a network in an associative memory style.
Every source problem computed by the evolutionary module will be directly pointed in the search space via the net.
A case is an entity within which is gathered various information on a past situation. The term «situation» is very general. A case is also an entity about which an

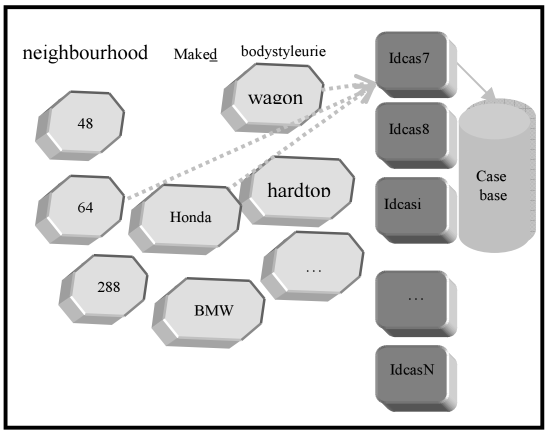

Fig. 1: The Associative Memory

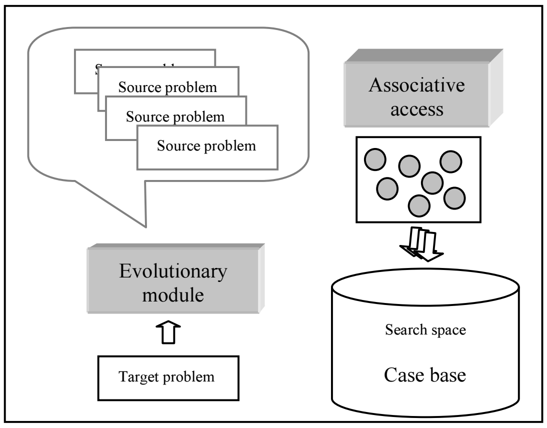

Fig. 2: A Global Vision

inference is possible by situating the new problem with regard to the definite circumstances in the case.
A case is constituted of descriptors, also called dimensions, distributed in three categories: the description of the problem, the solution and issues of the solution.
The description of the problem includes the context of the case. The solution is the solution of the problem or the reaction to this description (for example, the deliberation of a courthouse, the taken decision, etc.). It can also describe the used reasoning. The exit of the case is the description of the context after the implementation and execution of the solution. This part of the case is generally omitted and knowledge is reported on the other stages of the reasoning.
The retrieval step is based on problem description only. We focus know on the evolutionary module and propose a coding of problem description.

Coding: When a new problem is posed, the request to retrieve similar cases is generally, expressed with dimensions of problem description (Fig. 3).

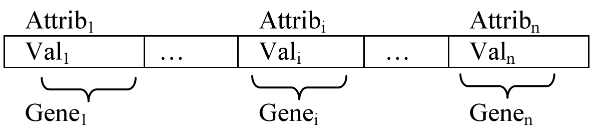

Fig. 3: Problem Description as a Chromosome

The coding of problem will be:

Table 1: Matching between CBR and EC Entities

Problem description : pbm	Chromosome
Descriptors : d_i	genes
Descriptor values val_j	Alleles

Pbm = { d_i } : an array of descriptors.
d_i = (Attribi,valij): a couple of attribute/value
val_{ij} € Dom_j : each value belongs to a specific domain which could be symbolic or numeric.

For our experimentation, we have used a sub set of ‘auto import database’, an UCML dataset. We are interested in:
problem description =<make, bodystyle, horsepower>

$\text{Domaine}_{\text{make}}$ = {alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, Volvo} discrete

So if we make a decimal coding:
Make € [1, 22] → 2 digits

For a binary coding we have:
Make € [00000, 10110] → 5 bits

$\text{Domaine}_{\text{bodystyle}}$ = { hardtop, wagon, sedan, hatchback, convertible } discrete

So if we make a decimal coding:
Bodystyle € [1, 5] → 1 digit

For a binary coding we have:
bodystyle € [000, 101] → 3 bits

$\text{Domaine}_{\text{horsepower}}$ = [48, 288] continuous
So if we make a decimal coding:
Horsepower € [48, 288] → 3 digits

For a binary coding we have:
horsepower€ [000110000,100100000] → 9 bits

An example of chromosome could be:

For a decimal coding

1	3	2	0	6	8

Binary coding is longer:
01101 010 001001000

Which correspond to the description of the following problem :

Prob=<nissan, wagon, 68>

The initial population is randomly generated.

The selection step is based on a strategy of similarity to the request (new problem). With a general shape :

$$D(G,Gk) = \sum wi\ di(G,Gk) \text{ for } i € [1, 3]$$

Where: wi is the weight of the descriptor i (gene i) and di is the partial distance:

$$di = 1-(|X^i-X^i_k|/\text{maximal discard})$$

Reproduction is essentially made by:

Cross-over: for both binary coding and decimal coding we have two crossing points. They are separations between genes.

Mutation of genes: The chromosome mutation corresponds to the troubling of the entry problem description in order to generate a neighbourhood.

The fitness function is based on similarity assessment between the input problem and the actual chromosome. It has the following form:

Maximizing $\sum$ D(G,Gk) for k=1 to N (N population cardinality).

The whole algorithm will be:

```
1. Initialise a population of chromosomes.
2. Evaluate each chromosome in the population.
3. Create offspring problems population by
   crossing then mating the current generation.
4. Evaluate offspring population.
5. if <stop criterion> is satisfied then stop else
   goto 3.
```

The stop criterion = population stabilisation or max Time

For the following input problem description :

1	3	2	0	6	8

An example of population (with card =5):

For decimal coding:

Ind1	0	9	4	1	1	6	0.76
Ind2	1	1	2	0	7	0	0.96
Ind3	0	4	3	1	6	0	0.76
Ind4	2	2	1	2	0	7	0.58
Ind5	1	4	3	1	1	0	0.84

The last column represents the selection function.

A first simulation with decimal coding leads to results presented in Fig. 4.

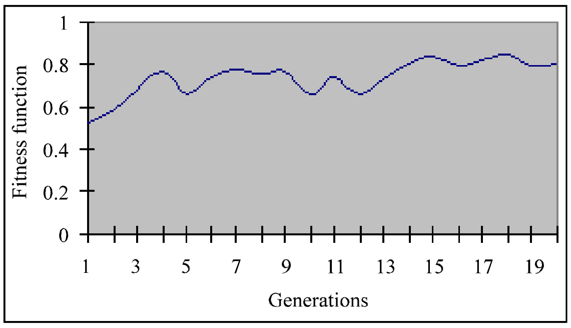

Fig. 4 : Simulation Results

We ought to notice that in our study, both selection function and fitness function expresses the same semantic since we aim to retrieve the most similar problems.

CONCLUSION

Many different approaches of case memory models have been proposed in literature [6]. However Evolutionary computing approach seems to be interesting for multiple reasons [11, 12]:

* Flexible knowledge representation.
* Good computation performances.
* Suitable for space exploration.
* A large scale of applicability.

Up to now their application in CBR was limited to the adaptation task. An evolutionary approach to case adaptation is presented in [7]. In [8], case adaptability is improved by a Genetic Algorithm.

Our approach leans on a memory structure reachable by the contents. Flexible, easy to construct and having a uniform knowledge representation according to the Evolutionary computing module.

It is very important to emphasize that the presented approach represents a general framework. When considering a specific application field we have to tune parameters of our system in order to improve the convergence.

Since our previous work was on adaptability guided retrieval memory [9, 10] it is interesting to consider an extension of the approach where the adaptability criterion is integrated to the fitness function.

REFERENCES

1. Marling, C. *et al*., 2002. Case-Based reasoning Integrations. In AI Magazine, Vol: 23.
2. Kolodner, J., 1993. Case Based Reasoning. Ed. Morgan Kaufmann.

3. Aamdot, A. and E. Plaza, 1994. Case Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. Published in IOS Press, 7: 39-59.
4. Lenz, M. *et al*., 1998. Diagnosis and decision support. In LNAI 1400, Éd. Springer.
5. Kolodner, J., 1992. Judging which is the best case for a case based reasoner. In DARPA Workshop on CBR.
6. Nouaouria, N. and M.T. Laskri, 2003. Toward a formal model of case based reasoning from the roots. Proc. CESA’2003, Lille France, pp: 9-11.
7. Gomez de Silva Garza, A. and M.L. Maher, 1999. An Evolutionary Approach to Case Adaptation. 3rd ICCBR’99, in LNAI 1650, Germany.
8. Purvis, L. and S. Athalye, 1997. Towards Improving Case Adaptability with a Genetic Algorithm. 2rd ICCBR’97, in LNAI 1266, USA.
9. Nouaouria, N. and M.T. Laskri, 2005. Adaptability versus Similarity : Why not the two. In proc. of 9th UKWCBR, Cambridge, UK.
10. Nouaouria, N. and M.T. Laskri, 2005. Case Retrieval Nets Augmented with an Adaptability Criterion. to appear in MJCS, Vol: 18.
11. Pal, S.K. and S.C. Shiu, 1994. Foundations of Soft Case-Based Reasoning. Ed. John Wiley and Sons Inc, 2004.
12. Goldberg, D.E., 1994. Genetic algorithms. Ed. Addison Wesley.