Constructing autonomous distributed systems using CBR-BDI agents

M. G. Bedia and J. M. Corchado

Departamento de Informática y Automática
University of Salamanca
Plaza de la Merced s/n, 37008, Salamanca, Spain
Email: corchado@usal.es
http://tejo.usal.es/~corchado/

Abstract: This chapter introduces a robust mathematical formalism for the definition of deliberative agents implemented using a case-based reasoning system. The concept behind deliberative agents is introduced and the case-based reasoning model is described using this analytical formalism. Variational calculus is introduced in this chapter to facilitate to the agents the planning and replanning of their intentions in execution time, so they can react to environmental changes in real time. A variational calculus based planner for constructing deliberative agents is the presented and compared with other planners. Reflecting the continuous development in the tourism industry as it adapts to new technology, the chapter includes the formalisation of an agent developed to assist potential tourists in the organisation of their holidays and to enable them to modify their schedules on the move using wireless communication systems.

1 Introduction

Technological evolution in today's world is fast and constant. Successful knowledge engineering systems should have the capacity to adapt at and should be provided with mechanisms that allow them to decide what to do according to their objectives. Such systems are known as autonomous or intelligent agents [28]. This chapter shows how a deliberative agent with a BDI (Believe, Desire and Intention) architecture can use a case-based reasoning (CBR) system to generate its plans. A robust analytical notation is introduced to facilitate the definition and integration of BDI agents with CBR systems. The chapter also shows how variational calculus can be used to automate the planning and replanning process of such agents in execution time.

Agents should be autonomous, reactive, pro-active, sociable and have learning capacity. They must be able to answer to events that take place in their environment, take the initiative according to their goals, interact with other agents (even human) and use past experiences to achieve present goals. There are different types of agents and they can be classified in different ways [28]. One type, the so-called deliberative agent with BDI - Belief, Desire and Intention - architecture, uses the three attitudes in order to make decisions on what to do and how to get it [16, 28]: their beliefs represent their information state - what the agents know about themselves and their environment; their desires are their motivation state - what they are trying to attain; and the intentions represent the agents' deliberative state. Intentions are sequences (ordered sets) of beliefs (also identified as plans). These mental attitudes determine the agent's behavior and are critical if a proper performance is to be produced when information about a problem is scarce [2,17]. BDI architecture has the advantage that it is intuitive - it is relatively easy to recognize the process of decision-making and how to perform it. Moreover, it is easy to understand the notions of belief, desires and intentions. On the other hand, its main drawback lies in determining a mechanism, which will allow its effective implementation. The formalization and implementation of BDI agents constitutes the research of many scientists [9, 16, 24]. Some of these researchers criticize the necessity of studying multi-modal logic for the formalization and construction of such agents, because they haven't been completely acclimatized and they aren't computationally efficient. Rao and Georgeff [22] assert that the problem lies in the great difference between the powerful logic of BDI systems and with that required by practical systems. Another problem is that these types of agents don't have learning capacity - a necessary element for them since they have to be constantly adding, modifying or eliminating beliefs, desires and intentions.

This chapter presents a robust analytical formalization for the definition of computationally efficient agents, which solves the first of the previously mentioned problems. This chapter also shows how a BDI agent implemented using a case-based reasoning (CBR) system can substantially solve the problems related to the learning capability of the agents. Implementing agents in the form of CBR systems facilitate their learning and adaptation. If the proper correspondence between the three mental attitudes of the BDI agents and the information that a case-based reasoning system manipulates can be established, an agent will be created not only with beliefs, desires, intention but also with learning capacity.

Although the relationship between agents and CBR systems have been investigated by other researchers [19, 21, 27], we propose a robust mathematical formalism, that will facilitate the efficient implementation of an agent in the form of a CBR system. Variational calculus is introduced to automate the reasoning cycle of the BDI agents, it is used during the retrieval stage of the CBR cycle to guaranty an efficient planning and replanning in execution time. Although different types of planning mechanism can be found in the literature [3, 11, 15, 18, 25], none of them allow the replanning in execution time, and agents inhabit changing environment in which replanning in execution time is required if goals have to be achieved successfully in real-time. Some of the approaches developed use planning techniques to select the appropriate solution to a given problem but without mechanisms to deal with the changes on the environment. For instance, in [11, 18] it is introduces a kind of plan schemas that need to be reprogrammed overtime, when the planning domain changes. In [4, 7, 8] it is proposed an architecture that tries to be more flexible by using planning strategies to create the plans. If new information must to be introduced from the environment to the system, it is only necessary to change the planning domain instead of reprogramming the plan schema by hand. This architecture allows building plans that contain steps with no detailed information. This is useful because if no specific information is supplied, the solution can handle planning generic operators, plans that are not influenced by unexpected changes. Now to know if the abstract proposed plan is adequate it is required to put it into practice in a real domain.

This operation requires a high amount of computational time and resources which may be a disadvantage, in for example, web related problems. The flexibility of this approach increases the time spent in applying the abstract solution to the real problem, which is a handicap for real time systems. In this chapter it is proposed a solution, a variational calculus based planner (VCBP) that deals adequately with environmental real-time problem changes without applying a reprogramming strategy and without the disadvantages shown in [4, 7, 8] because the technique used can solve a planning problem in execution time. This is achieved using variational calculus during the retrieval stage of the CBR life cycle. To begin with, the chapter will review the concepts of CBR system and deliberative agent using an analytical notation. Then it will be shown how a CBR system is used to operate the mental attitudes of a deliberative BDI agent. This section also shows the relationship between BDI agents and CBR systems. Then variational calculus will be introduced, and will be shown how it can be used to define agents with the previously mentioned characteristics.

2 Case-based reasoning systems

Case-based reasoning is used to solve new problems by adapting solutions that were used to solve previous similar problems [1, 10, 26]. The operation of a CBR system involves the adaptation of old solutions to match new experiences, using past cases to explain new situations, using previous experience to formulate new solutions, or reasoning from precedents to interpret a similar situation. A case is the basic knowledge representation structure. Figure 1 shows the reasoning cycle of a typical CBR system that includes four steps that are cyclically carried out in a sequenced way: retrieve, reuse, revise, and retain [1, 26].

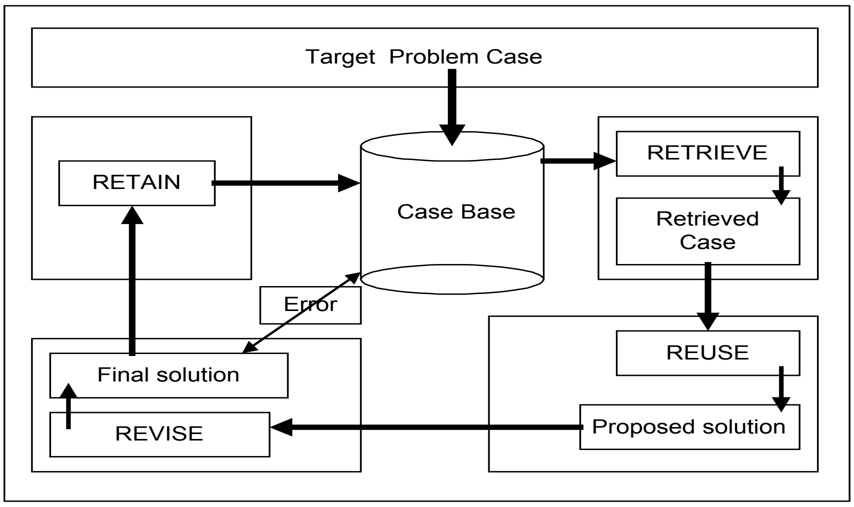

Fig. 1: CBR Cycle of Life

During the retrieval phase, those cases that are most similar to the problem case are recovered from the case-base. The recovered cases are adapted to generate a possible solution during the reuse stage. The solution is then reviewed and, if appropriate, a new case is created and stored during the retention stage, within the memory.

Therefore CBR systems update (with every retention step) their case-bases and consequently evolve with their environment. Each of the reasoning steps of a CBR system can be automated, which implies that the whole reasoning process could be automated to a certain extend [10, 13]. This assumption has led us to the hypothesis that agents implemented using CBR systems could be able to reason autonomously and therefore to adapt themselves to environmental changes. Agents may then use the reasoning cycle of CBR systems to generate their plans.

Based on the automation capabilities of CBR systems we have established a relationship between cases, the CBR life cycle, and the mental attitudes of the BDI agents. Based on this idea, a model is presented that facilitates the implementation of the BDI agents using the reasoning cycle of a CBR system.

3 Implementing Deliberative Agents using CBR Systems

This section identifies the relationships that can be established between BDI agents and CBR systems, and shows how an agent can reason with the help of a case-based reasoning system. The formalization presented in this chapter takes elements of other knowledge engineering systems [11, 19], and adapts them to the model presented here. Our proposal attempts to define a direct mapping between the agents and the reasoning model, paying special attention to two characteristics: (i) the mapping between the agents and the reasoning model should allow a direct implementation of the agent and (ii) the final agents should be capable of learning and adapting to environmental changes. An analytical notation has been introduced to facilitate an efficient integration between the BDI agent and CBR system and that allows the use of variational calculus for the planning and replanning in execution time. The notation used in the refereed works [19, 21, 27] do not have the required degree of expressivity and complexity to introduce differential calculus tools.

3.1 BDI Agents

The notation and the relationship between the components that characterise a BDI agent are first introduced.

We denote Θ as the set that describes the agent environment, and $T(\Theta)$ as the set of attributes $\{\tau_1, \tau_2, \ldots, \tau_n\}$ in which the world is expressed.

Definition 1. A belief *e* on Θ is a m-upla of attributes of $T(\Theta)$ denoted as:
$e=(\tau_1, \tau_2, \ldots, \tau_m)$ with $m \leq n$. With these notation, we call set of beliefs on Θ and denote $\zeta(\Theta)=\{(\tau_1, \tau_2, \ldots, \tau_j)$ where $j = (1,2,\ldots, m \leq n)\}$.

Example. It is supposed that the world T(Θ) includes all the attributes that are needed to characterise a tourist schedule in the city of Salamanca (European city of the Culture 2002), and we denote *T(Θ)=monument name, τ_2= time table, τ_3= cost,..., τ_n}*

In this world, a belief, for example, monument, is represented by a m-upla of attributes of T(Θ) that characterise the monument and we denote:

Monument belief =(τ_1=index, τ_2=time table, τ_3=visiting cost, τ_4=quality)

A particular belief, for example, the old cathedral (OC), can be represented by:

Old Cathedral=(idx=OC, time table є (10:00, 18:00), visiting cost=3 €, quality=1.5)

Definition 2. The operator " Λ of accessibility" is introduced between m believes ($e_1,e_2,e_3,\ldots,e_m$) where $\Lambda(e_1, e_2, e_3,\ldots,e_m) = (e_1 \wedge e_2 \wedge \ldots \wedge e_m)$ denotes that exists compatibility among the set of believes ($e_1, e_2, e_3,\ldots, e_m$). If any of the believes is not accessible, it will be denoted by $\Lambda(e_1, e_2, e_3,\ldots,e_m) = 0$.

Example. It is 12:00h. p.m. and the agent believes M1, M2 and D(A,A), which are described bellow (see Table 1). With these believes and given that it is 12:00 h, it is impossible to visit M1 and M2, and therefore the path (M1 $\wedge$ D(A,A) $\wedge$ M2) can not be constructed and Λ (M1,D(A,A), M2)=0.

Table 1. Values of believes M1, M2 and D(A,A)

Attribute	Value
Entity	M1
Class	monument
Time open	10-13 hrs.
Costs	6 €
Time for a visit	1 hr.
Zone or place	A

Attribute	Value
Entity	M2
Class	monument
Time open	10-14 hrs.
Costs	6 €
Time for a visit	1 hr.
Zone or place	A

Attribute	value
Entity	D(A,A)
Class	travel by taxi
Time	1 hr.
Costs	12 €

Table 2. Values of belief M3

Attribute	Value
Entity	M2
Class	monument
Time open	10-14 hrs.
Costs	6 €
Time for a visit	1 hr.
Zone or place	A

If M2 is substituted by M3 (see Table 2) then (M1 $\wedge$ D(A,A) $\wedge$ M3) is possible, and Λ (M1,D(A,A),M2)$\neq$ 0, which means that the agent has identified that we can visit the monument M1 and M3, taking into consideration that the time to go from the first to the second monument is given by D(A,A).

Definition 3. An intention i on Θ is a s-upla $i=(e_1, e_2,...,e_s)$ of compatible beliefs, with Λ (ei, ej)≠0. We denote $I(\Theta)=\{ i=(e_1, e_2, ..., e_k) \}$ the set of intentions on Θ.

Now a set of parameters will be associated to the space $I(\Theta)$ that characterizes any element of that set. The set of necessary and sufficient variables to describe the system may be obtained experimentally.

Definition 4. We call canonical variables of a set $I(\Theta)$ to any set of linearly independent parameters $\aleph=(A_1, A_2,..., A_v)$ that characterize the elements $i\in I(\Theta)$.

Example. If the agent identifies a visiting route through the number of monument to visit (P) and a maximum associated cost (C), and we express it as $\aleph=(A_1,A_2)=(P, C)$.

In this coordenates system the intention i_1 is expressed as:
$i_1=M1 \wedge D(A,A) \wedge M2 \wedge D(A,A) \wedge R1 \wedge D(A,B) \wedge M3 \wedge D(B,B) \wedge R2$
with the values represented in Table 3. It also has the values for P and C indicated.

Table 3. Values of the believes that constitute intention i_1 and values for (P,C) associated

Schedule (hr)	10-11	11-12	12-13	13-14	14-16	16-18	18-20	20-21	21-22
intention	M1	D(A,A)	M2	D(A,A)	R1	D(A,B)	M3	D(B,B)	R2
Costs(€)	6	0	6	0	12	0	0	0	12
Time (hr)	1	1	1	1	2	2	2	1	1
Quality	1	--	2	--	1	--	2	--	2

Attributes	
P	3
C(€)	36

Definition 5. A desire d on Θ is a mapping between

$$\begin{aligned} d : I(\Theta) &\longrightarrow \Omega(\aleph) \\ i=(e_1 \wedge \ldots \wedge e_r) &\rightarrow F(A_1, A_2, ..., A_v) \end{aligned}$$

where $\Omega(\aleph)$ is the set of mappings on $\aleph$. A desire d may be achieved constructing an intention i using some of the available beliefs, whose output could be evaluated in terms of the desired goals. We call set of desires on Θ and denote $D(\Theta)$ to the set:
$D(\Theta)=\{d: I(\Theta)\rightarrow \Omega(\aleph)$/ with $I(\Theta)$, set of intentions and $\Omega(\aleph)$ set of mappings on $\aleph\}$

Example. The desire function of a tourist "I want to visit at least three monuments and spend less than 50€", may be expressed as:

$$F(A_1, A_2) = F(P,C) = \left\{ \begin{matrix} P \geq 3 \\ C \leq 50 \end{matrix} \quad \text{with} \quad \begin{matrix} P \in (0,10) \\ C \in (0,100) \end{matrix} \right\}$$

Now, after presenting our definition of the agent's belief, desire and intention, section 3.2 defines the proposed analytical formalism for the CBR system.

3.2 Analytical formalism of Case-based Reasoning systems

The necessary notation to characterise a CBR system is introduced as follows. Let us consider a problem P, for which it is desired to obtain the solution S(P). The goal of a case-based reasoning system is to associate a solution S(P) to a new problem P, by reusing the solution S(P′) of a memorised problem P′.

A problem P is denoted as $P=(S_i,\{\theta_j\}, S_f)$ with S_i=initial state, S_f=final state and $j=(1,\dots,m)$. Its solution S(P) is defined as $S(P)=\{S_k, \theta_h\}$ where $k=(1,..,n+1)$ and $hl=(1,..,n \le m)$, $S_1=S_i$ and $S_{n+1}= S_f$.

Definition 6. The state S_k and the operator θ_j are defined as

$$S_k=\begin{pmatrix}\{O_r\}_{r=1,\dots,p}\\ \{R_s\}_{s=1,\dots,q}\end{pmatrix} \qquad \theta_j : S_k=\begin{pmatrix}\{O_r\}\\ \{R_s\}\end{pmatrix} \longrightarrow \theta_j(S_k)=\begin{pmatrix}\{O'_r\}\\ \{R'_s\}\end{pmatrix}$$

where $\{O_r\}_{r=1,\dots,p}$ and $\{R_s\}_{s=1,\dots,q}$ are coordinates in which a state S_k is expressed.

The coordinates type $\{O_r\}_{r=1,\dots,p}$ are introduced to express the objectives achieved and the coordinates type $\{R_s\}_{s=1,\dots,q}$ are introduced to express the resources lost.

Through these definitions, the parameter effectiveness, $\Im$, between two states S and S' can be defined, as a vector $\Im(S, S') = (\Im_x, \Im_y)$ which takes the form

$$\Im_x = \frac{O_r(S') - O_r(S)}{O_r \max} \qquad \Im_y = \frac{R_s(S) - R_s(S')}{R_s \max}$$

where the definition implies that ($0\le\Im_x\le1$) and ($0\le\Im_y\le1$).

In order to evaluate the rate of objectives achieved and resources used, between S and S', it is necessary to normalise every component of $\{O_r\}_{r=(1,\dots,p)}$, $\{R_s\}_{s=(1,\dots,q)}$

$$\Im_x = \frac{\sqrt{\left(\frac{O'_1-O_1}{\max O_1}\right)^2+\left(\frac{O'_2-O_2}{\max O_2}\right)^2+\dots+\left(\frac{O'_p-O_p}{\max O_p}\right)^2}}{\sqrt{\left(\frac{\max O_1}{\max O_1}\right)^2+\left(\frac{\max O_2}{\max O_2}\right)^2+\dots+\left(\frac{\max O_p}{\max O_p}\right)^2}} \qquad \Im_y = \frac{\sqrt{\left(\frac{R_1-R'_1}{\max R_1}\right)^2+\left(\frac{R_2-R'_2}{\max R_2}\right)^2+\dots+\left(\frac{R_q-R'_q}{\max R_q}\right)^2}}{\sqrt{\left(\frac{\max R_1}{\max R_1}\right)^2+\left(\frac{\max R_2}{\max R_2}\right)^2+\dots+\left(\frac{\max R_q}{\max R_q}\right)^2}}$$

Definition 7. A new parameter is also introduced - efficiency - that measures how many resources are needed to achieve an objective. Given a target problem P, and a solution S(P), let us say that the efficiency of the solution S(P), is obtained by: $\zeta[S(P)]= \Im_x / \Im_y$. The definition implies that $\zeta(S,S') \in (0, \infty)$

Definition 8. A case C is a 3-upla {P, S(P), $\Im$[S(P)]} where P is a problem description, S(P) the solution of P and $\Im$[S(P)] the effectiveness parameter of the solution. Then we can define a CBR′s case base as a finite set of cases memorized by the system denoted by $CB=\{C_k / k=(1,...,q)\}$.

3.3 Integration of the CBR system within the BDI Agent

The relationship between CBR systems and BDI agents can be established, associating the beliefs, desires and intentions with cases. Using this relationship we can implement agents (conceptual level) using CBR systems (implementation level).

So once the beliefs, desires and intentions of an agent are identified, they can be mapped onto a CBR system.

First, a mapping is introduced that associates an index to a given case Ck.

$$
\begin{aligned}
&idx: CB \rightarrow I(BC)\\
&C \rightarrow idx(C)= idx\{P, S(P), \Im[S(P)]\}= \{ idx\,(S_I), idx\,(S_F) \}=\\
&\qquad = \{ [S_I=(O_1,a_1), (O_2,a_2),\ldots,(O_p, a_p), (R_1,b_1), (R_2,b_2),\ldots, (R_q, b_q)],\\
&\qquad\quad [S_F=(O'_1,c_1),(O'_2,c_2),..,(O'_p, c_p),(R'_1,d_1),(R'_2,d_2),..,(R'_q,d_q)] \}.
\end{aligned}
$$

where the set I(BC) is the set of indices of a case base CB that is represented by frames composed of conjunction of attributes Oj, $R_k \in$ T(BC) and values a_i, b_j, c_k, d_l of the domain.

The abstraction realized through the indexing process allows the introduction of an order relation R in the CB that can be used to compare cases. Indices are organized in the form of a Subsumption Hierarchy.

$$(CB, R)=\{ [C_k / k=(1,..,q) \text{ and } q \in IN], R\}=\{ (C_1,..,C_q)/ idx(C_1) \subseteq \ldots \subseteq idx(C_q) \}$$

Definition 9. Let us say that two cases C and C'$\in$CB fulfill the relation

$$idx(C) \subseteq idx(C') \quad \text{if} \quad \begin{matrix} idx\,(S_I) \subseteq idx\,(S'_I) \\ idx\,(S_F) \supseteq idx\,(S'_F) \end{matrix}$$

If it is expressed in terms of their components,

$$idx\,(S_I) \subseteq idx\,(S'_I) \rightarrow \left\{ \begin{matrix} O_r(S_I) \geq O_r(S'_I) & \forall r=1,\ldots p \\ R_s(S_I) \leq R_s(S'_I) & \forall s=1,\ldots q \end{matrix} \right\}$$

$$idx(S_F) \supseteq idx(S'_F) \rightarrow \left\{ \begin{matrix} O_r(S_F) \leq O_r(S'_F) & \forall r=1,\ldots,p \\ R_s(S_F) \geq R_s(S'_F) & \forall s=1,\ldots,q \end{matrix} \right\}$$

Then, we say that S(P') is a possible CBR solution of the target P, if

$$\forall C'= (P' , S(P'), \Im[S(P')]) / idx(C') \supseteq P$$

Example. Given three cases, C1, C2, C3, and their indexes, where the initial states are null:

idx(C1)={(O'_1,a_1), (R'_1,b_1), (R'_2, b_2)}= {$(O'_1,1.7)$, $(R'_1,95)$, $(R'_2, 21.6)$}
idx(C2)={(O'_1,a_1), (R'_1,b_1), (R'_2, b_2)}= {$(O'_1,1.1)$, $(R'_1,80)$, $(R'_2, 19.2)$}
idx(C3)={(O'_1,a_1), (R'_1,b_1), (R'_2, b_2)}= {$(O'_1,0.9)$, $(R'_1,100)$, $(R'_2, 22)$}

If the problem to solve may be represented by P=(S_I, S_F) where its solution satisfy,

$$S(P) = \left\{ \begin{array}{l} O'_1 \geq 1 \\ R'_1 \leq 105 \ , \ \ R'_2 \leq 25 \end{array} \right\}$$

the relationship idx(C3)$\subseteq$ P$\subseteq$ idx(C1), idx(C2) may be established. So the definitions presented above let us know that idx(C3) is not a possible CBR solution of the target P, while idx(C1), idx(C2) are possible CBR solution for the problem P.

Definition 10. We use state ς of an intentional process $\{e_1 \wedge e_2 \wedge \ldots \wedge e_{s-1} \wedge e_s\}$ to describe any of the situations intermediate to the solution $\{e_1 \wedge e_2 \wedge \ldots \wedge$ er, with $r \leq s\}$ that admit a representation over $\aleph=(A_1, A_2, \ldots, A_v)$.

Definition 11. Given a canonical coordenates system $(A_1, A_2, \ldots, A_v)$ in $I(\Theta)$, the set may be reordered, and we denote $\aleph = \{F_m\} \cup \{G_n\}$ and $m+n=v$, where:
$\{F_m\} = \{A_{j \text{ with } j \leq v} / A_j$ growing$\}$ and $\{G_n\} = \{A_{k \text{ with } k \leq v} / A_k$ decreasing$\}$.

Giving an $i \in I(\Theta)$, a functional dependence relationship may be obtained in terms of the attributes, $i=i(\tau_1, \tau_2, \ldots, \tau_n)$, and in terms of its canonical or state variables, $i=i(A_1, A_2, \ldots, A_v) = i(F_1, F_2, \ldots, F_m, G_1, G_2, \ldots, G_n)$ which determines a functional relationship of the type $A_j = A_j(\tau_1, \tau_2, \ldots, \tau_n)$

Now the fundamental relationship between the BDI agents and the CBR systems can be introduced.

The solution S(P) for a given problem $P=(S_I, \{\theta_j\}, S_F)$ can be seen as a sequence of states $S_k=(\{O_r\}_{r=1,\ldots,p}, (\{R_s\}_{s=1,\ldots,q})$ interrelated by operators $\{S_k, \theta_h\}$.

Given a BDI agent over Θ with a canonical system, $\aleph=(A_1, A_2, \ldots, A_v)$ in the set $I(\Theta)$ that may be reordered as $\aleph = (F_1, F_2, \ldots, F_m, G_1, G_2, \ldots, G_n)$.

If we establish the relationship between the set of parameters,

$$\{F_m\} \longleftrightarrow \{O_r\}$$
$$\{G_n\} \longleftrightarrow \{R_s\}$$

an identification criteria may be established among the interrelated states, $\varsigma_i \in I(\Theta)$, and the CBR states, $S_k \in T(BC)$. Therefore a relationship may be established among the agents desires $I(\Theta)$ and the effectiveness operator $\Im$ [S(P)] of the CBR system.

Then the mathematical formalization proposed can be used as a common language between agents and CBR system and solves the integration problem.

Example: If we consider now two possible routes i_1 and i_2, together with their values, presented in Table 4.

i_1= M1 $\wedge$ D(A,A) $\wedge$ M2 $\wedge$ D(A,A) $\wedge$ R1 $\wedge$ D(A,B) $\wedge$ M3 $\wedge$ D(B,B) $\wedge$ R2

i_2= M2 $\wedge$ D(A,A) $\wedge$ R1 $\wedge$ D(A,B) $\wedge$ M3 $\wedge$ Tx(B,A) $\wedge$ M1 $\wedge$ Tx(A,B) $\wedge$ R2

Table 4. Values for the intentions i_1 and i_2

Schedule (hr)	**10-11**	**11-12**	**12-13**	**13-14**	**14-16**	**16-18**	**18-20**	**20-21**	**21-22**
intention	M1	D(A,A)	M2	D(A,A)	R1	D(A,B)	M3	D(B,B)	R2
Costs(€)	6	0	6	0	12	0	0	0	12
Time (hr)	1	1	1	1	2	2	2	1	1
Quality	1	--	2	--	1	--	2	--	2

Schedule (hr)	**10-11**	**11-12**	**12-13**	**13-14**	**14-16**	**16-18**	**18-20**	**20-21**	**21-22**
intention	M2	D(A,A)	R1	D(A,B)	M3	Tx(B,A)	M1	Tx(A,B)	R2
Costs(€)	6	0	12	0	0	3	12	3	12
Time (hr)	1	1	2	2	2	1	1	1	1
Quality	2	--	1	--	2	--	1	--	2

If our coordenates system is represented by $\aleph=(A_1, A_2, A_3, A_4) =(P, T, C, Q)$ where P=Places visited, T=Time spent in the visit, C=Cost of the visit, Q= Quality (visit satisfaction) then the previously presented intentions can be expressed as,

i_1--> P=3, T=12 (h), C=36(€), Q=1.6 i_2 --> P =3, T=12 (h), C=48(€), Q=2

Each intention may be considered as a case, with an associated index, and represented by the values P,T,C,Q

$$idx(C1)=\left\{\frac{(3,1.6)}{(36,12)}\right\} \quad idx(C2)=\left\{\frac{(3,2)}{(48,12)}\right\}$$

If a problem P is presented to the agent in the following terms:

$$S(P)=\begin{Bmatrix} P \geq 3 & Q \geq 1.5 \\ T \leq 12\,h, & C \leq 50\,€ \end{Bmatrix}$$

then idx(C1), idx(C2) are two possible CBR solutions because, given the previously presented definition, P$\subseteq$ idx(C1), idx(C2).

The desire function

$$d : I(\Theta) \longrightarrow \Omega(\aleph)$$
$$i=(e_1 \wedge \ldots \wedge e_r) \rightarrow F=F(i)=\{P\geq 3, Q\geq 1.5, T\leq 12, C\leq 50\}$$

may be expressed in terms of $\Im$ (S(P)) = ($\Im_x$, $\Im_y$). For this example the values are

$\Im_x$ = 0.412, where S(P) must achieve at least 41.2% of its objectives.

$\Im_y$= 0.790, where S(P) should not require more that 79% of the resources.

while the values of the efficiency parameters of cases idx(C1) and idx(C2) are:

$\Im_x$(C1)= 0.432, and $\Im_y$(C1)=0.738

$\Im_x$(C2) = 0.516, and $\Im_y$(C2)= 0.7754

The relationship, presented here, shows how deliberative agents with a BDI architecture may use the reasoning cycle of a CBR system to generate solutions S(P).

When the agent needs to solve a problem, it uses its beliefs, desires and intentions to obtain a solution. Previous desires, beliefs and intentions are stored taking the form of cases and are retrieved depending on the desire to achieve. Cases are then adapted to generate y proposed solution, which is the agent action plan. This initial solution is reviewed and finally a learning process is carried out by adapting, deleting, etc. cases.

The following section shows how the retrieval stage can by automated using variational calculus [12], which facilitate the agents replanning in execution-time.

4 Modelling dynamic CBR-BDI agents

The proposed analytical notation allows the definition of "CBR-BDI" agents. Such knowledge engineering systems have the ability to plan their actions, to learn and to evolve with the environment, since they use the reasoning process provided by the CBR system. CBR systems may be implemented and automated in different ways [10, 13] depending on the problem to solve. This section shows how variational calculus can be used during the retrieval stage of such agents to facilitate the planning and replanning of their intentions in execution-time. Variational calculus is therefore used in the framework of the CBR system to automate the retrieval stage, which gives the agents more autonomy. In general variational calculus provides the optimum solution (geodesical) to a problem [20]. Since we are using this mathematical formalism in a discrete environment (cases: believes, desires and intentions), it will be used to obtain the closest discrete solution to the optimal one [20].

4.1 Mathematical foundations: variational problems

Suppose a space m-dimensional $X=(X_1, X_2,...,X_m)$ and a mapping on X, V(X), that is defined as,

$$\begin{aligned} V : T(BC) &\longrightarrow T(BC) \\ (A_1, A_2,...A_v) &\longrightarrow V(A_1, A_2,...A_v) \end{aligned}$$

On the phase space, which is the set of all states of the process, the function V(X) becomes an m-1dimensional surface that shows all possible relationships between the parameters of V, that is denoted $G(X_1, X_2,...,X_m) = 0$.

Definition 12. Let us consider two points, ei and ef, that fulfil that: $e_i=(X_{i1},X_{i2},...,X_{im})$ holds $G(e_i)=0$, and $e_f=(X_{f1}, X_{f2},...,X_{fm})$ holds $G(e_f)=0$.
It is defined the set $\varphi(e_i,e_f) =\{ \varphi_1(e_i,e_f), \varphi_2(e_i,e_f), ..., \varphi_m(e_i,e_f) \}$ where $\varphi_j(e_i,e_f)$ are possible curves between e_i and e_f that are allowed by V (X) on $X=(X_1, X_2,...,X_m)$) and hold that $\forall\varphi_j(X_1, X_2,...,X_m) \in \{ \varphi(e_i,e_f) \}$ it is satisfied $G[\varphi_j(X_1, X_2,...,X_m)]=0$.

Given the set $\varphi(e_i,e_f)$, variational calculus shows how the optimal curve (geodesical) with respect to its length can be chosen automatically [12, 20].

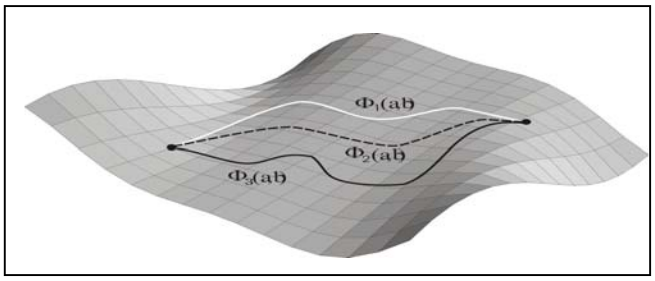

Fig. 2. Possible paths between two points in a 3D space

If m=3 (see Figure 2) and we denote X = (X_1, X_2, X_3) = (x, y, z), the following definition may be included.

Definition 13. A functional A[y=y(x)] defined on a space F is a continuous mapping of F into real numbers

$$\begin{array}{lll} A\colon \Omega^{\infty}(IR) & \longrightarrow & IR \\ y=y(x) & \longrightarrow & A[y=y(x)] \end{array}$$

Ω^{∞} (IR), set of functions infinitely differentiated on IR

If we have a functional A, we can demonstrate that the extreme solutions to this functional are the functions $y=y_0(x)$ such that δA[y(x)] =0. A functional A[y(x)] is called integrable if its expression takes the form

$$A\,[y(x)]=\int_{x1}^{x2}[\,F\,(\,x,\,y(x),\,y'(x)\,]dx$$

For these cases, it is known that a function $y=y_0(x)$ is optimal for the functional A[y(x)] if the Euler´s equation is satisfied [2].

$$A[y(x)]\text{ extremal}\longleftrightarrow\delta A[y(x)]=0\longleftrightarrow\frac{\partial F}{\partial y}-\frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right)=0\longleftrightarrow y=y_0(x)$$

Definition 14. Let us define on the surface G(x, y, z)=0 generated from V=V(x, y, z), the notion of Euclidean distance that associates to each pair of points (ei,ef) a real number D(ei , ef) obtained as

$$D\,(e_i\,,\,e_f\,)=\sqrt{(X_i-X_f)^2+(Y_i-Y_f)^2+(Z_i-Z_f)^2}\quad\text{where } e_i=(X_i,\,Y_i,\,Z_i),\ e_f=(X_f,\,Y_f,\,Z_f\,)$$

So the length of a curve is given by

$$L=\int_{x1}^{x2}[dl]=\int_{x1}^{x2}\sqrt{dx^2+dy^2+dz^2}=\int_{x1}^{x2}\sqrt{1+y'(x)^2+z'(x)^2}\,dx$$

It is known that $G(X, Y, Z)=0$ implies constraints between $y(x)$, $z(x)$, and it defines a pair of new coordinates (ρ,θ) that yields a new equation to be solved,

$$L= \int_{\theta 1}^{\theta 2} \sqrt{1+\rho(\theta)^2}\, d\theta = L\ \{ \theta, \rho(\theta), \rho'(\theta) \}$$

an expression in which Euler´s equation may be applied because L, with the previously shown dependence, is an integrable functional. A generalisation of Euler´s equation exists valid for any number of parameters. In this case, the solution is obtained solving an n-dimensional Euler´s system of differential equations.

4.2 Formalization of the integration of the CBR-BDI agents

The operations that are carried out during the reasoning process of the CBR system are now defined, using the previously introduced notation and Variational Calculus.

Retrieval . During this phase, a problem P′ stored in the case base BC and that is similar to the target problem P is identified.

Given the problems P and P', it is said that P' is "similar" to P and it is denoted $P' \approx P$, if the case $C'= (P', S(P'), \Im[S(P')]) \in CB$, is a possible CBR solution and holds $idx(C') \supseteq \{idx(C_k)\ k_=(1,...., n)\ \}$. So there is a set of possible solutions CBR that we denote $P \subseteq idx(C_1),\ldots,idx(C_m)$.

Now we use the parameter efficiency $\zeta[S(P)]$, that indicates the amount of resources that should be spent to achieve each objective. The cases for which the efficiency is maximum are selected and denoted by $\zeta[S(P)]max$, which is a subset of the previously selected solutions:

$P \subseteq idx(C_1), idx(C_2), idx(C_3),\ldots, idx(C_r)$, with $r \leq m$.

Now we need to identify which is the best case from this subset.

Example. Before to show how such case may be identified, a non-linearity effect in the relationship between the cases with their attributes is introduced below. The visit to a museum M_2, with $Q=1$, may cost $C=2€$, while a museum M_1 with $Q=2$, may be visited for free (if there is a public program of cultural promotion).

To incorporate such non-linearity to the problem, all the non linear processes are codified in the function $V=V(A_1,A_2,...A_v)\neq 0$. The function V on $\aleph= (A_1,A_2,...A_v)$ introduces constrains between such variables that can be graphically associated to "curvature" on the phase space, such as the one represented in Figure 3.

In terms of our tourist agent, considering only $\aleph=(A_1, A_2)=(P, C)$ and given a target problem to solve defined by a $P_{minimum}$ and a $C_{maximum}$, Figure 3 may represent three potential solutions φ_1, φ_2, φ_3, assuming non-linearity effects.

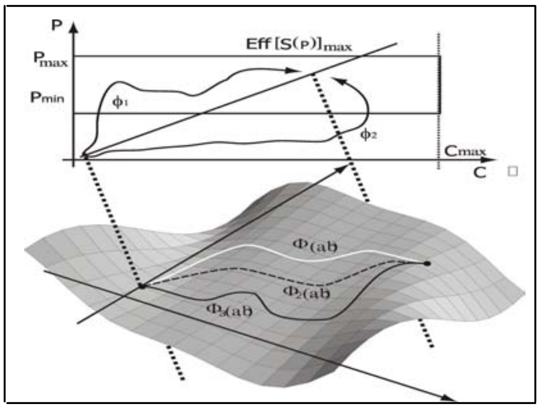

Fig. 3. Effects of no-linearity

Now it will be shown how variational calculus can be used to automate the retrieval process. Let us consider a case base (CB, R)={[C_k / k=1,.,q and q є IN],R} and the set of attributes of the case base T(BC)= (α_1, α_2,,..., α_m), α_j, є T(BC).

Using the relationships between BDI agents and CBR systems established, it is denoted T(BC)=(A_1, A_2,...A_v), which allows us to define a function V on the space I(Θ), that stores the information of all the cases Ck є CB.

$$\begin{aligned} V : T(BC) &\longrightarrow T(BC) \\ (A_1, A_2,...A_v) &\longrightarrow V(A_1, A_2,...A_v) \end{aligned}$$

If we consider two states (Si , Sf) initial and final, on the set I(Θ), the function V shows all the intentions i є I(Θ), that joins both states (Si ,Sf) and that has related a case C_k є CB**.** On the phase space, the function V=V(A_1,A_2,...A_v) is translated onto a surface $\Pi_0[A_1,A_2,...A_v]=0$, where the notion of Euclidean distance is defined.

In the m=3 case, and with A_1=x, A_2=y, A_3 =z, the theory of variational calculus says that a coordinate system (λ, μ) exists which allows an expression of the functional F=F(λ, μ), that associates to each curve between S_i and S_f on $\Pi_0[x,y,z]=0$ with its length, thus we can obtain a solution of

$\dfrac{\partial F}{\partial \mu} - \dfrac{d}{d\lambda}\left(\dfrac{\partial F}{\partial \mu'}\right) = 0$; that we call $\mu=\mu_0(\lambda)$ and that takes the form $\chi_0 = \chi_0[x,y,z]$

on the original coordinates (x, y, z). This function is named the geodesical curve.

In the most general case, the mapping V=V(A_1, A_2,..., A_m) generates curves that cannot be differentiated because V only takes values at discrete points corresponding to defined and stored cases.

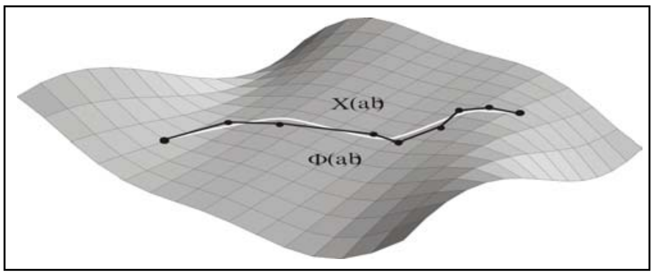

Fig. 4. Optimum (geodesical) and closer to the optimum curve

Let us now define a mapping σ, as $\sigma=(\chi_0 - \psi)$, where χ_0 is the solution obtained by Euler´s equation [20] and $\psi \in \{\varphi(S_i,S_f)\}$ is a path between Si, and Sf, stored in the case-base as a case C ϵ CB (see Figure 4). Then we will call "the closest to the optimal curve ψ_0" the mapping of $\{\varphi(S_i,S_f)\}$ given by the minimisation of

$$I = \int_{ei}^{ef} \{\sigma[x,y,z]\} dxdydz$$

where $\psi_0 = \{ S_i = S_0^{(0)} , S_1^{(0)}, S_2^{(0)}, S_3^{(0)}, S_4^{(0)},....., S_m^{(0)}, ..., S_s^{(0)} = S_f\}$, and S_k are the states obtained to achieve the solution.

So far it has been shown how variational calculus can be used to select the closest to the optimum curve. Variational calculus may then be used to select and retrieve the most appropriate case during the retrieval stage. The retrieved case is characterised for been the one that, in each of its stages, maintains most constant the efficiency.

Adaptation. During the adaptation phase, the system executes a transformational reasoning mechanism [1], that can be represented by the adaptation function A,

$$\begin{aligned} A:\ & (BC) \times \Sigma(P) \rightarrow C \\ & (C\ ,\ P) \rightarrow A[S(P'), P] = \{P, A[S(P')], \Im(A[S(P')])\} \end{aligned}$$

with $P \in \Sigma(P)$ is called set of problems, and $C=(P',S(P'),\Im([S(P')])$.

In [5] it is suggested a retrieval mechanism that identifies a case easy to adapt. Therefore the retrieval mechanism should be subordinate to the adaptation one. In our proposal we assign higher relevance to the retrieval strategy. If P=(Si, Sf) and during the retrieval stage it is obtained $C'=\{P',S(P'),\Im[S(P')]\})\in(BC)$, the adaptation function constructs a solution for P maintaining the sequence of operators given by S(P'). It at any point the sequence may not be applied, a new retrieval cycle is initiated from the state in which the sequence was interrupted.

Therefore the adaptation function can be seen as a serie of operators, where each operator is a part of a retrieved case, and we denote $A= (\alpha m \bullet \alpha m\text{-}1 \bullet ...... \bullet \alpha 2 \bullet \alpha 1)$.

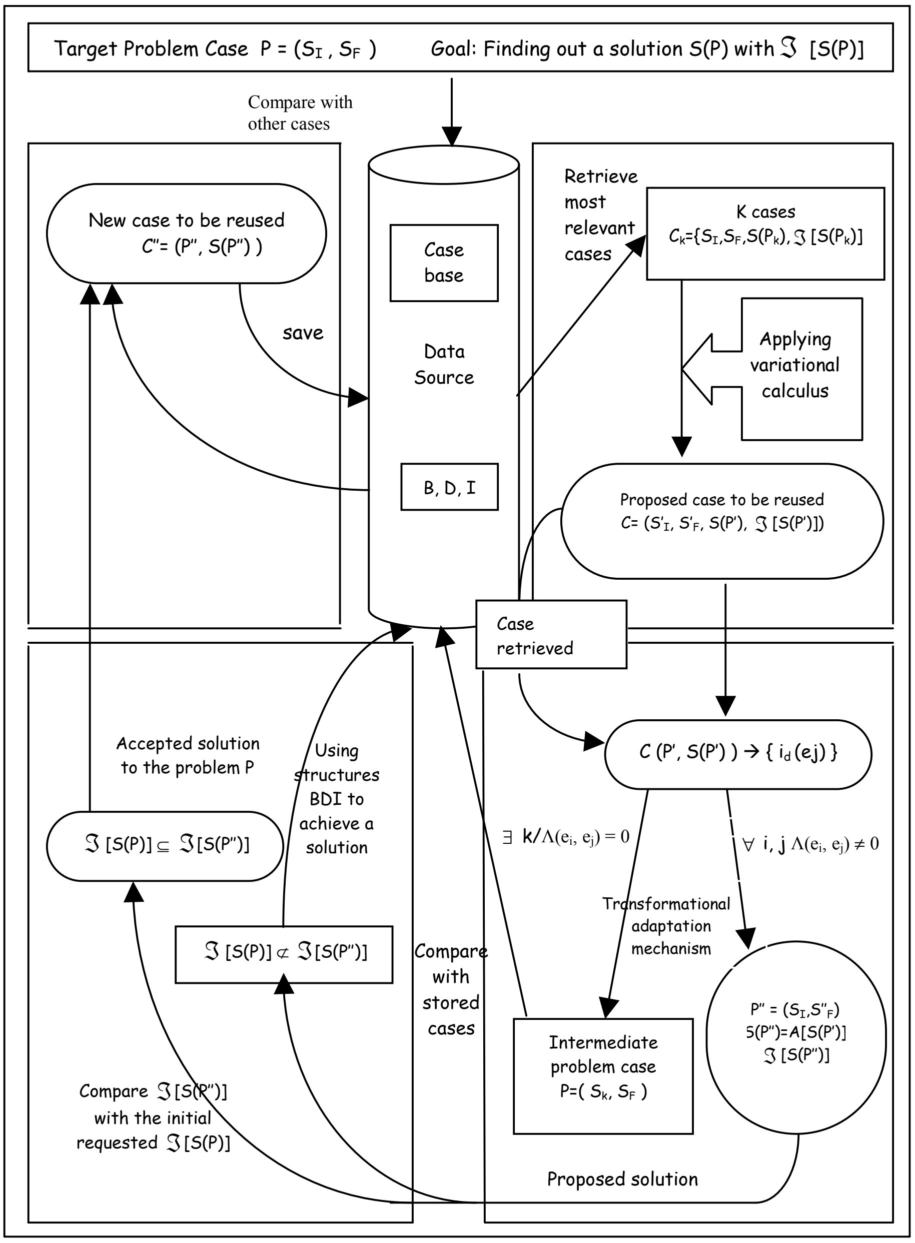
Target Problem Case P = (S_I , S_F)
Goal: Finding out a solution S(P) with ℑ [S(P)]
Compare with other cases
New case to be reused C''= (P'', S(P''))
save
Case base
Data Source
B, D, I
Retrieve most relevant cases
K cases C_k={S_I,S_F,S(P_k), ℑ [S(P_k)]
Applying variational calculus
Proposed case to be reused C= (S'_I, S'_F, S(P'), ℑ [S(P')])
Case retrieved
C (P', S(P')) → { i_d (ej) }
∃ k/Λ(e_i, e_j) = 0
∀ i, j Λ(e_i, e_j) ≠ 0
Transformational adaptation mechanism
Intermediate problem case P=(S_k, S_F)
P'' = (S_I,S''_F) S(P'')=A[S(P')] ℑ [S(P'')]
Proposed solution
Compare with stored cases
Accepted solution to the problem P
ℑ [S(P)] ⊆ ℑ [S(P'')]
Using structures BDI to achieve a solution
ℑ [S(P)] ⊄ ℑ [S(P'')]
Compare ℑ [S(P'')] with the initial requested ℑ [S(P)]

Fig. 5. Formal Model Detailed Schema

Figure 5 shows how variational calculus is applied during the retrieval stage to select the closest to the optimum case from the case-base. In this figure it can also be appreciated graphically the working and information flow during the reasoning process of the agent, introduced this section.

Revise and Memorisation. In this phase the case solution generated in the previous phase is evaluated and reviewed. A problem P occurs for which we want to obtain a solution S(P) with $\Im[S(P)]$. If the adaptation step ensures to the retrieved case, a solution S(P)=A[S(P')], the review must guarantee that: $\Im\{ A[S(P')]\} \supseteq \Im[S(P)]$. The problem target and the characteristics of the adapted solution can be memorized as a new case, which may be denoted by $C=\{P, A[S(P')], \Im(A[S(P')])\}=(P, S(P), \Im[S(P)])$.

4.3 Variational calculus based planner (VCBP)

This section shows how the variational calculus, introduced in the previous section, allows the agents planning and replanning in execution-time because the proposed Variational Calculus based Planner (VCBP) is used to select the most adequate case during the retrieval phase of the reasoning process to solve a given problem. Assuming that potentially significant changes can be determined after executing a primitive action, it is possible to control the dynamism of the new events of the domain and thus achieve an appropriate reconsideration of the problem [14, 16].

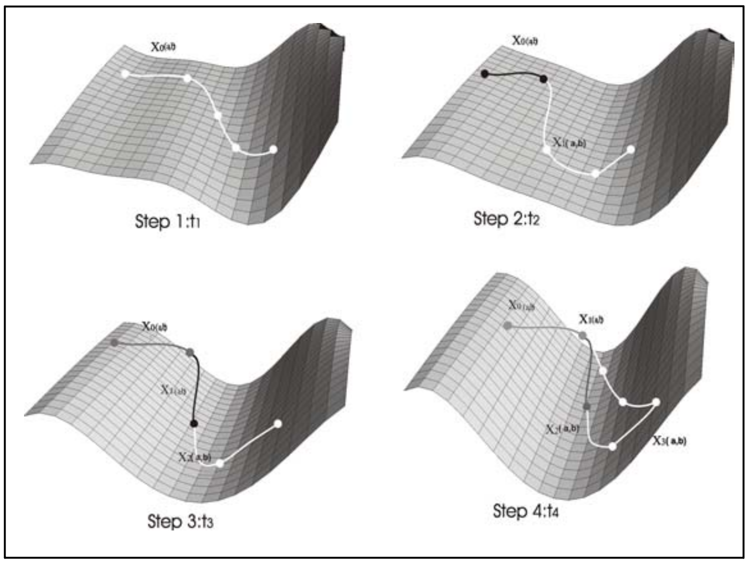

Fig. 6. 3D representation of a dynamic environment

If it is accepted that the environment changes, it is also necessary to define a reasoning mechanism capable of dealing with such changes by modifying the initial desires and intentions. Nevertheless the reasoning process may be maintain since the problem general description remain constant. If at t_0, the function V(x, y, z) takes the form denoted by $V_0(x, y, z)$, at t_1, V is denoted by $V_1(x, y, z)$, with the associated surface $\Pi_1(x, y, z)=0$ on the phases space, upon which it is possible to obtain the optimal curve between two new points, S_i and S_f where $S_i = S_1^{(0)}$ and $S_1^{(0)}$ is the second state of $\psi_0 = \{ S_i = S_0^{(0)},\ldots,S_s^{(0)} = S_f\}$ and S_f is the final state.

Solving the Euler′s equations, $\chi_1=\chi_1(x, y, z)$ is obtained, which may be used to calculate an expression for ψ_1, through the mapping σ,as:
$\psi_1 = \{ S_i = S_1^{(0)}, S_1^{(1)}, S_2^{(1)}, S_3^{(1)}, S_4^{(1)}, \ldots, S_m^{(1)}, \ldots\ldots, S_s^{(1)}= S_f\}$, and the same can be done for any t_j (see Figure 6).

From the previous equations, and based on variational calculus tools, an expression can be determined to identify the final solution of the CBR-BDI agent. This expression, which represents the agent plan, can be obtained in execution-time and takes the following form:

$$\Psi_{final} = \begin{cases} \Psi_0, & t \in (t_0, t_1) \\ \ldots\ldots\ldots\ldots\ldots\ldots & \\ \psi_{s-1}, & t \in (t_{s-2}, t_{s-1}) \\ \Psi_s, & t \in (t_{s-1}, t_s) \end{cases}$$

5 Case-Based Reasoning applied to Planning

Planning can be defined as the construction of a course of actions to achieve a specified set of goals in response to a given situation. The classical generative planning process consists mainly of a search through the space of possible operators to solve a given problem, but for most practical problems this search is intractable. Given that classical planning may engage in a great deal of effort without achieving very good results, several researchers have pursued a more synergistic approach through generative and case-based planning [6].

A case in case-based planning consists of a problem (initial situation and set of goals) and its plan. Given a new problem, the objective of the retrieval phase is to select a case from the case-base whose problem description is most similar to the description of the new problem. In case-based reasoning, two different approaches to reuse can be distinguished: transformational and derivational adaptation.

Transformational adaptation methods usually consist of a set of domain dependent concepts which modifies the solution obtained in the retrieved case directly. For derivational adaptation, the retrieved solution is not modified directly, but is used to guide the planner to find the solution. In case-based planning the “right” elements need to be determined in order to index the cases. In this section, we compare three systems that integrate generative and case-based planning: PRODIGY [8, 25], PARIS [3, 4, 15], and Variational Calculus Based Planner (VCBP), which is the method proposed in this chapter. These planners may be used in the development of deliberative agent-based systems. For example [7] presents an agent-based system that uses a modification of PRODIGY to generate their plans. [14] have also presented a review of this field and present an agent-based system that uses VCBP to achieve its goals. In PRODIGY and PARIS the workload imposed on the generative planner depends on the amount of modification that is required to adapt the retrieved cases. Looking at the structure, we can say that PARIS is a "domain-independent" case-based planner while PRODIGY is "domain semi-dependent".

On the other hand, although VCBP is domain dependent, it introduces a new interesting strategy to deal efficiently with the adaptation stage.

5.1 PRODIGY

PRODIGY was the first system that achieved a complete synergy between generative planning and case-based planning [25]. A planning case P consists of the successful solution trace. The trace is simply the sequence of subgoals.

$$P \rightarrow idx(P)=Tr(P)=\{ g_1, g_2, \ldots, g_n \}=\begin{Bmatrix} g_1 \\ g_2 \\ \ldots. \\ g_n \end{Bmatrix}$$

The system identifies the set of weakest preconditions necessary to achieve each goal, and, through a recursive regression, it is able to determine a guide for the solution beginning from the initial state. The trace of a problem is simply an orderly vector of subgoals and is represented as a vectorial column. After that, each case is multiply indexed by these different sets of interacting goals, building a string of relationships between them (Figure 7).

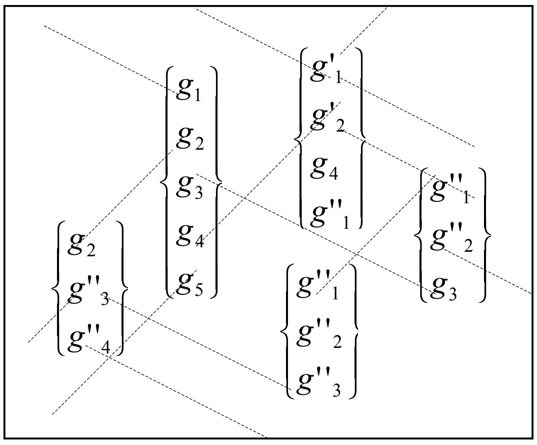

Fig. 7. Indexing string in PRODIGY

PRODIGY is a system for multi-case retrieval planning . When a new problem is presented to the system, the retrieval procedure must match the new initial state and the goal statement against the indices of the cases in the case library. Given a problem P, we denote, idx(P)={(g_1, g_2, g_3,…,g_n)} where {g_i}are subgoals, and therefore, the retrieval solution takes the form,

$$\text{Retrieve }(P)=\{ P'=(g'_1, g'_2, \ldots, g'_n) / \begin{cases} g_1 \approx g'_1 \in P_1 \\ ------ \\ g_n \approx g'_n \in P_n \end{cases} \}\text{where } P_1, P_2, \ldots, P_n \in CB.$$

Until now, PRODIGY has used an exploration mechanism in the state space to obtain the trace of a problem in terms of subgoals. Now it explores the plan space to obtain a solution pattern for the proposed problem. The system returns back to the state space to build a solution adapted to the target problem. So the adaptation stage is based on the "derivational analogy method" that builds lines of reasoning and adapts to a new problem as opposed to transformational methods that directly adapt final solutions. If we denote the problem factorised, the worst case time for finding the complete solution is,

$$S(P)=ADAP[\sum_{j=1}^{k} C_k] \text{ then, } O[S(P)]=O[ADAP(C_1)+\ldots+ADAP(C_k)]=O[f(n\ max)]$$

where "n max" is the highest factor of adaptation (corresponding to the subproblem that requires most execution time in the adaptation process).

The organization of the case base plays an important role in the system. PRODIGY maintains a smaller case base because it stores cases without its detailed description (and connected by subgoals, thereby avoiding repeated information). We can describe PRODIGY as a "semi-dependent work domain planner".

Assuming that the significant changes happen after the planner executes an action (the environment doesn't change before perception and selection, but during the process of execution of the selected action), it is possible to control the dynamism of the environment and thus achieve a re-planning approach through the new events in the domain. If we consider that the environment changes, and the original problem to be solved, P, becomes another problem denoted by P', we can study two situations:

- P'=P+δ P, the new problem can be expressed as a small disturbance with respect to the original. It is assumed that Tr(P)=Tr(P').
- P'=P+Δ P, the new problem generated by different dynamics holds that Tr(P)$\neq$ Tr(P'). In these cases, it is necessary to use backtracking mechanisms to achieve new subgoals and retrieve previous conditions.

5.2 PARIS

PARIS is a domain independent case-based planning system that differs from the traditional case-based reasoning approaches that retrieve, reuse and store cases in a concrete representation. PARIS [3, 4] introduces abstraction techniques into the process and retrieves, reuses and retains cases at different (higher) levels of abstraction. When a new problem P must be resolved, the system associates it with a set of abstract descriptions at different levels. The higher abstract levels are characterized through a reduced level of detail in the representations consisting of fewer features, relations, constraints, operators, etc.

It has been shown that traditional hierarchical problem-solving (e.g. ABSTRIPS, [23]) are too restrictive. Therefore, PARIS enables different levels of abstraction to have their own representation languages, allowing them to express different properties without falling into the problems of more primitive abstract planners.

$$P \rightarrow idx(P)= Abs(P)=\{Abs^{(j)}(P)\}_{j=0,1,\ldots,n} = \{P, Abs^{(1)}(P)\ Abs^{(2)}(P),\ldots,Abs^{(n)}(P)\}$$

The different levels are organized on the form of a hierarchy of cases at different levels of abstraction, as can be seen in Figure 8. Abstract cases at higher levels of abstraction are located above abstract cases at lower levels. The reason for using a structure of abstract cases is that cases at higher-level of abstraction can be used as a kind of prototype, which may be used as indices for a larger set of more detailed related cases, and increases the flexibility of reuse.

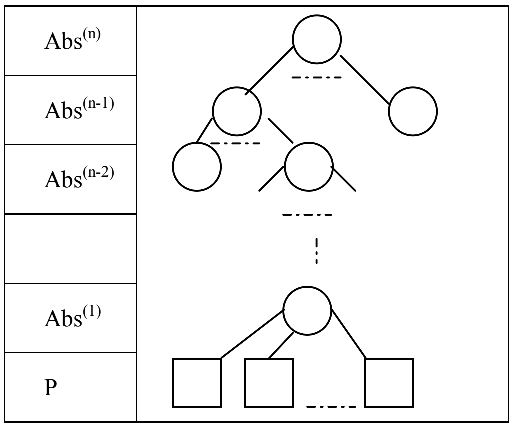

Fig. 8. Abstraction hierarchy in PARIS

When a new problem must be solved, an abstract case is retrieved, whose abstract problem description matches the current problem's level of abstraction. If several matching process are possible, the retrieval procedure selects the case that is located at the lowest level of abstraction (the motivation for this preference is that for more concrete cases, less effort has to be spent during refinement to achieve a complete solution). The branches are searched using a top-down strategy.

$$\text{Retrieve(P)}=\{\, P' \in \text{CB} \,/\, \min_{j}\left(Ab^{(j)}(P) \approx Ab^{(j)}(P')\right) \,\}$$

In the reuse phase, the retrieved abstract solution is refined. Note that PARIS allows the reuse of problem decompositions at different levels of abstraction. The worst case time for finding the complete solution is,

$$\text{S(P)=ADAP}[\sum_{j=1}^{k} C_k\,] \text{ then } O[\text{S(P)}]=O[\text{ADAP}(C_1)+\ldots+\text{ADAP}(C_k)]=O[f[\text{nmax(x)}]]$$

where n max(x) is not a parameter but a function that depends on the user's criteria.

The user can decide how much effort should be required in the adaptation stage and he/she can stop the adaptation process when the algorithm exceeds a fixed time of execution. The kind of storage in PARIS, an organized hierarchy of abstraction cases, allows the integration of a pruning mechanism to conserve the cases at higher levels of abstraction and remove lower levels. It reduces the associated computational costs but, on the other hand, maintains the information from all cases.

Given a problem P, with idx(P)={Abs$^{(j)}$ (P)}$_j$, PARIS matches the case base searching the abstract expression of a stored problem P' that serves as a solution to P. Once this high-level abstract expression is identified, the system begins a process of refinement of the solution through a top-down exploration.

If the environment changes, and the problem P becomes a new problem P', the system will begin an inverse process of exploration down-top, until a more abstract case is found in the tree.

5.3 Comparison between PARIS, PRODIGY and VCBP

The methodology introduced here, Variational Calculus-based Planner (VCBP), associates with each problem P, a representation matrix in terms of two parameters, “Objectives held” and “Resources lost”, as shown in this chapter.

$$\text{P} \rightarrow \text{idx(P)=Mx(P)=} \left\{ \begin{pmatrix} O_r(S_i)_{r=1,\dots,p} \\ R_s(S_i)_{s=1,\dots,q} \end{pmatrix}, \begin{pmatrix} O_r(S_f)_{r=1,\dots,p} \\ R_s(S_f)_{s=1,\dots,q} \end{pmatrix} \right\}$$

This kind of representation allows the organization of indices of the case base using a subsumption hierarchy. As mentioned in previous sections, if we denote all the cases stored as (C1,.., Cq)$\in$CB, the indexing structure can be expressed as

$$\text{idx}(C_1) \subseteq \text{idx}(C_2) \subseteq \dots \subseteq \text{idx}(C_q)$$

which is represented graphically in Figure 9.

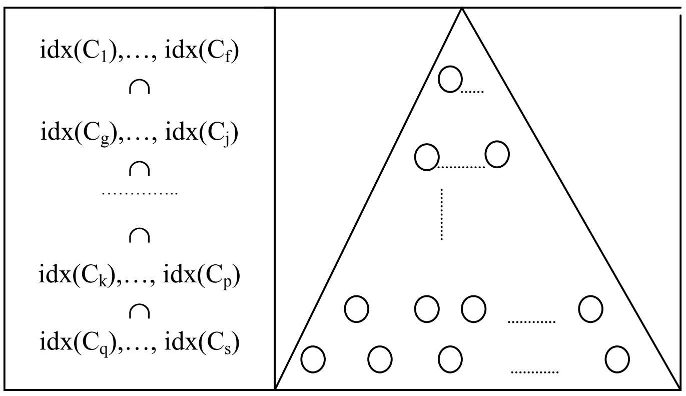

Fig. 9. Subsumption hierarchy in VCBP

Given a problem P, we offer “possible solutions CBR of P" in the following set,

$$\Gamma = \{C_k \in \text{CB} / \{\text{idx}(C_k)\}_{k=1,\dots,m} \supseteq \text{idx(P)}\} \subseteq \text{CB}$$

If it is $\overline{VC}$ called “variational calculus” operator, the retrieved case it is denoted as,

$$\text{Retrieve(P)=\{ P'} \in \text{CB / P'} = \overline{VC}(\Gamma)\}$$

Variational calculus determines an approach for selecting the best stored case to be retrieved. This strategy is part of a mechanism of transformational adaptation. In this system, it can be said that the "reuse stage" is subordinate to the "retrieval stage" [14]. For this reason, the computational costs at this stage, are lower than in PRODIGY or PARIS. If we express the solution in terms of the retrieved cases, and we call it

$$S(P)=ADAP[\sum_{j=1}^{k} C_k], \text{ then}$$

$$O[S(P)]=O[ADAP(C_1)+ADAP(C_2)+...+ADAP(C_k)]=O[C_1+C_2+...+C_k]= k$$

O[S(P)] is proportional to the number of cases that construct the solution. Consequently, the variational strategy reduces the effort of adaptation, increasing the number of cases at the retrieval stage. The transformational mechanisms force the application to solve problems that have domain dependence. In planners like PARIS, the dependence of the domain is smaller because it is based on strategies of abstraction, and therefore, it is more flexible in its application. But in other domains, using transformational adaptation reduces the computational costs and the time required for execution of the program. Formally, VCBP uses a reasoning process in execution time expressed in similar terms to when it solves static problems, since the general description of the problem remains constant.

On the one hand, the three systems presented in this chapter have in common the synergistic integration of generative planning with a case-based reasoner. On the other hand, they differ in many respects because of different motivations and mechanisms. Such similarities and differences are now outlined with respect to the indexing strategy, the retrieval, reuse and retention stages and the systems' dynamic behaviour.

Index

- In PRODIGY a goal regression process is performed to determine the features and subgoals relevant to a particular case. The result is called the trace of the problem.
- PARIS stores several abstractions of the same solution. The result is an abstraction vector for each problem.
- VCBP associates with each problem P, a representation matrix in terms of two parameters, "Objectives held" and "Resources lost".

Indexing

- In PRODIGY, indexing is based on the partially ordered solution plan. Connected components are determined and identified in order to index independent subparts of the cases.
- In PARIS, indexing is based on an abstraction process that represents several concrete solutions, and by refining it, a detailed solution to a target problem is achieved.
- VCBP uses as indexing strategy a Subsumption hierarchy to organize cases.

Retrieval

- PRODIGY. When matching the candidate cases and the target problem, only relevant features are taken into account. Multi-case Retrieval is bound to finding a case with a "reasonable" partial match.
- In PARIS, a case corresponding to an abstract plan is retrieved. This plan solves the abstracted problem description finding an appropriate set of cases that build a solution at the lowest possible level of abstraction.
- In VCBP, variational calculus determines the criteria to select which is the best stored case to be retrieved. The strategy supports, as shown, multi-case retrieval.

Reuse

- PRODIGY can construct a new solution from a set of guiding abstract cases as opposed to a single and concrete past case. It is based on the derivational analogy method.
- A retrieved case in PARIS is an abstraction of a plan. Adaptation refines the retrieved case, considered as a guide, to produce a solution at the concrete planning level.
- In VCBP, the adaptation function can be seen as a series of operators, where each operator is a part of a retrieved case. This is a transformational method that adapts directly final solutions.

Retention

- PRODIGY, through its structure, is a good planner in a variety of domains. It can be applied to problems with low domain dependence
- PARIS is a domain-independent case-based planning system.
- The transformational mechanism of VCBP forces its use in problems that have domain dependence.

Dynamics

- When the environment changes and a new problem is presented to the system, PRODIGY uses backtracking methods to achieve a previous situation.
- In PARIS, the abstraction hierarchy is traversed down-top, from the solution that fails, following those branches in which abstract cases are similar to the current problem.
- VCBP allows planning and replanning in execution-time because its formalism follows the same rules in static and dynamic environments.

The following table summarizes the previously mentioned characteristics.

Table 5. Summary of characteristics of planners

System	Index	Indexing	Retrieval	Reuse	Retention	Dynamics
PRODIGY	Tr(P)	Subgoals joint	Multi-case	Derivational	Intermediate domain dependence	Top-down
PARIS	Ab(P)	Abstraction tree	Multi-case	Derivational	Low domain dependence	Backtracking
VCBP	Mx(P)	Subsumtion herarchy	Multi-case	Transformational	Domain dependence	Execution time

6 TOURISTGUIDE-USAL: A "CBR-BDI" system to solve problems in the e-tourism domain

The framework in which this mathematical formalization and experiments are being developed aims to design and implement an agent-based tool, as well as integrating existing state of the art in order to create an open, flexible, global anticipatory system with mobile access for the promotion and management of inland and cultural tourism, which will be user-friendly, cost-effective and secure. The system will be standardized and interlingua. It will be aimed as both a B2B and B2C tool and thereby help individuals, private enterprise and public bodies connected directly and indirectly to tourism to achieve higher quality of service.

The integrated, multi-platform computer tool developed has been used to design an agent based system for the promotion of inland and cultural sites for tourism based on their cultural worth, the recreational activities on offer and new perspectives on sources of patrimonial interest. This has been combined with horizontally and vertically compiled information on hotel accommodation, restaurants, the commercial sector and transport, in order to meet the needs of the potential visitor on an individually customized basis and respond to requests for information, reservations and purchases in the precise moment that they are expressed.

The project aims to develop innovative, practical and multidisciplinary solutions which aim to use the varied knowledge of individuals at each location, and to organize the different services – often offered chaotically by different sources – within a single, dynamic, interconnected knowledge system. In order to achieve this, it will be necessary to integrate within a TIC platform, information that will facilitate the management of the knowledge lifecycle of various organizations.

One of the initial steps in this research is to develop an agent architecture for modelling autonomous agents. In this context our first experiment has been to design an individual agent, using the previously presented formalization, whose aim is to assist tourists in identifying an optimum schedule for a day trip in the city of Salamanca.

Figure 10 describes the interaction process between the user and the tourist guide-usal agent. The tourist uses a mobile device (i.e. mobile-phone, PDA, etc.) to contact with the agent. After, the user introduces his/her login and password, and indicates to the agent his/her preferences (monuments to visit, times for dinner and type, amount of money to spend, visit duration, etc.).

Then the agent assigned to that particular tourist generate a customized plan and show it to the user, using the previously described case-based reasoning model. Such CBR-BDI system stores the beliefs, desires and intentions of the agent. Moreover the agent has information about previous tourist requests, plans and about the degree of satisfaction of the tourist after using the agent based system. The tourists may also change their schedule on the move for example if they decide to move the dinner time forward repeat a visit to a particular place, etc.

The CBR-BDI agent then will be able to adapt the plan to the new tourist requirements in execution-time. Also, since the CBR is learning continuously, the agent is learning too and could provide different schedules at different points in time for the same tourist query.

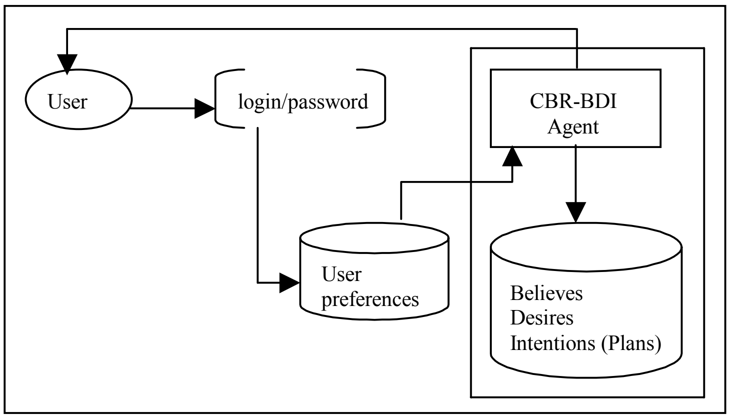

Fig. 10. Schema of relations in TOURISTGUIDE-USAL

The TOURISTGUIDE-USAL has proved successful and is now in the process of application. This methodology is being used to construct agent-based systems in areas such as robotics and traffic control processes.

7 Conclusions

The integration of CBR systems and BDI agents solves one of the problems of the BDI (deliberative) architectures, which is the lacking of learning capacity. The reasoning cycle of the CBR systems helps the agents to solve problems, facilitate its adaptation to changes in the environment and to identify new possible solutions. New cases are continuously introduced and older ones are eliminated. The CBR component of the architecture provides a straight and efficient way for the manipulation of the agents knowledge and past experiences. The proposal presented in this chapter reduces the gap that exists between the formalization and the implementation of BDI agents. What we propose in this article is to define the beliefs, desires and intentions clearly (they don't need to be symbolic or completely logic), and to use them in the life cycle of the CBR system, to obtain a direct implementation of a BDI agent.

A mathematical formalism has been introduced to facilitate the representation of BDI deliberative agents and of CBR systems. This analytical formalism also allows the integration of both models and provides a robust framework for the definition and the automatization of the reasoning cycle of the agents, here presented.

Agents need to respond on real time to the user request and to adapt their solutions in real time, since the inhabit dynamic environments. Variational calculus has been introduced in this chapter to facilitate the agents to define their plans and to replanning in execution-time in order to provide the best possible service. Variational calculus can be used to obtain the most adequate plan to achieve a goal in environment with uncertainty.

This chapter has shown how the proposed knowledge engineering architecture may be used to design an agent for an e-tourism problem. The work presented in this chapter is just the first step toward the development of an ambitious framework for developing communities of agents capable of solving problems in an autonomous and intelligent manner. Although the architecture and formalization described have been applied to the e-tourism domain, we believe it could be also used in any other domain in which agents with learning and adaptation capabilities are required.

Acknowledgements

This work has been partially supported by the IST-2001-34240 project, by the CICYT projects TEL99-0335-C04-03 and SEC2000-0249 and the PGIDT00 project MAR30104PR.

References

1. Aamodt A. and Plaza E. (1994) Case-Based Reasoning: foundational Issues, Methodological Variations, and System Approaches, AICOM. Vol. 7., March, pp 39-59.
2. Bratman M.E. (1987) Intentions, Plans and Practical Reason. Harvard University Press, Cambridge, M.A.
3. Bergmann, R. and W. Wilke (1995). Learning abstract planning cases. In N. Lavrac and S. Wrobel (Eds.), Machine Learning: ECML-95, 8th European Conference on Machine Learning, Heraclion, Greece, April 1995. Number 912 in Lecture Notes in Artificial Intelligence, pp. 55-76. Berlin, Springer
4. Bergmann, R. and W. Wilke (1996). On the role of abstraction in case-based reasoning. Lecture Notes in Artificial Intelligence, 1186, pp. 28-43. Springer Verlag.

5. Bergmann R. and Wilke W. (1998). Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. European Conference in Artificial Intelligence (ECAI'98), John Wiley and Sons, pp.224-236.
6. Bergmann, R., Muñoz-Ávila, H., Veloso, M. and Melis, E. (1998). CBR Applied to Planning. In Lenz, M. Bartsch-Sporl, B., Burkhard, H. and Wess, S. (Eds.) Case-Based Reasoning Technology: From Foundations to Applications. Lecture Notes in Computer Science 1400, pp. 169-200. Springer 1998, ISBN 3-540-64572-1.
7. Camacho D., Borrajo D. And Molina J. M. (2001) Intelligence Travell Planning: a multiagent planing system to solve web problems in the e-turism domain. International Journal on Autonomous agens and Multiagent systems. 4(4) pp 385-390. December.
8. Carbonell J.G., Knoblock C. A., Minton S. (1991). Prodigy: An integrated architecture for planning and learning. In K. VanLenh (Ed.), Architectures for Intelligence, pp.241-278. Lawrence Erlbaum Associates, Publishers.
9. Cohen P.R. and Levesque H.J. (1990) Intention is choice with commitment. Artificial Intelligence, 42(3) pp: 213-261, 1990.
10. Corchado J. M. and Lees B. (2001) A Hybrid Case-based Model for Forecasting. Applied Artificial Intelligence. Vol 15, no. 2, pp105-127.
11. Corchado J. M. and Laza R. (2002). Construction of BDI Agents from CBR systems. First German Workshop on Experience Management, Berlin, March, 2002. ISBN 3-88579-340-7, ISN 1617-5468, Vol. 1, pp 47-60.
12. De Groot M. H. (1970) Optical Statiscal Decisions. McGraw-Hill. New York
13. Fyfe C. and Corchado J. M. (2001) Automating the construction of CBR Systems using Kernel Methods. International Journal of Intelligent Systems. Vol 16, No. 4, April 2001. ISSN 0884-8173.
14. González Bedia M. and Corchado J. M. (2002) A planning strategy based on variational calculus for deliberative agents. Computing and Information Systems Journal. Vol 10, No 1, 2002. ISBN: 1352-9404, pp: 2-14.
15. Holte, R. C., T. Mkadmi, R. M. Zimmer, and A. J. MacDonald (1995). Speeding up problem solving by abstraction: A graph-oriented approach. Technical report, University of Ottawa, Ontario, Canada.
16. Jennings N.R. (1992) On Being Responsible. In Y. Demazeau and E. Werner, editors, Decentralized A.I. 3. North Holland, Amsterdam, The Netherlands.
17. Kinny D. and Georgeff M. (1991) Commitment and effectiveness of situated agents. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91), pages 82-88, Sydney, Australia.
18. Knobolock C. A., Minton S., Ambite J. L., Muslea M., Oh J. and Frank M. (2001) Mixed- initiative, multisource information asistants. 10th International world wide web conference (WWW10). ACM Press. May 1-5,pp.145-163.
19. Martín F. J., Plaza E., Arcos J. L. (1999). Knowledge and experience reuse through communications among competent (peer) agents. International Journal of Software Engineering and Knowledge Engineering, Vol. 9, No. 3, 319-341.
20. Morse P. M and Feshbach H. (1953) Methods of Theoretical Physics. McGraw-Hill. New York.
21. Olivia C., Chang C. F., Enguix C.F. and Ghose A.K. (1999) Case-Based BDI Agents: An Effective Approach for Intelligent Search on the World Wide Web",

AAAI Spring Symposium on Intelligent Agents, 22-24 March 1999, Stanford University, USA, pp. 76-84.
22. Rao A. S. and Georgeff M. P. (1995) BDI Agents: From Theory to Practice. First International Conference on Multi-Agent Systems (ICMAS-95). San Francisco, USA, pp 312-319.
23. Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces. Artificial Intelligence 5, 115-135.
24. Shoham Y. (1993) Agent-Oriented programming. Artificial Intelligence, 60(1): pages 51-92.
25. Veloso, M. M. (1994). Planning and Learning by Analogical Reasoning. Number 886 in Lectures Notes in Computer Science. Berlin, Springer.
26. Watson I. and Marir F. (1994) Case-Based Reasoning: A Review. Cambridge University Press, 1994. The knowledge Engineering Review. Vol. 9(3), pp. 249-293.
27. Wendler J. and Lenz M. (1998) CBR for Dynamic Situation Assessment in an Agent-Oriented Setting. Proc. AAAI-98 Workshop on CBR Integrations. Madison (USA) 1998, pp.172-186.
28. Wooldridge M. and Jennings N. R. (1994) Agent Theories, Architectures, and Languages: A Survey. Procs. ECAI-94 Workshop on Agent Theories, Architectures, and Languages, pp.317-361.