Using Case-Based Reasoning in Robotics*

Igor Jurišica
Department o Computer Science
University of Toronto
Toronto, Ontario M5S 1A4 Canada
Tel.: (1—416) 978-7589
email: juris@ai.utoronto.ca

Janice Glasgow
Dept. of Computing and Information Sciences
Queen's University
Kingston, Ontario K7L 3N6
Tel.: (1—613) 545-6058
email: janice@aucis.queensu.ca

Abstract

The paper describes an application of a case-based reasoning system ***TA3*** *to robotics. Our main application is robot control but we discuss deployment of* ***TA3*** *for recognition, design and diagnostics.* ***TA3*** *differs from previous methods in its approach to relevancy-based retrieval, which allows for greater flexibility and for improved accuracy in performance.*

Most successful robotic manipulators are precise, fast, smooth and have high repeatability. Their major drawback is their tendency to have only a limited repertoire of movements that can only be extended by costly inverse kinematic calculations or direct teaching. Our approach also starts with a limited repertoire of movements, but can incrementally add new positions and thus allows for higher flexibility.

The proposed architecture is experimentally evaluated on two control tasks and the results are compared to other machine learning algorithms applied to the same problem. In addition, we have evaluated ***TA3's*** *performance on character recognition and design tasks.*

* This research was supported by the Information Technology Research Centre of Ontario. Authors are indebted to J. Mylopoulos for helpful comments and suggestions.

1. Introduction

Robot control has been studied for the past three decades [35]. Starting with a simple joint-based position control of serial link manipulators, the field then diversified to sophisticated control techniques, such as adaptive control and learning control.

A control system capable of remembering a repeated operation is called a learning control system [2,4,39]. A learning process can increase the number of solvable tasks (i.e., acquire new knowledge) or it can improve the efficiency of solving repeated tasks. Most current applications of machine learning in robotics are oriented to the perfection of the control process. A learning control system for manipulators, whose endpoint is moving under geometrical constraints on surface, was proposed in [31]. [13] describes an iterative learning system for precise tracking control with guaranteed convergence.

In [29] the Theo-Agent architecture of a reactive system is proposed. The system is used in robot control and combines a stimulus-response subsystem for rapid reaction with a search-based planner for handling unanticipated situations, and an explanation-based learning mechanism to construct stimulus-response rules to cover new situations. Thus, the system becomes increasingly reactive, since over time less planning is required as the machine learning component acquires more control rules.

The Candide system [27] has been used for the acquisition of a control knowledge. The system learns a qualitative model of the system first, then, based on the selected mode, it creates a rule-based incremental controller for a given system.

The SINS system [35] is a self improving reactive control system for autonomous robotic navigation. The system is based on using case-based techniques and reinforcement learning. The learning module monitors the system and incrementally modifies the case representations to accommodate the changes. Because of the utility problem [28], the SINS system uses only 10 cases to allow for real time performance.

In this paper, ***TA3***$_{\mathrm{CTRL}}$, a novel learning control system that incorporates ***TA3*** case-based reasoning (CBR) system is proposed. A case-based approach [26] relies on the concept that similar problems have similar solutions. Thus, facing a new problem, a case-based system retrieves similar cases stored in a case base and adapts them to fit the problem at hand. The evaluation of the ***TA3***$_{\mathrm{CTRL}}$ system in this paper and the evaluation of the ***TA3*** system elsewhere [19,14,18,20,21,22,23,15,17] suggest that the proposed reasoning architecture can be used in a robotic domain for improving manufacturing flexibility.

The ***TA3*** system is similar to the Theo-Agent architecture [29] in the sense that it acquires new knowledge over time to improve the control component. The difference is in the form of acquired knowledge. Theo-Agent constructs stimulus-response rules, while ***TA3*** acquires episodic knowledge. ***TA3*** is also similar to SINS, mainly because both systems use case-based techniques. The case base of SINS consists of continuous, time-varying, analog information that is hard to explain. Moreover, the retrieval of such knowledge is expensive which limited its applications to those with small case bases. ***TA3***'s cases are easy to explain and the system also allows for larger case bases without sacrificing performance.

Informally, a case comprises the problem, the solution and feedback (an evaluation of the solution after the control systems loop is completed, when the result of the applied solution can be assessed). Case-based reasoning involves the processes of: accepting a new problem description (a case without a solution and feedback) from a client (user or another part of the system); retrieving relevant cases from a case base (past problems with similar input); and adapting retrieved cases to fit the problem at hand and producing a solution for it.

Considering the case-based reasoning cycle, one can say that the more similar the cases are the less adaptation is necessary. Thus, in general, the proposed solution is more precise. Now, the task is how to measure case relevance (similarity or closeness) to guarantee retrieving only highly relevant cases. In many mission-critical applications, it is better to retrieve fewer cases (or none) than to retrieve less relevant cases that would result in a poor solution. If, however, the CBR system is used as an optimization module of the underlying system, it may be perfectly acceptable for the CBR module to produce an approximate result fast, which can then be optimized later. Clearly, flexibility of the relevance measure allows for different scenarios to be modeled by the system and thus is desirable [23].

2. The *TA3* Case-Based Reasoning System

As shown in various theoretical and experimental evaluations, ***TA3*** allows for flexible computation under various conditions which enables changing the tradeoff between resources used and accuracy/precision.

As was already mentioned, retrieving only relevant cases is the core part of the proposed system. A theoretical basis for the proposed system is introduced in the subsequent sections (formal definitions are provided in [19,15]).

2.1. Case Representation

In knowledge-based systems, more sophisticated knowledge representation schemes usually imply more powerful reasoning systems. However, with the increased complexity of knowledge representation comes difficulties with knowledge acquisition. Thus, in order to allow for an automatic and incremental acquisition of experiences, required in the real world robotic domain, the ***TA3*** system uses simple case representation.

A source case C is a representation of a real world episode or a problem; e.g., for an inverse kinematic task (IKT), the problem is described as a desired position of the end effector and the solution is a set of corresponding joint angles. A case will be represented as a finite set of attribute/value pairs (tuples): $C = \{< a_0{:}V_0 >, < a_1{:}V_1 >, \ldots, < a_n{:}V_n >\} = \{A_i\}_0^n$, where $A_i =< a_i{:}V_i >$ is an attribute/value pair. Source cases are stored in a persistent storage -- a case base: $CB = \{C_1, C_2, \ldots, C_k\}$.

An input case C_{input} is a special source case that describes a given problem either in full or only partially, i.e., only a subset of all attributes may have defined values.

When comparing cases in a case base, a case C_1 is **equal** to a case C_2 if and only if for all attributes all values of a case C_1 are equal to values of a case C_2. Thus, using an IKT example, two cases are equal if, for a given robot, they match the problem description, e.g., if $P_x = P_x^{'}, P_y = P_y^{'}, P_z = P_z^{'}$. Then, it is assumed that they have the same solution.

2.2. Context

A partial equality exists if a subset of attribute values of a case C_1 match only a subset of values of a case C_2. In order to control the partial matching we define context. The goal is to retrieve not only exact matches (equivalent cases) but partial matches (similar cases) as well. In short, context is a parameter of a relevance relation that maps a case base onto a set of relevant (in terms of context) cases. Context allows for controlling what can and what cannot be considered as a partial match. More formally, a context Ω is defined as a finite set of attributes with associated constraints on the attribute values: $\Omega = \{< a_0{:}CV_0 >, < a_1{:}CV_1 >, \ldots, < a_n{:}CV_n >\}_0^n$, where a_i is an attribute name and constraint CV_i specifies the set of "allowable" values for attribute a_i. Context specifies two things:

1. important attributes and
2. how "close" attribute value must be in order to be considered as match.

Context allows for specifying a variety of situations, ranging from completely ignoring the value of an attribute, through considering only certain ("allowed") values for an attribute, to considering a certain set, consisting of all values except some "prohibited" values. In the next sections we demonstrate that this allows for a flexible retrieval specification and for a simple method for fine-tuning queries. Context can be changed dynamically; it supports dynamic similarity function.

An obvious problem, however, is selecting a method for specifying a context. There are several possible scenarios for creating the initial context and their advantages/disadvantages depend on the actual implementation [21,15].

In the robotic domain, the initial context should be specified without human intervention, but should be changeable, based on different tasks being solved. In the simplest approach, a complete input case is mapped to the context: $\forall (a_i \in C), (a_j \in \Omega){:}V_i \rightarrow CV_j$. In the IKT, the problem would comprise the coordinate of the end-effector and characteristics of the robot. The initial context may be subsequently changed by automatic context modifications, as explained in the sequel.

A machine learning or knowledge mining algorithm [1] can be used to select important attributes for a given task and to specify characteristic values for them [22]. This information can then be used for initial context specification and for the specification of context modification strategies.

We say that a case satisfies (or matches) a particular context, denoted $sat(C,\Omega)$, if and only if for all pairs $< a_i{:}CV_i > \in \Omega$, there exists a pair $< a_i{:}V_i > \in C$ such that V_i is in CV_i:

$$sat(C,\Omega) \ \mathit{iff} \ \forall < a_i{:}CV_i > \in \Omega, \ \ \exists < a_i{:}V_i > \in CV \ | \ V_i \in CV_i$$

Assume a case representation used in a Servo Domain[1]

MOTOR	E
SCREW	D
PGAIN	4
VGAIN	5
RISE TIME	0.28125095

then if the context specifies *motor* and *screw* as important attributes, with their corresponding values to be *E* and *D* respectively then all cases matching the values on these two attributes are considered relevant.

One of the possible approaches for changing context dynamically is to use generalization hierarchies of constraints in a given context. This allows for automatic relaxation/restriction of

[1] ftp://ics.uci.edu/machine-learning-databases/servo/ contains the domain description and test data.

context using domain knowledge. A constraint value CV_j is called a **generalization** of a constraint value CV_i, denoted $CV_i \prec CV_j$ if and only if CV_j subsumes CV_i, i.e., if and only if either $CV_i \in CV_j$ or CV_i is-A CV_j. Conversely, if CV_j is a generalization of CV_i then we say that an attribute value CV_i is called a **specialization** of an attribute value CV_j. However, such hierarchies are not always available, as is the case for a servo domain.

It is obvious that generalization of attributes leads to enlarging the number of retrieved cases since more cases would satisfy it. Specialization of context leads to a decreased number of cases that satisfy it. The following definition of relaxation and restriction of context allows us to control the number of cases returned during the retrieval process. A context Ω_1 is a **restriction** of a context Ω_2, denoted $\Omega_1 \prec \Omega_2$:

$$\Omega_1 \prec \Omega_2 \ \ iff \ \ \forall < a_i : CV_i > \in \Omega_1,\ \exists < a_j : CV_j > \in \Omega_2 : a_i = a_j \wedge CV_i \subseteq CV_j$$

Conversely, if Ω_1 is a restriction of Ω_2 then we say that Ω_2 is a **relaxation** of Ω_1.

Based on the previous definition we can define two types of context modification: relaxation and restriction. Context changes can modify the number of attributes required to match (termed cardinality relaxation/restriction), or they can modify attribute values (termed value relaxation/restriction):

- **Value relaxation/restriction** covers the situation when the initial value constraints are made more general/specific. This can be done either by extending/restricting the attribute value or by generalizing/specializing the value. Value restriction is performed in the opposite direction, making the attribute constraint more specific. Using the context formalism presented in [30], value relaxation/restriction can be expressed as $\Omega'' = \Omega \pm \Omega'$, where Ω is the initial context, $\Omega' = \{< a_k : \{V\} >$, $\pm$ is a relaxation/restriction operator and Ω'' is a modified context.
- **Cardinality relaxation/restriction** changes the initial matching requirement so that only $m < n$ attributes are required to match, instead of all *n* attributes. Although *m* can be initially set to an arbitrary value $0 < m \leq n$, it makes sense to change it only incrementally/decrementally during restriction/relaxation. Thus, after *m* is specified, it does not need to be an input parameter to the relaxation/restriction function. If no value for *m* is explicitly stated, we assume *m* = *n*. It should be noted that despite the possible range for values of *m*, only *m* "close" to *n* make sense. If $m << n$ (i.e., *m* is substantially smaller than *n*) then a loose matching criteria are defined. The experiments presented in [32] show that setting *m* = *n* - *2* allowed for the best results considering the ratio of correct and incorrect answers.

Using the previous servo case example, cardinality relaxation means that we do not require all four attributes to match (*motor*, *screw*, *pgain* and *vgain*) but only, for example *2_of_4*. In this case it can be specified that only *motor* and *screw* must match, or that any two attributes need to match. Value relaxation could be performed, for example, by allowing *motor* to have values of *D* or *E* instead of value just *D*.

2.3. Similarity

In the proposed theory, similarity is considered as a relation between two cases. It is defined to supplement equivalence by allowing for partial matches. In order to control to what level the match is partial, we use context as part of the similarity relation. A case C_1 is **similar** to a case C_2 for a given context Ω, denoted $C_1 \approx_\Omega C_2$, if and only if both C_1 and C_2 satisfy context Ω:

$$C_1 \approx_\Omega C_2 \leftrightarrow sat(C_1, \Omega) \wedge sat(C_2, \Omega)$$

Thus, cases that satisfy a given context can be treated as equivalent in that context. It is assumed that cases with equivalent problem descriptions have equivalent solutions.

The similarity relation between two cases can be generalized into a relation among a set of cases [12]. Assume a set of cases, *SC*, a context Ω and a case base *CB*. The relation $sim(SC, \Omega, CB)$ states that the cases contained in set $SC \subseteq CB$ are considered similar in a context Ω and thus are relevant to a given request. Retrieved cases are considered relevant if they are similar to the query with respect to a given context. All cases that are relevant to a given request can be treated as equivalent.

3. The *TA3*$_{\text{CTRL}}$ System

Based on the novel theory of relevance assessment introduced above, the ***TA3***$_{\text{CTRL}}$ case-based reasoning system has been designed and implemented. The architecture of the proposed system is presented in Figure 1. Individual components are described in sequel.

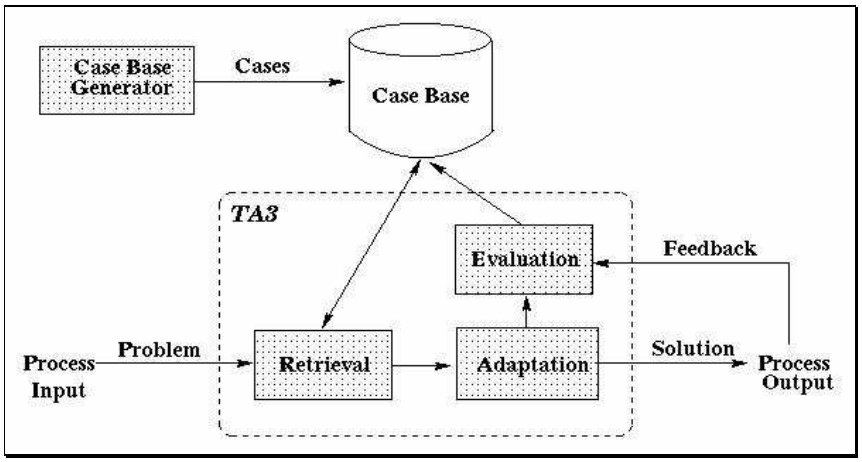


Figure 1. ***TA3***$_{\text{CTRL}}$ architecture.

3.1. The CB Generator

The CB Generator generates a case base (CB), a persistent storage for cases. There are three main possibilities:

1. generating cases at random;
2. generating cases uniformly across the working space and
3. generating cases via learning by observing the process.

The first two approaches are off-line learning methods and the third approach is an on-line method. Even though these methods will be discussed in more detail in the experimental studies, the main idea here is to use off-line learning first and then to support flexibility by additional on-line learning. The issue is to determine how much knowledge should be learned on-line.

3.2. The Retrieval Component

Based on the problem description, the retrieval component locates all relevant cases from a case base, i.e., cases that are similar to a given problem. A novel theory of relevance measure is used, based on a context-based similarity assessment.

Using the servo case example, the problem description includes first four attribute/value pairs. The retrieval algorithm locates all cases that match the problem description and the subsequent adaptation module predicts the value for a rise time attribute. The basic retrieval algorithm is presented in Figure 2.

```
retrieval_naive (Ω, CB)
    C_input → Ω
    for (i=1;i<|CB|;i++)
            retrieve (C_i)
            if sat(C_i, Ω) then mark C_i as a match
            else continue
    end
    return all matching cases
end
```

Figure 2. Naive retrieval algorithm.

This algorithm is simple for two reasons:

1. it uses an input case to specify context and no modifications are used to change the initial context and
2. the algorithm does not use any indexing or clustering to improve search performance.

Later we will show two major improvements of this algorithm:

- using an incremental view maintenance algorithm for context modification; and
- using inductive learning and knowledge mining techniques for retrieval process.

As was already mentioned, it is desired that not only exact matches but also close (similar) matches are retrieved. It should be noted that not all attributes are equally important during matching. The attributes that are preferred for the retrieval process depend not only on the application domain but on the task as well.

Our retrieval algorithm allows for flexibly changing the importance of individual attributes, which helps in retrieving only highly relevant cases even if the problem solving task is changed. For example in the Servo Domain, *motor* and *screw* attributes are the most important for locating cases relevant for predicting servomechanism rise time. The other two attributes also contribute but a partially matching case (not all attribute/value pairs in a given problem match a case in a case base) on *motor* and *screw* is more relevant than a partially matching case based on *pgain* and *vgain*. For another task, attributes *pgain* and *vgain* may be more important. This switch is possible via increasing importance of the last two attributes, while simultaneously decreasing importance of the first two attributes. This can be performed by the human expert on-line, or in more stable domains it can be supported via predefined profiles for individual tasks. It should be noted that these profiles can be encoded off-line by human experts or they can be learned using machine learning or knowledge mining techniques [22].

If no relevant cases are retrieved (or too many cases are retrieved) using the initial context, then the system automatically modifies the context. It is apparent that after the context is changed by relaxation or restriction, the system must reevaluate the query. A naive approach would take the new query and would submit it to the system. A more sophisticated approach can take advantage of an already processed query by incrementally modifying its result. Incremental approaches have been successfully used in database systems for view maintenance [8,9,10].

In most situations, an incremental approach is substantially more efficient than a naive one. Efficiency improvement is more visible when several consecutive changes to the query are required, i.e., during iterative browsing, and when a large case base is used. Small updates to the query generally produce only small changes to the query result. Thus, only local changes to the query are required if an incremental approach is used [10].

The basic idea of incremental computation in ***TA3*** is to store query results and reuse them when similar queries are computed, similarly as in [6]. If the number of attributes per case is significantly smaller than the number of cases then the incremental context restriction/relaxation outperforms recomputation of the answer from scratch. ***TA3*** supports two types of incremental context modification:

1. partial results are kept at the attribute level;
2. partial results are kept at the category level.

It is obvious that the first approach poses more requirements on storage. However, it is more versatile and thus is more suitable for case bases with frequent context modifications and with fewer attributes per case. The second approach is a compromise between performance gains from an incremental approach and modest storage requirement. This approach is useful for less frequent context changes and also for case bases containing cases with many attributes.

- initial evaluation is the same as for $\text{retrieval}_{\text{naive}}$ except that partial matching results are stored into SC_i

```
retrieval_incremental (Ω, CB)
    C_input → Ω
    for (i = 1; i < |Ω|; i++)
        if SC_i does not exist for a given attribute in Ω
            then generate SC_i using sat(C_ij, Ω_i); j = 1..|Ω|
        else continue
    end
    SC ← {SC_i ∪ SC_{i+1}}_{i=1}^{|CB|-1}
    return all cases in SC
end
```

Figure 3. Incremental retrieval algorithm.

Consider the naive and incremental algorithms presented in Figures 2 and 3. Using a naive approach, it is apparent that *k* iterative context modifications require *k* algorithm evaluations. However, using an incremental approach requires only one algorithm evaluation. For a given context, all matching cases are included in *SC* (a set of cases) that is obtained by evaluating SC_{part} using an *eval* function. This function evaluates a partial result of the matching, composed of the partial results of the existing cases SC_j: $SC = eval(SC_{part}) = \{SC_j \cup SC_{j+1}\}_{j=1}^{|CB|-1}$, where |CB| is the number of cases. Each partial result comprises partial results on the attribute level: $SC_j = \{SC_{ji} \cap SC_{j(i+1)}\}_{i=1}^{n-1}$, where *n* is the number of attributes.

The idea of incremental context relaxation and restriction is based on the use of differential queries [5]. First, it is determined what parts of the context are affected by the change, i.e., what parts of context are relevant to the change. Second, only the affected parts are recomputed. If the attribute *p* is removed from context, then $\Omega' = \Omega - \Omega_{change}$, where:

$$\Omega_{change} = a_k : SC' = eval(SC'_{part}),$$
$$SC'_{part} = SC_{part} - SC_{jp} = SC'_{part} = SC_{part} - SC_{jp} =$$
$$= \{(SC_{j1} \cap SC_{j2} \ldots SC_{j(p-1)} \cap SC_{j(p+1)} \ldots SC_{jn}) \cup$$
$$(SC_{(j+1)2} \ldots SC_{(j+1)(p-1)} \cap SC_{(j+1)(p+1)} \ldots SC_{(j+1)n})\}_{j=1}^{d-1}$$

If the category *q* is relaxed to *Some(r),* then $\Omega' = \Omega_0 \cup \Omega_1 \cup \ldots \Omega_l = \bigcup_0^l \Omega_i$, where

$$l = \left(\frac{n}{n-k}\right):\ SC'_{part} = \left\{\begin{matrix} \left(SC_{j1} \cap_r SC_{j2} \ldots SC_{j(p-1)} \cap_r SC_{j(p+1)} \ldots \cap_r SC_{jn}\right) \cup \\ \left(SC_{(j+1)1} \cap_r SC_{(j+1)2} \ldots SC_{(j+1)(p-1)} \cap_r SC_{(j+1)(p+1)} \ldots \cap_r SC_{(j+1)n}\right) \end{matrix}\right\}_{j=1}^{d-1};$$

$\cap_r$ means that at least *r* attributes for the particular category must match. Cardinality restriction is performed analogously, by adding an attribute into the context. Value restriction and relaxation are based on the same principle.

3.3. The Adaptation Module

The adaptation module takes all relevant cases as an input and produces a solution for a given problem as an output. In the current case studies, an automatic adaptation has been used in the form of a weighted average of attribute values from retrieved cases. In other words, retrieved cases are ordered according to relevance measure, giving the most relevant cases the highest weight. An average value for desired attributes (e.g., time rise in a servo domain) is computed. It should be noted that in some domains the automatic adaptation might not be possible/desirable because of existence of nonnumeric and complex attribute values.[2] Other domains may have specific adaptation rules that can improve accuracy by providing more domain knowledge.

Another possibility for improving adaptation results is to consider the current modification in context with other cases or other modifications. Thus, when planning an end-effector movement along a certain trajectory, some of the relevant cases retrieved might be discarded based on the information from previous steps. This allows for discarding cases with similar descriptions but different solutions.

3.4. The Evaluation Component

The evaluation component considers produced solution and feedback, which can be generated by the underlying problem solver (an inverse kinematic task solver or a path planner), by other parts of the system (sensors) or by the user during the training phase. If there is a disparity between them, the case base and/or context modification strategies are updated in order to prevent inaccurate solutions in the future. The process is as follows:

---

[2] It should be noted that if concept hierarchies are available in a particular domain, adaptation can be automatically performed using the same principle of generalization/specialization as described in \S{sec-context.

- if several relevant cases have been used to produce a solution and the solution was not correct, a given problem is solved by other means and is stored as an exception;
- if one of the retrieved cases proposes a different solution than the majority, the adaptation strategy is modified so that this case is no longer relevant in a given situation.

4. Case Studies

In this section we demonstrate the utility of the proposed system on several real world examples; namely, learning control, object recognition, system design and fault diagnosis:

1. **Learning Control:** We have tested the ***TA3*** system on two control domains [14,15]:
   - predicting the rise time of a servomechanism in terms of two gain settings and two choices of mechanical linkages – ***TA3***$_{\text{SERVO}}$, and
   - predicting the joint angles for a three-link spherical, angular robot when given desired end-effector coordinates – ***TA3***$_{\text{IKT}}$.
2. **Object Recognition:** In order to support the thesis that ***TA3*** can be used for object (or scene) recognition we have applied our prototype to character recognition domain [22,15].
3. **System Design:** Besides classification task, ***TA3*** can also be used during designing complex robotic systems [24].
4. **Fault Diagnosis:** Classification [16] and prediction [17,18] are similar tasks to fault diagnosis. Having a description of fault cases (i.e., fault symptoms with a fault class) stored in a case base ***TA3*** can asses similarity between new set of symptoms and source cases.

4.1. Case Study 1: Servo Domain

Servo Domain is a publicly available data set created in 1986 by Karl Ulrich at MIT.[3] Due to its availability and nontriviality, the database was used for testing of various machine learning algorithms, e.g., [33,34]. Servo Domain covers an extremely nonlinear phenomenon – predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages.

4.1.1. Case Base

Servo case base contains 167 cases with 5 attributes per case:

Attribute Name	Attribute Values
motor	A,B,C,D,E
screw	A,B,C,D,E
pgain	3,4,5,6
vgain	1,2,3,4,5
rise time	a real number from 0.13 to 7.10

To create a Servo case base, a simulation of a servo system involving a servo amplifier, a motor, a lead screw/nut and a sliding carriage was used. Case were created during problem solving by storing problem/solution pairs – case base generation by observation. The output

[3] ftp://ics.uci.edu/pub/machine-learning-programs/servo/

value of the servo system simulation represents the time required for the system to respond to a step change in a position set point (a rise time).

4.1.2. Evaluation of Retrieval Function

Even though Servo is not a complex (in terms of number of attributes) or a large (in terms of number of cases) domain, it is of interest since it involves continuous class prediction and various machine learning algorithms have been evaluated and compared with respect to it. Table 1. summarizes results for two versions of ***TA3***$_{\text{SERVO}}$ and other machine learning techniques, obtained by using simple cross-validation (leave-one-out method) and averaged over 10 and 60 random trials respectively. Errors for one trial were determined by removing a random case from the case base, using it as an input and predicting its rise time. Obtained value is compared to the actual value to determine prediction error.

Method	Average Abs. Error	Relative Error
Guessing mean	1.15	1.00
Instance-based	0.52	0.26
Regression	0.86	0.49
Model trees	0.45	0.29
Neural nets (G. Hinton)	0.30	0.11
Regression+instances	0.48	0.20
Model trees+instances	0.30	0.17
NN+instances	0.29	0.11
TA3$_{\text{SERVO-1}}$ (60 trials)	0.118	0.073
TA3$_{\text{SERVO-2}}$ (60 trials)	0.09	0.056

Table 1. Servo-mechanism domain – Absolute and relative errors on predicting rise time. Results obtained by simple cross-validation; average over 20 random trials.

Relaxation strategy 1 was used to obtain the results for ***TA3***$_{\text{SERVO-1}}$, i.e., only the *motor* attribute was relaxed and only to the left side (e.g., B would be relaxed to A or B). ***TA3***$_{\text{SERVO-2}}$ results were obtained by using relaxation strategy 2, i.e., only the *motor* attribute was relaxed but in both directions (e.g., B would be relaxed to A or B or C). It should be noted that relaxing A and E would be the same in both scenarios. In a situation where the initial relaxation did not yield a result, additional methods are used. First, the cardinality relaxation for *motor* and *screw* is tried. Second, the cardinality relaxation for *pgain* and *vgain* is performed.

Although (direct) error comparison can be used for performance evaluation and system comparison, it is not sufficient in general. In the data presented in Table 1 there is no information on the confidence in the results, nor is there any information about the significance of the errors or about the confidence intervals. In other words, statistical evidence is missing. Thus, besides the comparison presented in Table 1 we also conducted a statistical evaluation of our system, the results of which are presented in Table 2.

Method	Average Abs. Error 99% Confidence	Relative Error 99% Confidence
TA3$_{\text{SERVO-1}}$ (60 trials)	0.118 ± 0.042	0.073 ± 0.073

TA3$_{SERVO-2}$ (60 trials)	0.09 ± 0.035	0.056 ± 0.012

Table 2. Servo-mechanism domain – Absolute and relative errors on predicting rise time. Results obtained by simple cross-validation; average over 60 random trials and using two different retrieval strategies and two confidence level intervals.

Table 1 and 2 show that it is advantageous to have the option of relaxing the constraints on attribute values, since ***TA3***$_{SERVO-2}$ obtained more accurate results compared to ***TA3***$_{SERVO-1}$. Moreover, considering the statistical confidence of the result, it was observed that 40 trials are not sufficient for 99% confidence with a reasonable small confidence interval. Thus, an average from 60 trials was used.

4.2. Case Study 2: Inverse Kinematic Task

IKT robotic domain covers an extremely nonlinear phenomenon – predicting the joint angles for the three-link spherical, angular robot when given desired end-effector coordinates (an inverse kinematic task). Spherical, angular robot is a universal robot with a large working area (see Figure 4). Since for this task we had robot parameters and necessary equations, we created several case bases using different criteria (varying database size, distribution of instances, etc.). Due to the practical nature of these problems, data availability and objective relevance measure, similar tasks have been used to test machine learning algorithms in the past [3].

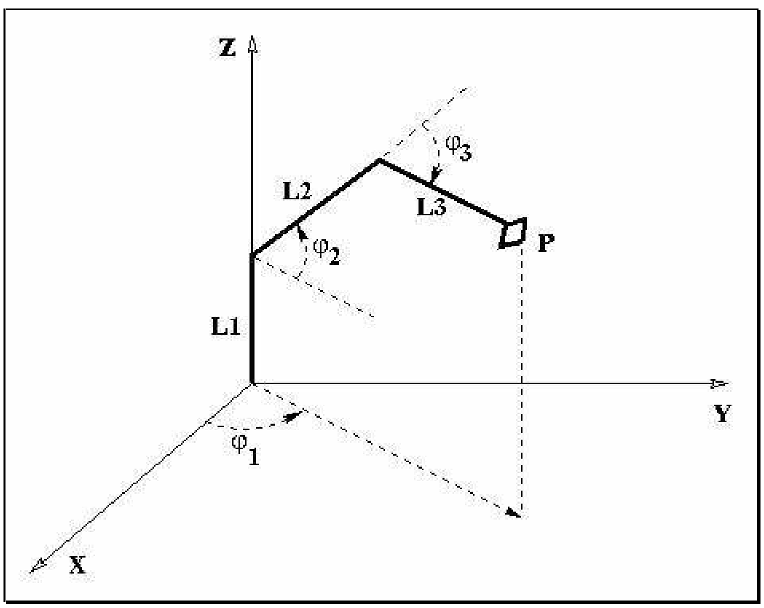


Figure 4. Spherical angular robotic system.

The task under consideration is known as an inverse kinematic (e.g., computation of the robot's joint angles to ensure that the end-effector coordinates will reach a desired position). This is a complex task that might be computationally intractable for complex systems. The complexity increases with the number of robot arms: with just two arms and with no additional constraints, there are at least two solutions for the inverse kinematic task. In such a situation, other methods are used to approximate the solution or to constraint the task. Approximation methods include fuzzy and neural net control [38,7]. The search space can also be constrained, for example, by limiting the energy consumption, maximizing the speed or maximizing the accuracy.

In our study, we used an existing small robot as an example. The robot's parameters are presented in Table 3, where L_1, L_2, L_3 denote the lengths of the links (robotic arms), $\varphi_1, \varphi_2, \varphi_3$ are the ranges for joint angles.

L_1	L_2	L_3	φ_1	φ_2	φ_3
0.203m	0.178m	1.178m	$\left(-90^o, 90^o\right)$	$\left(-35^o, 145^o\right)$	$\left(-150^o, 0^o\right)$

Table 3. Robot characteristics. L_1, L_2, L_3 are robot links; $\varphi_1, \varphi_2, \varphi_3$ are joint angles between links.

For the given architecture, the following equations are used to compute the end-effector coordinates *P* (a forward kinematic task):

$$P_x = L_2 \cos\varphi_1 \cos\varphi_2 + L_3 \cos\varphi_1 \cos\left(\varphi_2 + \varphi_3\right)$$
$$P_y = L_2 \cos\varphi_1 \cos\varphi_2 + L_3 \sin\varphi_1 \cos\left(\varphi_2 + \varphi_3\right)$$
$$P_z = L_1 + L_2 \sin\varphi_2 + L_3 \, s\mathit{in}\left(\varphi_2 + \varphi_3\right)$$

The inverse kinematic computation does not necessarily yield only one solution. Moreover, complex kinematic structures might not have known analytical solution at all. In such situations, some optimality criteria need to be defined and used for computation. The other possibility is to use previous knowledge and compute only increments from the current position. Nevertheless, the following are equations for inverse kinematic of our architecture:

$$\varphi_1 = \arctan\frac{P_y}{P_x}$$
$$\varphi_2 = \arctan\frac{P_z - L_1}{\sqrt{P_x^2 + P_y^2}} \pm \arccos\frac{L_2^2 - L_3^2 + P_x^2 + P_y^2 + (P_z^2 - L_1)^2}{2L_2\sqrt{P_x^2 + P_y^2 + (P_z^2 - L_1)^2}}$$
$$\varphi_3 = \pm\left[\pi - \arccos\frac{L_2^2 + L_3^2 - P_x^2 - P_y^2 - (P_z^2 - L_1)^2}{L_2 L_3}\right]$$

The obvious need for an inverse kinematic task is during planning – a robot's end-effector is in the point *X* and should be moved to point *Y*. For example in welding, the usual task is to follow a particular curve during the process. Thus, not only must the inverse kinematic be computed, but it must be computed reasonably fast. Using a simple example, we propose that by storing solutions to IKT and using case-based reasoning a reasonable performance can be achieved while allowing for an easy increasing of movements repertoire.

4.2.1. Case Base

The IKT case base contains 2,000 cases with 9 attributes per case:

Attribute Name	Attribute Meaning	Attribute Values
P_x	x coordinate of the end-effector	float
P_y	y coordinate of the end-effector	float
P_z	z coordinate of the end-effector	float
L_1	length of the first link	float
L_2	length of the second link	float
L_3	length of the third link	float
φ_1	base angle	float
φ_2	shoulder angle	float
φ_3	elbow angle	float
X, Y, Z	coordinate frame	float

In the first experiment, the whole knowledge base was created uniformly, i.e., all 2,000 cases were uniformly distributed over the working space of the robot. This is especially useful for off-line learning to guarantee a reasonable answer in any possible situation.

The other possibility is to generate cases randomly, which is not useful in general. The main reason is that it does not represent the real world situation, because rarely there is just a random move of the robotic arm in its working space.

The last possibility is to "observe" actual cases while the robot is performing normal tasks. The problem with this approach is that it requires on-line operation and that most probably the working space will not be represented evenly with the cases. Thus, the system will be able to work in the similar situations to the ones observed during learning, but the performance will degrade if there is a change in a robot task. Off-line generated knowledge base can be extended by on-line learning. Off-line generated knowledge base can be used for generalizations and for knowledge-based adaptation of retrieved cases. On-line learning can provide useful experience in two ways:

1. recency information defines preferences among existing cases and
2. new information allows for graceful task extension or change.

4.2.2. Evaluation of Retrieval Function

During system evaluation we varied several parameters, such as size of the case base, density of individual cases, distribution of individual cases and we used various adaptation techniques while we also varied the relaxation process. During evaluation we used simple cross-validation (leave-one-out method) and we computed significance intervals and variances from the averaging 20 random trials (see Table 4).

	Average Abs. Error			Relative Error		
Method	φ_1	φ_2	φ_3	φ_1	φ_2	φ_3
TA3$_{IKT-1}$	0.4855 ± 0.2128	0.0	0.0	0.00586863 ± 0.0025	0.0	0.0
TA3$_{IKT-2}$	0.0	2.5 ± 1.096	0.0	0.0	0.0274725 ± 0.012	0.0

Table 4. Inverse kinematic task – Absolute and relative errors on predicting joint angles $\varphi_1, \varphi_2, \varphi_3$. Results obtained by simple cross-validation; average over 20 random trials and with 95% confidence intervals.

In ***TA3***$_{IKT-1}$ only cardinality relaxation was used (i.e., only *m* of *n*, *m*<*n*, attributes were required to match). A similar approach presented in [32] yielded analogous results – accuracy improvement for *m*<*n*. ***TA3***$_{IKT-2}$ used value relaxation (i.e., relaxing attribute values) first and if this failed then cardinality relaxation was performed. It is interesting to note that ***TA3***$_{IKT-1}$ outperformed ***TA3***$_{IKT-2}$, although it uses a domain independent relaxation method.

Similar to IKT, a case-based approach can be utilized for other control tasks [35]. The ***TA3*** system differs from traditional knowledge-based approaches to learning control by its approach to similarity assessment. It allows for flexibly changing the priorities – from precision-oriented control to real-time control, by deploying anytime algorithm [15,23].

4.3. Other Case Studies

4.3.1. Character Recognition

Character recognition is a data set created in 1991 by David J. Slate and it is publicly available on a machine learning database server, along with references to previous experimental evaluation of various machine learning algorithms on this domain.[4]

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. Usually, systems are trained on the first 16,000 items. Accuracy is measured by applying resulting model to predict the letter category for the remaining 4,000 letters.

The case base consists of 20,000 cases described by a letter (class) and 16 integer-type attributes (horizontal and vertical position of a box, width and height of a box, total number of pixels, mean x/y of pixels in a box, mean x/y variance, mean xy correlation, etc. Thus, there is 26 possible target classifications (letter A to Z). The case base consists of 20 prototypes per class with uniform distribution of instances among concepts and normal distribution of instances among prototypes.

For the classification task the system is presented with 16 attributes describing a letter and the task is to identify the letter, based on previously seen cases. Results presented in Table 5 have been obtained using simple cross-validation (leave-one-out method) and averaged over twenty random trials.

Method	**Accuracy** (20,000 cases)	**Accuracy** (2,000 cases)
Backpropagation	–	81.9 ± 0.6%
IB1	95.7 ± 0.4%	81.7 ± 0.7%
CN2	87.9 ± 0.8%	68.7 ± 1.0%
C4	86.4 ± 0.7%	67.4 ± 0.8%
Genetic Algorithm	82.7	–

[4] ftp:ics.uci.edu:pub/machine-learning-programs/

$TA3_{letter-1}$ (20 trials)	90% ± 0.1461	80% ± 0.2011
$TA3_{letter-2}$ (20 trials)	100%	85% ± 0.1742
$TA3_{letter-3}$ (20 trials)	100%	95% ± 0.1001
$TA3_{letter-4}$ (20 trials)	100%	100%

Table 5. Character recognition domain – absolute and relative errors on servo-data using simple cross-validation; average over 20 random trials (95% confidence intervals)

In Table 5. all algorithms but ***TA3***$_{letter}$ were trained on the first 16,000 and 1,600 cases respectively and then tested on the last 4,000 cases. Reported results are average values over 10 trials. Since ***TA3***$_{letter}$ does not require a learning period, we used leave-one-out testing method on the first 2,000 and on all 20,000 cases. Results are average values over 20 random trials.

In ***TA3***$_{letter-1}$ we used a modified nearest-neighbor approach – attributes were grouped into categories for selective cardinality relaxation/restriction. ***TA3***$_{letter-2}$ was obtained by using a value relaxation first (i.e., relaxing attribute values to include values of immediate neighbors). If more cases were needed, cardinality relaxation was performed. ***TA3***$_{letter-3}$ was obtained by using combination of previous approaches with equal voting for each of them, i.e., the final answer was composed of the two so that the majority vote wins. ***TA3***$_{letter-4}$ was obtained by using different category groupings for attributes. In the previous test, categories were created sequentially, each group having four attributes (letter attribute was an extra category). In the latter test we used knowledge mining techniques [22] to find out the most relevant attributes in recognizing individual letters and we grouped attributes accordingly.

The presented example shows how ***TA3*** can be used for recognition task. Its architecture requires image pre-processing module though. Once the visual input is pre-processed ***TA3*** can support robot vision for planning or recognition subsystem.

4.3.2. Robot Design

We have applied the ***TA3*** system into the task introduced in [25]. The robot is described by four parameters: cost, load capacity, velocity and repeatability. The presented example consisted of 28 robot cases. Even though the case base is quite small the example enables us to illustrate how the proposed system can be utilized for the design task. The task is to find a suitable robot in a given situation, i.e., to find a robot with parameters in a specified range or parameters similar to specified example.

It is hard to compare the experimental evaluation of our system with the system presented in [25] since the case base size is so small. However, considering the robot similarities, we have been able to show that ***TA3*** approach to flexible retrieval is useful for determining a suitable robot.

In addition, we have extended a given example in two ways:

1. Considering a more realistic and complex robot representation (namely, individual robots were represented by 107 attributes). This included other parameters, such as considered robotic environment and type of work to be performed by the robot, robot type, needed accuracy and precision, required movements, etc.
2. Acquiring more robot cases from a catalogue [], namely about 100 new robots.

The work on extending this application is ongoing [24].

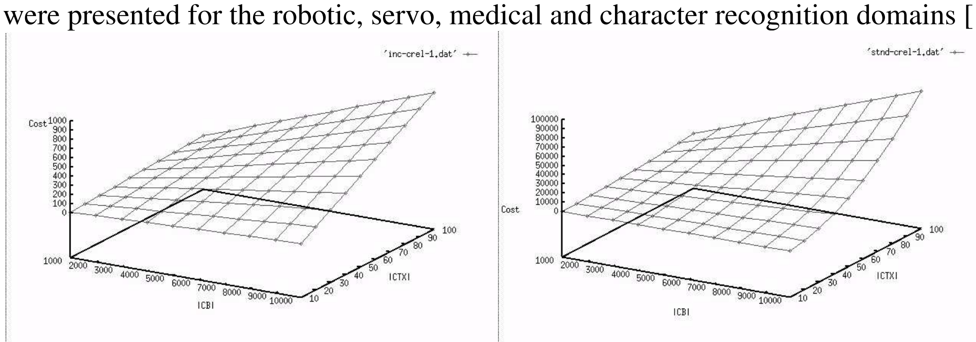
4.4. Summary on Case Studies

Besides a control task in robotic domains [14], we have applied the ***TA3*** system to other domains: prediction and knowledge mining in medicine [18,17], character recognition [22,15], supporting software reuse [21], iterative browsing and intelligent retrieval [20]. Each domain has different characteristics and thus allows for a better performance evaluation. In presented studies, we have measured not only task-performance (accuracy) but time-performance (scalability) as well. Especially in the robotic domain, the results must not only be accurate but also, they must be delivered in real time. Time constraints in the manufacturing process demand fast and accurate results.

There are several factors affecting performance: the size of a case base, size of a case and context, query complexity and the relaxation/restriction strategy used. We considered average queries and varied the case base size and the context-modification strategies. The computer time (in ms) required to perform three consecutive context modifications was measured, considering the same query for different case base sizes.[5]

Comparing naive and incremental algorithms shows that *k* consecutive context modifications require *k* applications of the naive algorithm. Thus, a naive approach requires $k \times |CB| \times |\Omega|$ evaluations. In contrast to this, an incremental algorithm is evaluated only once to produce SC_{part} and subsequent context modifications are handled by incremental changes to the result. Thus, assuming that the size of the initial retrieval result $|SC_{part}|$ is substantially smaller than the case base size $|CB|$, producing *SC'* incrementally is significantly more efficient. For all categories that are not affected by a context modification, no recomputation is necessary. For categories affected by a change, local changes handle the recomputation as presented in [15]. It should be noted that both naive and incremental algorithms can be improved by indexing which avoids accessing all cases in a case base; however, this is not a simple issue and is beyond the scope of this paper.

Figure 5. shows how value/cardinality relaxation and restriction depend on the case base size, $|CB|$ and context size $|\Omega|$. We compare both standard and incremental approaches during value and cardinality relaxations, as well as during value and cardinality restrictions. Experimental results were presented for the robotic, servo, medical and character recognition domains [15,16].



[5] ***TA3*** is implemented in C++ on a SUN workstation. Presented experiments were obtained on the SparcStation IPC with 32MB of memory.

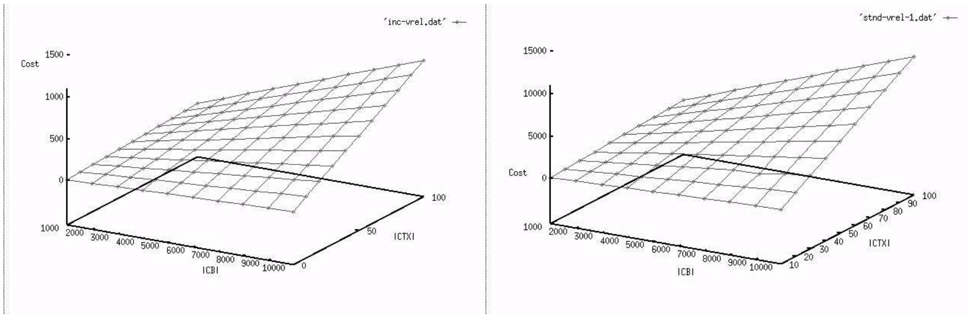


Figure 5. Cost of retrieval for ten context modifications.

Presented results support the claim that the incremental context manipulation is more efficient than the naive approach. Performance improvement is increased when the case base size is substantial, cases have many attributes and several subsequent context modifications are required. However, for simple retrievals (case base and/or no consecutive context modifications) the naive approach would be more efficient, due to smaller memory requirements.

5. Conclusions

The results of the experiments suggest the viability of our approach to learning control, recognition, design and diagnostic tasks in robotics. The inverse kinematic task experiment provides motivation for work on robot path planning, where the system not only selects a path for the robot based on the task definition but may also create a sequence of values for joint angles to be followed in order to fulfill the task. The case-based approach to control is suitable for tasks in industry, where many situations are repetitive and thus the system can gain experience over time.

Most successful robotic manipulators are precise, fast, smooth and have high repeatability. Their major drawback is their tendency to have a limited repertoire of movements that can only be extended by costly inverse kinematic calculations or direct teaching [36]. Our approach also starts with a limited repertoire of movements, but it can incrementally add new positions and thus allows for greater flexibility. The combination of on-line and off-line learning with the case-based reasoning system offers both performance and flexibility.

Precision of presented results can be increased by three approaches:

1. using more relevant cases via task-focused case acquisition process;
2. applying more accurate adaptation process using more domain knowledge;
3. combining a CBR approach for an approximate solution and an optimizer for producing an accurate solution.

We plan to experiment with various problems to assess the optimal case base size and the best case distribution. We intend to combine a given approach with a more complex system, such as robot path planning and autonomous robot navigation.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. IEEE Transactions on Knowledge and Data Engineering., 5(6):914-925, 1993.
2. S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation for robots by learning. Journal of Robotic Systems, (1):123-140, 1984.

3. D. Aha and S. L. Salzberg. Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks. In P. Cheeseman and R. W. Oldford, editors, Selecting Models from Data: Artificial Intelligence and Statistics IV, 1994.
4. P. Bondi, G. Casalino, and L. Gambardella. On the iterative learning control theory for robotic manipulators. IEEE Journal of Robotics and Automation, 4(1):14-22, 1988.
5. J. A. Blakeley, Per-Ake Larson, and F. W. Tompa. Efficiently updating materialized views. In ACM-SIGMOD, pages 61-71, 1986.
6. L. Baekgaard and L. Mark. Incremental computation of time-varying query expressions. IEEE Trans. on Knowledge and Data Engineering, 7(4):583-589, 1995.
7. S. N. Balakrishnan and R.D. Weil. Neurocontrol: A literature survey. Mathematical and Computer Modeling, 23(1-2):101-117, 1996.
8. S. Ceri and J. Widom. Deriving production rules for incremental view maintenance. In VLDB-91, pages 577-589, Barcelona, Spain, 1991.
9. T. Griffin and L. Libkin. Incremental maintenance of views with duplicates. In Proc. of the ACM SIGMOD International Conference on Management of Data, pages 328-339, San Jose, CA, 1995.
10. A. Gupta, I. S. Mumick, and K. A. Ross. Adapting materialized views after redefinitions. In Proc. of the ACM SIGMOD International Conference on Management of Data, pages 211-222, San Jose, CA, 1995.
11. R. Greiner. AAAI Fall Symposium Series on Relevance. AAAI Press, 1994.
12. K. P. Jantke. Nonstandard concepts of similarity in case-based reasoning. In Proc. of the 17th Annual Conference of the Gesellschaft für Klassifikation e.V., Kaiserslautern, Germany, 1994. Springer Verlag.
13. T. J. Jang, C. H. Choi, and H. S. Ahn. Iterative learning control in feedback systems. Automatica, 31(2):243-248, 1995.
14. I. Jurisica and J. Glasgow. Applying case-based reasoning to control in robotics. In 3rd Robotics and Knowledge-Based Systems Workshop, St. Hubert, 1995.
15. I. Jurisica and J. Glasgow. A case-based reasoning approach to learning control. 5th International Conference on Data and Knowledge Systems for Manufacturing and Engineering, DKSME-96, Phoenix, Arizona.
16. I. Jurisica and J. Glasgow. Case-based classification using similarity-based retrieval. In 8th IEEE International Conference on Tools with Artificial Intelligence, Toulouse, France, 1996.
17. I. Jurisica, J. Mylopoulos, J. Glasgow, H. Shapiro, and R. F. Casper. Case-based reasoning in IVF: Prediction and knowledge mining. AI in Medicine. Special issue on CBR in Medicine, 1996. Under revisions.
18. I. Jurisica and H. Shapiro. A computer model for case-based reasoning in IVF. In The 51st Conf. of the American Society for Reproductive Medicine, Seattle, Washington, 1995.
19. I. Jurisica. How to retrieve relevant information? In [11], pages 101-104, 1994. An extended version available as a Technical. Report DKBS-TR-94-5, University of Toronto, Department of Computer Science.
20. I. Jurisica. ***TA3***: Case-based intelligent retrieval and advisory tool. In ACM Conference on Society and the Future of Computing, Durango, CO, 1995.
21. I. Jurisica. A similarity-based retrieval tool for software repositories. In: The 3rd Workshop on AI and Software Engineering: Breaking the Mold. IJCAI-95, Montreal, Quebec, 1995.
22. I. Jurisica. Inductive learning and case-based reasoning. In Canadian AI Conference, Workshop on What is inductive learning?, Toronto, Canada, May 1996.
23. I. Jurisica. Supporting flexibility. A case-based reasoning approach. In The AAAI Fall Symposium. Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities, Cambridge, Massachusetts, 1996.

24. I. Jurisica. Designing complex robotic systems. A case-based reasoning approach. Technical Report. University of Toronto, Department of Computer Science, In preparation, 1996.
25. M. Khouja and D. E. Booth. Fuzzy clustering procedure for evaluation and selection of industrial robots. Journal of Manufacturing Systems. 14(4): 244-251, 1995.
26. J. Kolodner. Case-Based Reasoning. Morgan Kaufmann, 1993.
27. D. Luzeaux and B. Zavidovique. Process control and machine learning: Rule-based incremental control. IEEE Trans. on Automatic Control, 39(6):1166-1171, 1994.
28. S. N. Minton. Quantitative results concerning the utility of explanation-based learning. In Proc. of AAAI-88, pages 564-569, St. Paul, MN, 1988. Also published in Journal of Artificial Intelligence, 42, 1990, pages 363-381.
29. T. M. Mitchell. Becoming increasingly reactive. In Proc. of the Workshop on Innovative Approaches to Planning, Scheduling and Control, San Diego, CA, 1990.
30. J. Mylopoulos and R. Motschnig-Pitrik. Partitioning information bases with contexts. In The 3rd Int. Conf. on Cooperative Information Systems, Vienna, 1995.
31. T. Naniwa and S. Arimoto. Learning control for robot tasks under geometric endpoint constraints. IEEE Trans. on Robotics and Automation, 11(3):432-441, 1995.
32. J. Ortega. On the informativeness of the DNA promoter sequences domain theory. Journal of Artificial Intelligence Research, 2:361-367, 1995. Research Note.
33. R. Quinlan. Learning with continuous classes. In Proc. 5th Australian Joint Conference on AI, pages 343-348, 1992.
34. R. Quinlan. Combining instance-based and model-based learning. In Proc. Of the 10th International Conference on Machine Learning, Amherst, MA, 1993.
35. A. Ram and J. C. Santamaria. Continuous case-based reasoning. In Proc. of the AAAI-93 Workshop on Case-Based Reasoning, Washington, DC, 1993.
36. K. Sasaki. History of robot control research. International Journal of the Japan Society for Precision Engineering, 29(3):197-200, 1995.
37. F. Smieja, U. Beyer, and G. Richter. Adaptive control of a robot arm using driver programs. Artificial Intelligence in Engineering, 10(1):39-59, 1996.
38. S. G. Tzafestas. Neural networks in robot control. Artificial Intelligence in Industrial Decision Making, Control and Automation, 14:327-387, 1995.
39. Wang, Y. C. Soh, and C. C. Cheah. Robust motion and force control of constrained manipulators by learning. Automatica, 31(2):257-262, 1995.