PRINCIPLES OF CASE-BASED REASONING

Miquel Sànchez-Marrè

Secció d'Intel·ligència Artificial
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campous Nord-Edifici C5
C. Jordi Girona 1-3. 08034 Barcelona.
miquel@lsi.upc.es

1.1 The cognitive model

Case-Based Reasoning (CBR) [Aamodt and Plaza, 1994; Kolodner, 1993; Riesbeck and Schank, 1989] derives from a view of understanding problem-solving as an explanation process. The foundations of Case-based Reasoning rely on the early work done by Schank and Abelson [Schank and Abelson, 1977] where they proposed that our general knowledge about situations is recorded as *scripts*. The cognitive model behind the Case-based reasoning is based on the theory of Dynamic Memory [Schank, 1982] that introduces indexing as the key to use experience in understanding. The main premise was that remembering, understanding, experiencing, and learning cannot be separated from each other, and that the human memory is dynamic, and change as a result of its experiences.

CBR systems improve their performance becoming more efficient by remembering old solutions given to similar problems and adapting them to fit a new problem rather than having to solve it from scratch. This, in fact, augments the ideas about the components of expertise [Steels, 1990] using the solved cases as an episodic memory: the memorisation of problem-solved episodes allows methods to be integrated since they require accessing the past experience to improve the system performance. Also, Case-based reasoners become more competent in their evolution over time, so that they can derive *better* solutions when faced against less experienced situations, preventing them to repeat the same mistakes in the future (learning process).

1.2 Organisation and Representation of Cases

The reasoning by analogy of CBR is based in collecting a lot of relevant cases or experiences in a particular domain. Storing a case means to keep a description of the experience as well as the solution provided to that experience. The set of stored cases or experiences is usually named as the Case Library or the Case Base or the Case Memory. In the next subsections, the main case library organisations and case structures will be described.

1.2.1 Case Library

Main memory organisations, in CBR systems, can be summarised in two general approaches: *flat memories* and *hierarchical memories* such as shared feature networks, prioritised discrimination networks/trees or redundant discrimination networks/trees.
Flat memories always retrieve the set of cases best matching the input case. Moreover, adding new cases to memory is cheap. But they have a major disadvantage: the retrieval time is very expensive since every case in the memory is matched against the current case, commonly using a Nearest Neighbour (NN) algorithm [Watson, 1996].
On the opposite side there are the *hierarchical memories.* In such kind of memories, matching process and retrieval time are more efficient, due to the fact that only few cases are considered for similarity assessment purposes, after a prior discriminating search in the hierarchical structure. Anyway, they also have some disadvantages. Keeping the hierarchical structure in optimal conditions requires an overhead on the case library organisation, and the retrieval process could miss some optimal cases searching a wrong area of the hierarchical memory. This later problem specially becomes hard in prioritised discrimination networks/trees.

1.2.2 Case Structure

The cases stored in the case library are real experiences, which have been captured and learned in such a way that they can be reused to solve future causalities. A case does incorporate a set of features such as:

- an *identifier* of the case
- the *description* of the case
- the *diagnostic* of the case
- the *solution* of the case
- the *derivation* of the case, i.e. from where the case has been derived/adapted
- the *solution result*, information indicating whether the proposed case solution has been a successful one or not
- an *utility measure* of the case in solving past cases when it was used
- other *relevant information* about the case

An example of a case representation could be the following:

```
(  :identifier  CASE-18
   :situation-description  ( (Water-inflow 35,198 m3/day)
                            (Inflow-Chemical-Oxygen-Demand 289 mg/L)
                                        . . . )
   :diagnostics  NORMAL-SITUATION
   :actuation-plan  ( (1 Maintain-the-numerical-control-algorithm)
                      (2 Adjust-Dissolved Oxygen (DO)-value)
                         . . . )
   :case-derivation  INITIAL-CASE / CASE-13
   :solution-result  SUCCESS /FAILURE
   :utility-measure  0.7
   :distance-to-case  0.3782 )
```

1.3 The Case-Based Reasoning Cycle

The basic reasoning cycle of a CBR agent can be summarised by a schematic cycle (see figure 1.1). In Aamodt and Plaza [Aamodt & Plaza, 1994] they adopt the four REs schema:

• *Retrieve* the most similar case(s) to the new case.

• *Adapt* or *Reuse* the information and knowledge in that case to solve the new case. The selected best case has to be adapted when it does not match perfectly the new case.

• *Evaluate* or *Revise* of the proposed solution. A CBR-agent usually requires some feedback to know what is going right and what is going wrong. Usually, it is performed by simulation or by asking to a human oracle.

• *Learn* or *Retain* the parts of this experience likely to be useful for future problem solving. The agent can learn both from successful solutions and from failed ones (repair).

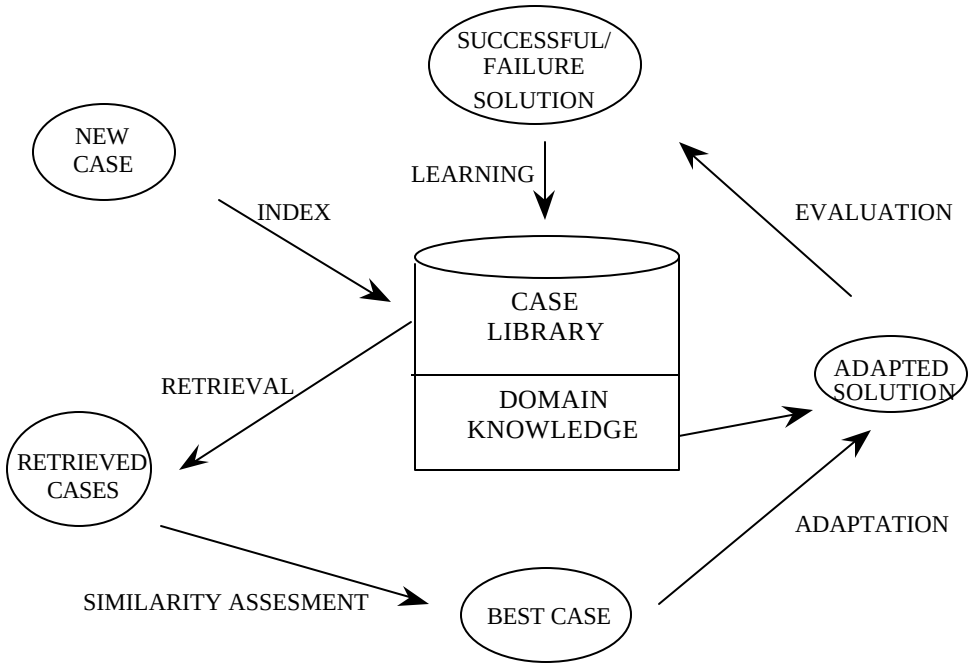

Figure 1.1. The general case-based reasoning paradigm

The quality of the new case(s) extracted by the CBR-agent depends upon some of the following criteria: a) The usefulness of the case(s) extracted and selected; b) The ease of use this (these) case(s); c) The validity of the reasoning process; and d) The improvement of knowledge through experience.

When setting a CBR-agent one has to take into account some design decisions or static problems such as: a) How to describe the domain problems? b) Which will be the case structure? c) Which will be the case library structure? d) How to deal with the missing information problem? e) Which will be the criterion for indexing the case library? and f) How to asses similarity between cases?

2.3.1 Case Retrieval and Similarity Assessment

The task of retrieving cases in the case library is slightly more difficult than typical retrieval in databases. In database systems the recalling algorithms use an exactly matching method, whereas in a case library retrieval, because the very nature of the structure, a partial-matching strategy should be used. A retrieval method should try to maximise the similarity between the actual case and the retrieved one(s). And this task implies most of the time the use of general domain knowledge.

The retrieving process of a case (or a set of cases) from the system's memory strongly depends on the case library organisation. Major case library structures are flat memories or hierarchical ones. Flat memories have an intrinsic problem of bad performance in time, so that

the retrieval time is proportional to the size of the case library. Hierarchical memories are very effective in time retrieval because only a few cases are considered for similarity assessment purposes, after a prior discriminating search in the hierarchical structure. Although sometimes could not reach optimal cases because is exploring a wrong area of the hierarchy.
The retrieval process in hierarchical case libraries, usually consists of two main substeps:

• *Searching* the most similar cases to the new case: the goal of this stage is recalling the most promising cases –given that the subsystem has a goal and therefore the relevance of the cases depends upon that goal– based on using some direct or derived features of the new case as indexes into the case library.

• *Selecting* the best case(s): the best case(s) among those ones collected in the previous step are selected. Commonly, this selection is made by means of a case ranking process through a similarity or distance function. The best-retrieved case is the closest one (most similar) to the new case.

Selecting the best similar case(s), it is usually performed in most Case-based reasoning agents by means of some evaluation heuristic functions or distances, possibly domain dependent. They are usually named as nearest neighbour (NN or k-NN) algorithms [Watson, 1996]. The evaluation function usually combines all the partial matching through a dimension or attribute of the cases, into an aggregate or full-dimensional partial-matching between the searched cases and the new case. Commonly, each attribute or dimension of a case has a determined importance value (weight), which is incorporated in the evaluation function. This weight could be static or dynamic depending on the Case-based reasoning agent purposes. Also, the evaluation function computes an absolute match score (a numeric value), although a relative match score between the set of retrieved cases and the new case can also be computed.

Most Case-based reasoners such as REMIND [Cognitive, 1992], MEDIATOR [Kolodner and Simpson, 1989], PERSUADER [Sycara, 1987], etc., use a generalised weighted distance

$$dist(C_i, C_j) = \sum_{k=1}^{n} w_k \times atr\_dist(C_{ik}, C_{jk})$$

function (NN) such as,

Some others measures have been defined in the literature, such as in [Leake et al., 1997; Osborne and Bridge, 1998; Sànchez-Marrè *et al*., 1998; Warren *et al*., 1998].

Two kinds of *impasses* could happen in a CBR agent:

- The situation is *unknown*, i.e. there is no memory about this situation or there is not a successful solution to this situation

- There are *several ways* (solutions) to proceed, i.e. there are several methods that may be applicable to a situation with the same degree of confidence.

For these impasse situations, CBR agents can use the expert general knowledge coded into the system or can generate an alarm that has to be solved by the user. Other approaches such as in NOOS [Arcos & Plaza, 1995] generate a reflexive task whose goal is to solve that impasse.

2.3.2 Case adaptation

When the best partial-matching case selected from the case library does not match perfectly with the new case, the old solution needs to be adapted to fit more accurately the new case solution. This reusing process can happen during the solution formulation (adaptation), or after some feedback has pointed out some problem in the evaluation step, which needs to be fixed (repair).
There are a lot of strategies that have been used in the Case-based reasoners. All these techniques can be grouped [Kolodner, 1993; Riesbeck and Schank, 1989] as *null adaptation*, *structural adaptation* and *derivational adaptation*, although in most Case-based reasoners, several mixture kinds of adaptation methods are implemented.

Null adaptation could be a right strategy in Case-based systems with very simple actions in the solution (like accept/reject, a fault diagnosis, etc.) such as the first adaptation method used in the PLEXUS system [Alterman, 1988]. In those systems, the old solution is applied directly to the new case.

There are several *structural adaptation* methods, where the adaptation process is directly applied to the solution stored in a case. The structural adaptation methods can be divided in three major techniques: substitution methods, transformation methods and special-purpose adaptation heuristics or critic-based adaptation methods.

- *Substitution methods* provide the solution of the new case with appropriate components or values computed from components or values in the retrieved solution. Most outstanding substitution techniques are: parameter adjustment or parameterised solutions, where the differences between the values of the retrieved case and those ones of the new case are used to guide the modification of the solution parameters in the appropriate direction. This approach has been used, for example, in HYPO [Ashley, 1990] and PERSUADER [Sycara, 1987], JUDGE [Bain, 1986]. Another kind of methods, such as direct reinstantiation used in CHEF [Hammond, 1989], local search used in JULIANA [Shinn, 1988], PLEXUS and SWALE [Kass and Leake, 1988], query memory used in CYRUS [Kolodner, 1985] and JULIANA, specialised search used in SWALE, etc., can be named as abstraction and respecialization methods. When there is a component as an object or avalue, etc, of the retrieved solution, that does not fit in the new problem, these methods look for abstractions of that component of the solution in a certain knowledge structure (concept generalisation tree, etc.) that do not have the same difficulty; the last kind of

substitution methods is the case-based substitution methods. They use the differences between the new and the retrieved case to search again cases from the case library to eliminate these differences. These techniques have been used, for instance, in systems such as CLAVIER [Hennessy and Hinkle, 1992], JULIA [Hinrichs, 1992], CELIA [Redmond, 1992], etc.

- The *transformation methods* uses either some common sense transformation rules such as deleting a component, adding a component, adjusting values of a component, etc.), as in JULIA system, or some model-guided repair transformation techniques based on a causal knowledge, such as in KRITIK [Goel and Chandrasekaran, 1992] or CASEY [Koton, 1989] systems.

- The *special-purpose adaptation* techniques or critic-based adaptation methods are based on some specific rules of repairing, called critics [Sacerdoti, 1977; Sussman, 1975], like those used in PERSUADER. Other systems such as CHEF and JULIA, use some domain specific adaptation heuristics and some structure modification heuristics.

Derivational adaptation methods do not operate on the original solutions, but on the method that was used to derive that solution. The goal is rerunning the same method applied to derive the old solution, to re-compute the solution for the new case. This methodology was first implemented in ARIES system, and was named as derivational replay [Carbonell, 1986]. In such a techniques, reinstantiation occurs when replacing a step in the derivation of the new solution, like in systems such as PRODIGY/ANALOGY [Veloso and Carbonell, 1993], JULIA [Hinrichs, 1992] or MEDIATOR [Kolodner and Simpson, 1989].

2.3.3 Case Evaluation

This step is one of the most important steps for a case-based reasoner. It gives the system a way to evaluate its decisions in the real world, allowing it to receive feedback that enables it to learn from success or failure.
Evaluation can be defined as the process of assessing the goodness or performance of the proposed solution for the new case derived from the solution of the best similar remembered case. The evaluation process can point out the need for additional adaptation –usually called repair– of the proposed solution, although this only make sense, in non real-time world domains. Commonly, this evaluation step can be performed either by asking to a human expert (oracle) whether the solution is a good one or not, or by simulating the effects of the proposed solution in the real world such as in most planning or design domains, or by directly getting a feedback on the results of the proposed solution, from the real world.

2.3.4 Case Learning

Learning is an interesting and essential cognitive task of the Case-based systems. Mainly, there are two major kinds of learning in a Case-based system: *learning by observation* and *learning by own experience*. Learning by observation happens when the system is provided with a set of initial cases, either by an expert or by direct observation (experientiation) of real data. Also, it can learn a new case by direct observation provided by an expert in any moment.

2.3.4.1 Learning by observation

It is important to remark that a Case-based reasoner starts with a representative set of cases. They are like the training set of other supervised machine learning methods. To this end, the initial Case Library is usually seeded with some situations obtained by classification procedures of historical databases. See for example [Sànchez-Marrè *et al*., 1997b].
From these new discovered classes, some objects (cases) belonging to each class are selected to be included in the initial Case Library.

2.3.4.2 Learning by experience

Learning by own experience is being done after each cycle of the Case-based reasoner. After an evaluation step appears the opportunity to increase the problem solving capabilities of the system. So, it can learn from the new experience. If the proposed solution has been a successful one, the system can learn from this fact, in the sense that if this experience is stored in memory, when a new similar case to this one appears, it can be solved as the past one (learning from success). If the system has failed, it must be able to prevent itself from making the same mistake in the future (learning from failure). Not all the Case-based systems have both kinds of learning.

Learning from success

When the new case has been successfully solved, the main option to follow is to store this new case in the case library. So, this task means to insert the new experience in the appropriate place into the case library, so that it can be remembered when it can be more useful, and cannot be recalled indiscriminately. In other words, the case must be placed in the neighbourhood space of the memory where it would be easily recalled in the retrieval step. Thus, good indexes must be chosen to implement this strategy. While the new case is being placed in the case library, memory's indexing structure and organisation is updated appropriately.

Learning from failure

Assuming that the adaptation method is correct, two reasons could originate a failure. One is that the case retrieved is the best one for solving this new situation, given the current case

library, although it is not very similar. The problem here is that there are not enough cases (experience) in the case library to cover the whole space of cases. The solution relies on learning a lot of new relevant experiences to store in the case library. The other reason is that although it exists a very similar case to the new one, it has not been retrieved. Thus, there is something wrong in the retrieval process. Perhaps the distance function is not correct because the strategic importance of attributes (weight) is not well suited or the function do not capture accurately the differences between qualitative and quantitative values of attributes. Perhaps what is wrong is the discrimination tree organisation. If the discriminating list of attributes provided by the experts is not good, then the retrieval algorithm can miss the best similar case due to the fact that it will be searching in some other region of the tree, where the best similar case is not there. The solution to this problem relies in re-organising the case library's structure [Veloso & Carbonell, 1993].

When the new case has failed, there are several possible actions to be taken, in order to ensure that this failure cannot be repeated in future. First, the case-based system can store the failed case into its memory to prevent taking, another time, the same failed solution for similar cases to this one. Some case-based systems maintain a separate case library of failed cases [Hammond, 1989], and others maintain only one case library structure.

In the first kind of systems, a previous step is added to the general case-based reasoning cycle: the *anticipation phase*. Before retrieving any successful case in memory, it is recalled whatever case in the failed case library that matches the input new case to avoid repeating the failure. In the other systems, the equivalent of that anticipation step is implemented as a filtering task applied to the searched cases in the retrieving process. They eliminate the previous failed cases and the cases that were the source experience to derive those failed cases, from the list of retrieved cases. Thus, the system can avoid making the same incorrect action that in past situations.

Another interesting action to do when there is an available human expert, is incorporate to the system's memory the right solution that he proposes to solve the new situation. So, in the future, this recorded experience could be remembered and used appropriately.

Other tasks that can be performed are updating the weight of the attributes, and thus, modifying the distance function such as in [Bareiss, 1989; Koton, 1989], or changing their order in the discriminating list. Another feature is to update the utility measure of the retrieved cases that could derive the new case.

1.4 Applications of Case-Based Reasoning

CBR systems have been used in a broad range of domains to capture and organise past experience and to learn how to solve new situations from previous past solutions. CBR systems have been applied to planning (PRECEDENTS [Oxman & Voß, 1996],

CAPLAN/CBC [Veloso et al. ,1996], CHEF [Hammond, 1989]), design (NIRMANI [Perera & Watson, 1996], JULIA [Hinrichs, 1992]), classification (PROTOS [Bareiss, 1989]), diagnosis (CASEY [Koton, 1989]), understanding and analysis (AQUA [Ram, 1993; Ram & Hunter, 1992]), interpretation (HYPO [Ashley, 1990]), troubleshooting detection (CASIOPÉE, LADI [Lenz et al., 1996]) and explanation (SWALE [Kass & Leake, 1988]). For Case-based Reasoning in continuous situations there are CIDA [Joh, 1997], an assistant for conceptual internetwork design, and NETTRAC [Brandau et al., 1991] as a case-based system for planning and execution monitoring in traffic management in public telephone networks.

CBR provides an adequate framework to cope with *continuous* domains, where a great amount of new valuable experiences are generated in a non-stop way. In Environmental Sciences, CBR has been applied in different areas with different goals, because of its general applicability:

- In information retrieval from large historical meteorological databases [Jones and Roydhouse, 1995].
- In optimisation of sequence operations for the design of wastewater treatment systems [Krovvidy and Wee, 1993]
- In supervisory systems for diagnosing and controlling WWTP systems [Sànchez-Marrè *et al*., 1999; Sànchez-Marrè *et al*., 1997a].
- In decision support systems for planning forest fire fighting [Avesani *et al*., 2000].
- In case-based prediction for rangeland pest management advisories [Branting et al., 1997].
- In case-based design for process engineering [Surma and Brauschweig, 1996].

References

[Aamodt & Plaza, 1994] A. Aamodt and E. Plaza. Case-based reasoning: fundamental is-sues, methodological variations and system approaches. AI Communications 7(1):39-59, 1994.

[Alterman, 1988] R. Alterman. Adaptive planning. Cognitive Science 12:393-422, 1988.

[Althoff & Aamodt, 1996] K. D. Althoff and A. Aamodt. Relating case-based problem solving and learning methods to task and domain characteristics: towards an analytic framework. AI Communications 9(3):109-116, 1996.

[Arcos & Plaza, 1995] J.L. Arcos and E. Plaza. Reflection in NOOS: an Object-centered representation language for knowledge modelling. In IJCAI Workshop on Reflection and Meta-level architectures and their application in AI (IJCAI'95), pages 1-10, Montréal, 1995.

[Ashley, 1990] K.D. Ashley. Modelling legal argument: reasoning with cases and hypotheticals. The MIT Press, 1990.

[Avesani et al., 2000] P. Avesani, F. Ricci and A. Perini. Interactive Case-based planning for forest fire fighting. Applied Intelligence 13(1):41-58, 2000.

[Bain, 1986] W. Bain. Case-based reasoning: a computer model of subjective assessment. Ph. D. Dissertation. Dept. of Computer Science. Yale University, 1986.

[Bareiss, 1989] E.R. Bareiss. Exemplar-based knowledge acquisition: a unified approach to concept representation, classification and learning. Academic Press, 1989.

[Brandau et al., 1991] R. Brandau, A. Lemmon and C. Lafond. Experience with extended episodes: cases with complex temporal structure. In Proc. of Workshop on case-based reasoning (DARPA). Washington D.C., 1991.

[Branting et al., 1997] Branting. L.K., Hastings, J.D., and Lockwood, J.A. Integrating cases and models for prediction in biological systems. AI Applications 11(1):29-48, 1997.

[Carbonell, 1986] J. Carbonell. Derivational analogy: a theory of reconstructive problem solving and expertise acquisition. Machine Learning vol. 2, 1986.

[Cognitive, 1992] Cognitive Systems. ReMind Developer's Reference Manual. Boston, 1992.

[Fox & Leake, 1995] S. Fox and D.B. Leake. Using introspective reasoning to refine indexing. Proc. of 14th Int. Joint Conference on Artificial Intelligence (IJCAI'95), pp. 391-397. Montréal, 1995.

[Goel & Chandrasekaran, 1992] A. Goel and B. Chandrasekaran, Case-based design: a task analysis. In Artificial Intelligence approaches to Engineering design, vol. 2: Innovative design (C. Tong and D. Sriram, editors). Academic Press, 1992.

[Hammond, 1989] K. Hammond. Case-based planning: viewing planning as a memory task. Academic Press, 1989.

[Hennessy & Hinkle, 1992] D.H. Hennessy and D. Hinkle. Applying case-based reasoning to autoclave loading. IEEE Expert 7(5):21-26, 1992.

[Hinrichs, 1992] T.R. Hinrichs. Problem solving in open worlds: a case study in design. Lawrence Erlbaum, 1992.

[Joh, 1997] D.Y. Joh. CBR in a changing environment. Procc. of 2nd Int. Conf. On Case-based Reasoning (ICCBR'97). LNAI-1266, pp. 53-62, 1997.

[Jones & Roydhouse, 1995] Jones, E. and Roydhouse, A. Retrieving structured spatial information from large databases: a progress report .Procc. of IJCAI Workshop on Artificial Intelligence and the Environment, pp. 49-57, Montréal, 1995.

[Kass & Leake, 1988] A.M. Kass and D.B. Leake. Case-based reasoning applied to constructing explanations. Proc. of Workshop on case-based reasoning (DARPA). Clearwater, Florida. 1988.

[Kolodner, 1993] J.L. Kolodner. Case-Based Reasoning. Morgan Kaufmann, 1993.

[Kolodner, 1985] J.L. Kolodner. Memory for experience. In The psychology of learning and motivation vol. 19 (G. Bower editor). Academic Press, 1985.

[Kolodner & Simpson, 1989] J.L. Kolodner and R.L. Simpson. The MEDIATOR: analysis of an early case-based problem solver. Cognitive Science 13(4):507-549, 1989.

[Koton, 1989] P. Koton. Using experience in learning and problem solving. Ph. D. dissertation. Dept. of Computer Science. MIT, 1989.

[Krovvidy & Wee, 1993] S. Krovvidy and W.G. Wee. Wastewater Treatment Systems from Case-Based Reasoning. Machine Learning 10, pp. 341-363, 1993.

[Leake et al., 1997] D.B. Leake, A. Kinley and D. Wilson. Case-based similarity assesment: estimating adaptability from experience. Proc. of American Association of Artificial Intelligence (AAAI-97), pp. 674-679, 1997.

[Lenz et al. ,1996] M. Lenz, H-D. Burkhard, P. Pirk, E. Auriol and M. Manago. CBR for diag-nosis and decision support. AI Communications 9(3):138-146, 1996.

[Miyashita & Sycara, 1995] K. Miyashita and K. Sycara. Improving system performance in case-based iterative optimization through knowledge filtering. Proc. of 14th Int. Joint Conference on Artificial Intelligence (IJCAI'95), pp. 371-376. Montréal, 1995.

[Osborne and Bridge, 1998] H. R. Osborne and D. G. Bridge. A case base similarity framework. Proc. of 4th European Workshop on Case-Based Reasoning (EWCBR'98), pp. 309-323, 1998.

[Oxman & Voß, 1996] R. Oxman and A. Voß. CBR in design. AI Communications 9(3):117-127, 1996.

[Perera & Watson, 1996] S. Perera and I. Watson. NIRMANI: an integrated case-based system for strategic design and estimating. Progress in Case-Based Reasoning. LNAI #1020. Pp 186-200, 1996.

[Ram, 1993] A. Ram. Indexing, elaboration and refinement: incremental learning of explanatory cases. Machine Learning 10(3):201-248, 1993.

[Ram & Hunter, 1992] A. Ram and L. Hunter. Goals for learning and understanding. Applied Intelligence 2(1):47-73, 1992.

[Redmond, 1992] M.A. Redmond. Learning by observing and understanding expert problem solving. Georgia Institute of Technology. College of Computing. Technical report GIT-CC-92/43, 1992.

[Riesbeck & Schank, 1989] C.K. Riesbeck and R.C. Schank. Inside Case-Based Reasoning. Lawrence Erlbaum Associates Publishers, 1989.

[Sacerdoti, 1977] E.D. Sacerdoti. A structure for plans and behavior. North-Holland, 1977.

[Sànchez-Marrè et al., 1999] M. Sànchez-Marrè, U. Cortés, I. R-Roda, and M. Poch. Sustainable case learning for continuous domains. Environmental Modelling & Software 14(5):349-358, 1999.

[Sànchez-Marrè et al., 1998] M. Sànchez-Marrè, U. Cortés, I. R.-Roda, and M. Poch. L'Eixample distance: a new similarity measure for case retrieval. Procc. of 1st Catalan Conference on Artificial Intelligence (CCIA'98), pp. 246-253.Tarragona, Catalonia, EU, 1998.

[Sànchez-Marrè et al., 1997a] M. Sànchez-Marrè, U. Cortés, I. R.-Roda, M. Poch, and J. Lafuente. Learning and Adaptation in WWTP through Case-Based Reasoning. Special issue on Machine Learning of Microcomputers in Civil Engineering 12(4):251-266. July, 1997.

[Sànchez-Marrè et al., 1997b] M. Sànchez-Marrè, U. Cortés, J. Béjar, J. De Gràcia, J. Lafuente and M. Poch. Concept Formation in WWTP by means of Classification Techniques: a Compared Study. Applied Intelligence 7(2):147-166. April, 1997.

[Schank, 1982] R. Schank. Dynamic memory: a theory of learning in computers and people. Cambridge University Press, 1982.

[Schank & Abelson, 1977] R. Schank and R. Abelson. Scripts, plans, goals and understanding. Lawrence Erlbaum, 1977.

[Shinn, 1988] H.S. Shinn. Abstractional analogy: a model of analogical reasoning. Proc. of Workshop on case-based reasoning (DARPA). Clearwater, Florida. 1988.

[Smyth & Keane, 1995] B. Smyth and M.T. Keane. Remembering to forget. Proc. of 14th Int. Joint Conference on Artificial Intelligence (IJCAI'95), pp. 377-382. Montréal, 1995.

[Steels, 1990] L. Steels. Components of expertise. AI Magazine 11(2):28-49, 1990.

[Surma & Brauschweig, 1996] J. Surma and B. Brauschweig. Case-Based Retrieval in Process Engineering: Supporting Design by Reusing Flowsheets. Engineering Applications of Artificial Intelligence. 9(4): 385-391, 1996.

[Sussman, 1975] G.J. Sussman. A computer model of skill acquisition. American Elsevier, 1975.

[Sycara, 1987] K. Sycara. Finding creative solutions in adversarial impasses. Proc. of 9th Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, 1987.

[Veloso & Carbonell, 1993] M.M. Veloso and J.G. Carbonell. Derivational Analogy in PRODIGY: automating case acquisition, storage and utilization. Machine Learning 10(3):249-278, 1993.

[Veloso et al., 1996] M. Veloso, H. Muñoz-Avila and R. Bergmann. Case-based planning: selected methods and systems. AI Communications 9(3):128-137, 1996.

[Warren et al., 1998] T. Warren Liao, Z. Zhang and C.R. Mount. Similarity measures for retrieval in case-based reasoning systems. Applied Artificial Intelligence 12:267-288, 1998.

[Watson, 1996] I. Watson. An introduction to Case-based reasoning. Progress in Case-Based Reasoning. LNAI #1020. Pp 3-16, 1996.