Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1998), *12*, 107–115. Printed in the USA.
Copyright © 1998 Cambridge University Press 0890-0604/98 $12.50

Learning to verify design solutions from failure knowledge

PAULO GOMES,[1] CARLOS BENTO,[1] and PEDRO GAGO[1,2]

[1]CISUC–Centro de Informática e Sistemas da Universidade de Coimbra, Polo II, 3030 Coimbra, Portugal
[2]Escola Superior de Tecnologia e Gestão—Instituto Superior Politécnico de Leiria, Morro do lena, 2400 Leiria, Portugal

(Received June 27, 1997; Revised October 17, 1997; Accepted November 10, 1997)

Abstract

Learning is an intrinsic product of case-based reasoning. Acquiring new cases is one possible way of learning in a case-based system. These cases comprise mainly success knowledge. The successful cases are essentially used to generate new design solutions. But a case-based system also can make use of failure knowledge. In this paper we present how a case-based system can acquire failure cases for verification of the solution created by success cases. We describe IM-RECIDE, a system that uses case-based reasoning for solving design problems that are imperfectly described and explained. The learning aspect is focused and some of the machine learning dimensions in design are criticized. Experimental results in the domain of room configuration also are presented.

Keywords: Case-based Reasoning; Design; Failure Knowledge; Learning

1. INTRODUCTION

Design is a cognitive task in which a structure is generated to comply with a set of constraints and functional requirements (Tong & Sriram, 1992). Design can be further divided in routine and nonroutine designs (Gero, 1994). The term *routine design* is used when the designer knows all of the necessary knowledge. Routine design involves the generation of instances of well-known design classes. *Nonroutine design* is divided in two subclasses: innovative and creative design. In innovative design, the space of possible designs is extended. This occurs when a new artifact differs from the instances of the same class in some substantive way. This difference can be in terms of particular value(s) assigned to the design variables that are not commonly used. Creative design expands the space of possible designs through the use of new design variables.

In solving design problems, designers make use of old solutions. While in routine design old solutions are applied nearly unchanged, nonroutine design uses old designs in novel ways. The reuse of designs in nonroutine design involves structural and behavioral modifications to the old cases.

Case-based reasoning (CBR) uses old solutions to solve new problems (Kolodner, 1993; Maher et al., 1995). This paradigm of artificial intelligence is the basis of various creative design systems (Kolodner & Wills, 1993; Gomes et al., 1996). In a CBR system, learning is a crucial aspect, especially if the domain of application is not well defined and there are no strong domain models. This is the case of creative design, the domain of application of IM-RECIDE,[1] a case-based reasoning shell.

There are various types of knowledge that a CBR system can learn. The most usual one is success knowledge in the form of cases. However, this type of knowledge is mainly for the generation of new designs. An important ability of a system that generates new knowledge is the verification of the generated knowledge. IM-RECIDE solves this problem by learning failure cases (Hammond, 1986) that can be used as constraints. These cases allow the validation and evaluation of created designs. Successful cases also are learned by the system, but these are used for the generation of new knowledge. Both types of cases are important in the improvement of the system performance. They are also important in creative design, because while the success cases

Reprint requests to: Paulo Gomes, CISUC–Centro de Informática da Universidade de Coimbra, Polo II, 3030 Coimbra, Portugal. E-mail: pgomes@eden.dei.uc.pt

[1]IM-RECIDE—IMaginative REasoning with Cases Imperfectly Defined and Explained

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press

provide new knowledge to generate new solutions, failure cases provide the means to restrict the bizarre solutions created by the system, keeping the new solutions within acceptable boundaries.

The IM-RECIDE system, which is a case-based system that implements our model of creative design, is presented in Section 2. The system is described with the focus on case representation, reasoning cycle, case retrieval, and solution generation. Section 3 explains the role of failure cases in IM-RECIDE. Section 4 describes the learning process, showing how success and failure cases are learned. The evaluation of the influence of the learning of failure cases is reported in Section 5. This section shows experimental results in the room configuration domain and analyzes them. In Section 6, our work is classified along the proposed dimensions of machine learning in design. Finally, Section 7 contains some final remarks about the importance of learning failure cases in IM-RECIDE.

2. IM-RECIDE

This section describes IM-RECIDE and its architecture. We first describe the reasoning cycle of IM-RECIDE. Because the case representation determines to a great extent the capabilities of the entire system (Maher et al., 1995), the case representation used is also presented. Understanding the case representation allows a better understanding of what might be learned by the system. However, a CBR system also depends on the recall and reuse of old experiences. We also describe these two important parts of IM-RECIDE in this section.

2.1. Reasoning cycle

The reasoning cycle comprises several steps: problem definition, initialization, problem solving, verification, and evaluation (see Figure 1).

The first step comprises problem specification, where the user states a new problem in terms of goals and constraints. In the initialization phase, the system clusters cases in different sets, each one called a *reasoning space*. These spaces allow gradual exploration of the case library, enabling the system to generate new designs.

In the problem-solving phase, reasoning operators are applied to old solutions for the generation of new ones. If no solution is generated within a specific space, the system switches to the next space from the list that was created during the initialization phase. When a solution cannot be generated, and there are no more spaces to search for, the user is asked to give a solution for the problem.

After a solution has been generated, it has to be verified and evaluated. In a first step, it is validated internally by failure cases (verification). Failure cases represent constraints in the generation of new solutions. If a failure case is triggered by the new solution, then the solution is rejected. If a solution passes the internal validation, the user is asked to accept or reject the new case. If she/he rejects the new solution, the user is asked to explain this rejection in terms of failure cases. After this, the process returns to the problem-solving step. This last phase is called *evaluation*, because the user has to make a decision about the creativity and validity of the case. Although the creativity evaluation is not learned by the system, the validation of a case is learned through the storage of new failure cases provided by the user.

2.2. Case representation

Within IM-RECIDE a case is represented by a triple $\langle P, S, R \rangle$ with P and S, respectively, a set of facts representing the problem and solution descriptions, and R a set of rules representing a causal justification (for a more detailed description, see Bento et al., 1995). A fact is composed by functor and n arguments, with n equal to or greater than zero. The justification is a causal tree, which links problem facts to solution facts through rules.

The case library also comprises failure cases. Failure cases represent design constraints. There are two kinds of failure cases: unsplitable and unmergeable. The syntax for failure cases is the same as the one used for successful cases, with the exception that at the present moment we do not consider the set of causal justifications (R) in them. A more detailed description of the failure cases is given in Section 3.

2.3. Case retrieval

For case retrieval, we consider four spaces of knowledge: Space I, Space II, Space III, and Space IV. Each space comprises the cases possessing a set of common properties concerning the target problem. Each space forms a cluster in

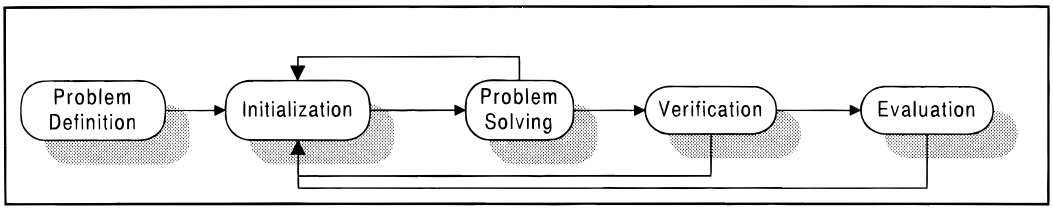


Fig. 1. The principle reasoning steps of IM-RECIDE.

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press

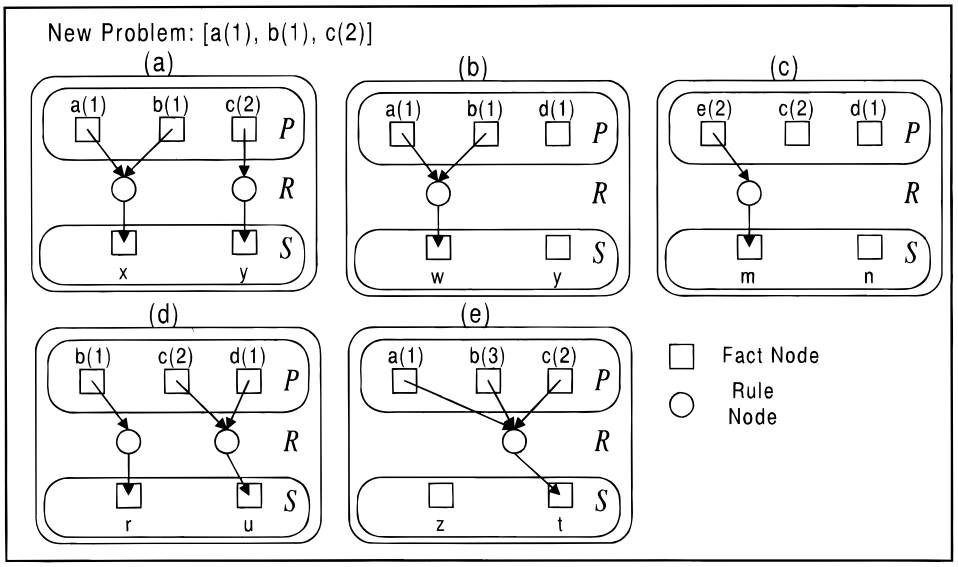


Fig. 2. Example cases of IM-RECIDE.

the case library. Each space is defined in terms of the characteristics that the cases within this space share with the target problem. Creativity can be seen as a result of reasoning on spaces of cases increasingly further away from the target problem. As the system goes from Space I to Space IV, it drifts away from the problem, trying to find nontrivial solutions to the problem. The retrieval process starts with the cases in Space I, going from space to space until it reaches Space IV.

Space I comprises the cases for which all functor/argument pairs belonging to the problem description match the new problem. As an example, consider the case library represented in Figure 2 and the target problem described by the set of functor/argument pairs [a(1), b(1), c(2)]. Case (a) is the single case in Space I because it is the only case in which facts describing the problem are the ones in the new problem. Space I is the space that is normally associated with the current problem. Most of the current CBR systems use cases from this space.

Cases for which all functors describing the problem component match the new problem belong to Space II. The cases from Figure 2, which are in this space, are cases (a) and (e). In these cases, the functors in the problem description are the same functors that describe the new problem. This space is related to problems that are similar to the target problem. This space often is called the *innovation space*, where parametric adaptations are usually done, sometimes resulting in using a novel value for a well-known functor.

Space III contains the cases that have explanation rules with all of the functor/arguments pairs matching the target problem. In the example of Figure 2, space III comprises cases (a), (b), and (d). In case (a), the case pieces comprising explanations [a(1) $\wedge$ b(1) $\Rightarrow x$] and [c(2) $\Rightarrow y$] are the ones responsible for considering this case. The case piece in case (b) responsible for its inclusion in Space III is [a(1) $\wedge$ b(1) $\Rightarrow w$]. Case (d) is also in Space III due to case piece [b(1) $\Rightarrow r$]. Space III is defined by using causal knowledge, which makes similarities between the cases and the target problem more abstract, but also more important. This space usually is associated with creative solutions, but also with bizarre ones.

Space IV gathers all of the cases that contain at least one explanation rule with at least one functor/arguments pair matching the target problem. The cases from Figure 2, which are included in Space IV, are (a), (b), (d), and (e). As an example, case (e) is included because its case piece [a(1) $\wedge$ b(3) $\wedge$ c(2) $\Rightarrow t$] shares a(1) and c(2) pairs with the new problem. This means that there is a chance that fact *t* in case (e) also belongs to the solution for the new problem. Space IV is like a speculation space, where cases have remote similarities with the target problem, because the problem constraints were relaxed. This relaxation allows the system to explore cases considered distant from the target problem. Once again, the knowledge used to do this is the causal knowledge comprised in the explanations.

In the next subsection we describe the adaptation mechanisms used in each space. These adaptation mechanisms are used for case modification to solve the target problem.

2.4. Solution generation

This section describes the adaptation mechanisms, called *adaptation operators*, used in IM-RECIDE. These operators modify cases in the current space to solve the new problem (see Figure 3). The cases that are used for generation of the new case are called the *source cases*. A metric is used for selection of the cases for adaptation (Bento & Costa,

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press

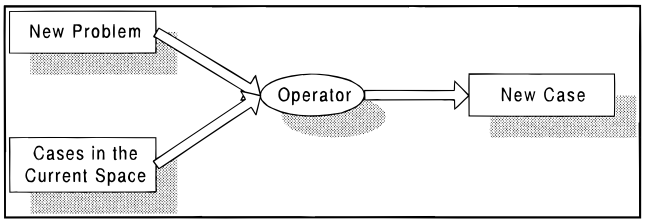


Fig. 3. Inputs and outputs of an adaptation operator.

1994). These cases are selected from the set of cases in the current working space.

Each space possesses a set of predefined adaptation operators. These operators are used according to the type of cases that the space comprises. The operators in Space IV are more powerful than the operators in Space I. This is an obvious situation, because cases in Space IV have less similarities with the problem. To reach a valid solution, more difficult adaptation operations must be done. Associated with the operator capabilities is the complexity of the computational process originated by each operator, the cognitive risks involved, and the probability of generating a more creative solution.

A solution of one case belonging to Space I does not need to be modified to become a solution to the target problem. Therefore, a metric is used for selection of the best case, and its solution is the one for the target problem. Considering again the example in Figure 2, the solution $[x, y]$ for case (a) is the solution for the new problem [a(1), b(1), c(2)].

Cases in Space II have only some different values regarding the problem description. To meet the target problem requirements, it is necessary to modify the old case solution. In Figure 2, one possible solution for the target problem in Space II is $[z, t']$, with t' being a modified fact t. This solution is derived from case (e) by the propagation of the differences in the old problem to the old solution. The causal knowledge is used to guide the propagation process. The selected case is chosen by using a metric function, which measures the similarity of cases against the target problem.

Space III generates new solutions by splitting and merging the case pieces. IM-RECIDE starts selecting the most similar case from the set of episodes comprising Space III. Then it splits the case into pieces, selecting the pieces that match part of the target problem. These pieces then are merged to form a new case. If the problem description in the new case has some missing parts in regard to the target problem, then other cases are selected to contribute with case pieces to complete the new case. Pieces from these cases that are relevant for the new case are merged to the new case. Splitting and merging in Space III is used to generate solution $[w, y]$ for the target problem in Figure 2. Cases (a) and (b) contribute to this solution. Case (a) uses the piece $[c(2) \Rightarrow y]$, while case (b) contributes piece $[a(1) \wedge b(1) \Rightarrow + w]$. This is an alternative solution that results from the exploration of the possible solutions in this space; others also could have been shown. Case (b) is first selected and split into three pieces. Only the piece containing the explanation is used. Then IM-RECIDE looks for cases with the fact c(2) and selects case (a), splitting the case into two pieces and selecting the one with fact c(2).

In Space IV there are several adaptation operators, and they can be applied in sequence. Splitting and merging is one of the operators within this set. The other operators are elaboration, reformulation, substitution, and generalization. Elaboration comprises relaxing and/or strengthening the constraints described in a case problem, to match the target problem description. The case solution is suggested as the new solution. Reformulation involves changing the new problem description according to constraints imposed by the failure cases. Substitution comprises replacing a functor/argument pair in the past case to make it similar to the new problem. The solution that results from this substitution is given as the one for the new problem. Generalization involves considering values initially not considered in the problem description of a past case, and assuming that the case solution remains unchanged. Four possible solutions were created in Space IV: $[r,v]$, $[z, t]$, $[w,v]$, and $[w, t]$. Substitution was used on case (d) to generate solution $[r,v]$. Generalization was applied to case (e) to create solution $[z, t]$. Generation of solutions $[w,v]$ and $[w, t]$ involved a more complex process, in which more than one operator was used. Solution $[w,v]$ was created first by splitting and merging cases (b) and (d) and then relaxing some constraints by elaboration of the new case. The solution $[w, t]$ was generated by splitting and merging cases (b) and (e) and then using generalization in the new case.

After a solution is created, it is verified by the failure cases. If the solution matches one failure case (see the next section for a detailed description), then it is rejected. Only solutions that the system assumes to be correct, by its current knowledge, are shown to the user.

3. USING FAILURE CASES

The failure cases are used during the verification process. IM-RECIDE proposes new cases to the verification module, and this module tries to recall relevant failure cases. The recalled cases are checked to see if they are violated by the proposed new cases. If any of the new cases violate a failure case, then they are automatically excluded from the set of new cases that will be presented to the user as a new solution. In the remainder of this section we will explain how a failure case can be violated by a new case.

As stated before, failure episodes can be unsplitable or unmergeable cases. Three different subtypes of failure cases are considered:

1. Failure cases comprising only the problem description ($P \neq \emptyset$; $S = \emptyset$).
2. Failure cases comprising only the solution description ($P = \emptyset$; $S \neq \emptyset$).

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press

3. Failure cases comprising both descriptions ($P \neq \varnothing$; $S \neq \varnothing$).

We now explain how cases can be interpreted depending on their type.

3.1. Unsplitable cases

Suppose that a generated new case contains P' as its problem description and S' as its solution description, and the unsplitable case that is being matched up against the new case has P and S as problem and solution descriptions, respectively. Then in a situation like case (1), if a fact belonging to P' also belongs to P, then all of the other facts in P must be in P'. In a situation like the one in case (2), if a fact belonging to S' also belongs to S, then all of the other facts in S must be in S'. In a situation like the one in case (3), if P' comprises P, then S' must comprise S.

For example, suppose that the description of the new case is $P' = (a, b, d)$ and $S' = (1,2,3)$, and P and S are defined by Figure 4. Case (1) would be violated by the new case because facts a and b are present in P' and fact c does not belong to P'. Case (2) is not violated because facts 1 and 2 are both present in S'. Case (3) is not violated because P' is different from P. In conclusion, the facts of an unsplitable case must all appear together.

3.2. Unmergeable cases

Suppose that a new case comprises P' as its problem description and S' as its solution description, and the unmergeable case that is being matched up against the new case has P and S as problem and solution descriptions, respectively. Then in the situation considered in case (1), P' cannot comprise P. In a situation like in case (2), S' cannot comprise S. In case (3), if P' comprises P, then S' must not comprise S.

For example, suppose that the description of the new case is $P' = (a, b, c, d)$ and $S' = (1,2,3)$, and that P and S are defined by the failure cases in Figure 4. Case (1) is violated by the new case, because facts a, b, and c are present in P'. Case (2) is violated because facts 1 and 2 are both present in S'. Case (3) is violated because a, b, and c belong to P' and 1 and 2 both belong to S'. In conclusion, the facts of an unmergeable case may not appear together.

4. LEARNING SUCCESS AND FAILURE CASES

IM-RECIDE learns new knowledge by storing new success and failure cases. There are two ways to acquire a new case. The first one occurs when a new case is generated by the system and is considered to be quite different from the cases already in the library. In this situation, the new case also is stored in the case library. Another situation is when the user provides a solution to a problem that the system tried to solve but was not able to come up with a solution that pleased the user. Although new cases can be learned in the above two ways, failure cases can only be acquired from the user.

The criteria for storing a new success case is to store episodes created by the splitting and merging operator, or by the reformulation operator. These two operators are the most powerful ones in IM-RECIDE, and, as a consequence, they produce cases that are different from the cases in the library. If all of the cases were stored in the library, it would degrade its performance because it would comprise many similar or equal cases. Thus, by considering the operators that make cases different from the ones in the library, the space of possible solutions covered by the library cases is increased.

Two different situations can arise when a case (successful or failure) is acquired by the system. One is when the system presents a new solution and it is not accepted. If this happens, the user can explain to the system why the solution is not accepted. The user can do this by giving to IM-RECIDE new failure cases covering the situations that were violated by the new solution. This is a kind of failure-driven learning, because it was the failure situation that triggered it. The second situation is where the system cannot create a solution. If this happens, it will ask the user for a solution. This can be interpreted as a normal learning phase, which is necessary to all systems that learn from experience.

5. EVALUATION

Our application of IM-RECIDE is for the placement of equipment and furniture in a room. A problem comprises: a set of equipment to place in a room; the location of windows and doors; room functionalities; and room type.

A solution describes the objects and their locations in a room to fulfill the problem specifications. For simplicity of the example, below we consider a square room with a window and a door (see Figure 5a). A room comprises nine re-

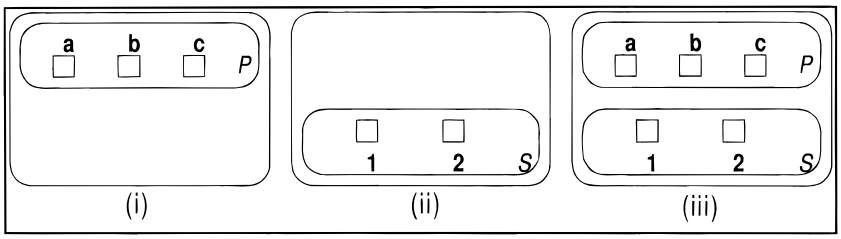


Fig. 4. Examples of failure cases.

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press

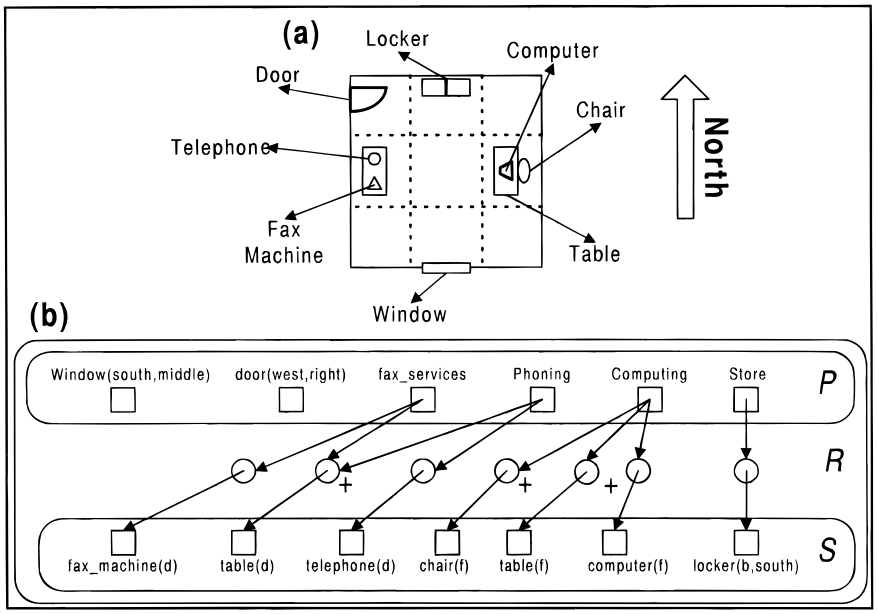
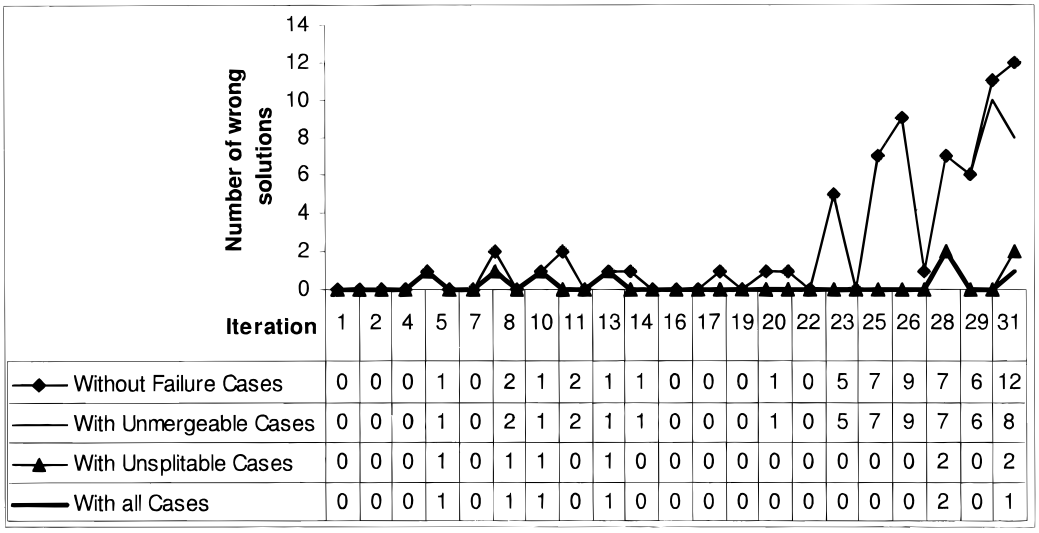


Fig. 5. Graphic representation of room and IM-RECIDE representation.

gions, as represented in Figure 5a. The room is oriented north. Walls are named as east wall, west wall, north wall, and south wall. To specify positions within a wall, we have three wall regions: left, middle, and right. Figure 5b represents the case in Figure 5a.

The influence of the failure cases in the learning process was studied with an experience that compared four different runs of the system. Each run started with no cases in the library. Then, in each iteration a successful case is provided, and if the solutions presented are invalid, new failure cases are provided to the system. In one run only successful cases are provided, another with successful and unsplitable cases, the third with successful and unmergeable cases, and the last with successful, unsplitable and unmergeable cases. A run comprises several iterations, and in each iteration a problem was proposed to IM-RECIDE. After the problem specification, the system gave a set of solutions, and they were classified as right and wrong solutions. At the end of each iteration a successful case was given, and failure cases were provided if necessary to keep the system from making the same mistakes.

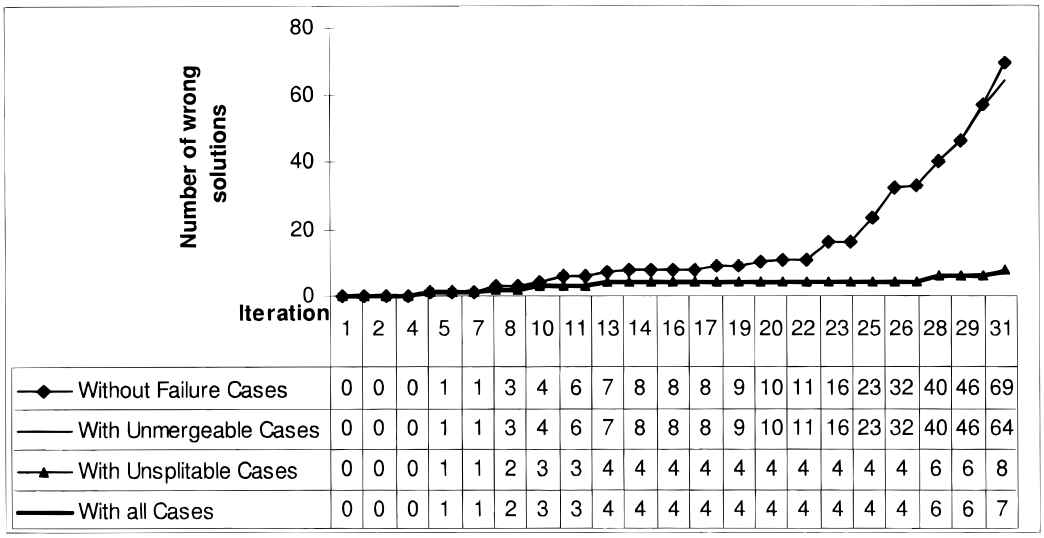
Figure 6 shows the number of solutions considered wrong in each iteration using the four different runs. These results show that the unsplitable cases drastically decrease the number of solutions considered wrong. Therefore, failure cases are necessary to filter the solutions that are presented to the user. The unmergeable cases did not have much influence on the results; this is probably due to the type of domain.



Iteration	1	2	4	5	7	8	10	11	13	14	16	17	19	20	22	23	25	26	28	29	31
Without Failure Cases	0	0	0	1	0	2	1	2	1	1	0	0	0	1	0	5	7	9	7	6	12
With Unmergeable Cases	0	0	0	1	0	2	1	2	1	1	0	0	0	1	0	5	7	9	7	6	8
With Unsplitable Cases	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	2	0	2
With all Cases	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	2	0	1

Fig. 6. Graph presenting the number of wrong solutions presented to the user.

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press



Iteration	1	2	4	5	7	8	10	11	13	14	16	17	19	20	22	23	25	26	28	29	31
Without Failure Cases	0	0	0	1	1	3	4	6	7	8	8	8	9	10	11	16	23	32	40	46	69
With Unmergeable Cases	0	0	0	1	1	3	4	6	7	8	8	8	9	10	11	16	23	32	40	46	64
With Unsplitable Cases	0	0	0	1	1	2	3	3	4	4	4	4	4	4	4	4	4	4	6	6	8
With all Cases	0	0	0	1	1	2	3	3	4	4	4	4	4	4	4	4	4	4	6	6	7

Fig. 7. Cumulative number of wrong solutions presented to the user.

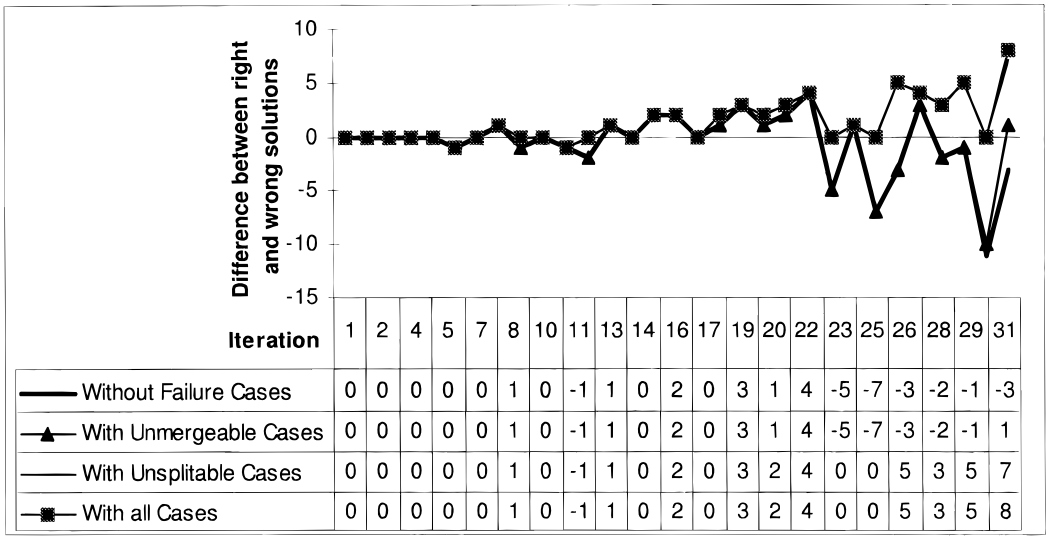
Figure 7 presents the cumulative number of wrong solutions presented to the user. As expected, the unsplitable cases greatly reduce the number of wrong solutions presented to the user, from 60 wrong solutions in the unmergeable run to 8 and 7 wrong solutions in the unsplitable and all cases run, respectively.

Figure 8 and Figure 9 present a quality measure for the presented solutions based on the difference between the number of right and wrong solutions given to the user. Figure 8 presents this measure in each iteration, while Figure 9 presents the results in accumulation. In Figure 8 we can see that, after iteration 23, the system's results decay, except in the runs that consider unsplitable cases.

6. DIMENSIONS OF MACHINE LEARNING IN DESIGN

In this section our work is classified into seven different dimensions of machine learning in design:

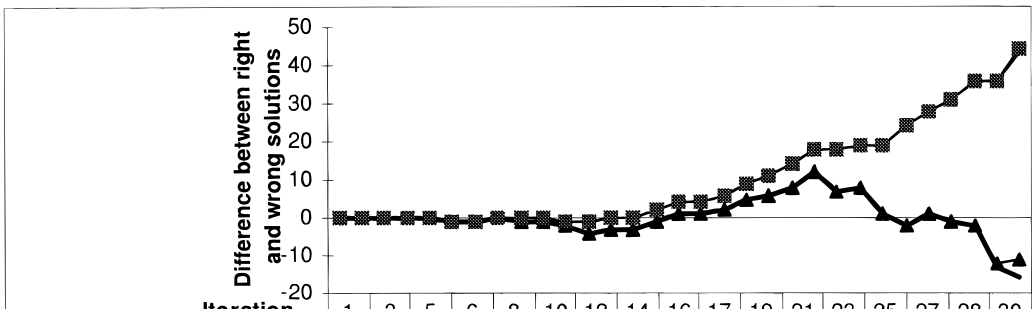
1. What can trigger learning?
2. What are the elements supporting learning?
3. What might be learned?
4. Availability of knowledge for learning.
5. Methods of learning.
6. Local versus global learning.
7. Consequences of learning.



Iteration	1	2	4	5	7	8	10	11	13	14	16	17	19	20	22	23	25	26	28	29	31
Without Failure Cases	0	0	0	0	0	1	0	-1	1	0	2	0	3	1	4	-5	-7	-3	-2	-1	-3
With Unmergeable Cases	0	0	0	0	0	1	0	-1	1	0	2	0	3	1	4	-5	-7	-3	-2	-1	1
With Unsplitable Cases	0	0	0	0	0	1	0	-1	1	0	2	0	3	2	4	0	0	5	3	5	7
With all Cases	0	0	0	0	0	1	0	-1	1	0	2	0	3	2	4	0	0	5	3	5	8

Fig. 8. Graph showing the differences between right and wrong solutions.

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press



Iteration	1	3	5	6	8	10	12	14	16	17	19	21	23	25	27	28	30
Without Failure Cases	0	0	0	-1	0	-1	-4	-3	1	1	5	8	7	1	1	-1	-13
With Unmergeable Cases	0	0	0	-1	0	-1	-4	-3	1	1	5	8	7	1	1	-1	-12
With Unsplitable Cases	0	0	0	-1	0	0	-1	0	4	4	9	14	18	19	28	31	36
With all Cases	0	0	0	-1	0	0	-1	0	4	4	9	14	18	19	28	31	36

Fig. 9. Graph showing the cumulative differences between right and wrong solutions.

There are three different learning situations in IM-RECIDE. The first one happens when a new case is stored in the case library. The second occurs when the user provides failure cases to the system in the sequence of rejection of solutions. The last one happens when the system cannot give a solution(s) to the new problem and the user is asked to give a new one.

6.1. What can trigger learning?

The first learning situation in IM-RECIDE occurs when a new case is generated by the system using operations that are capable of creating new knowledge. In these circumstances, the system extends the case library. This leads to an improvement in the system performance because time is spared in solving similar problems.

The second situation that can occur is when a new solution is presented to the user and failures are detected in it. This triggers the learning of failure cases. This type of knowledge disables making the same mistake again. In this situation, the important aspect is the teamwork developed by the system and the user. The user acts as a teacher, giving guidelines to its student (IM-RECIDE).

The final situation, which is when the user provides a solution to the problem, is similar to the previous one. In both situations, learning is triggered by failure in solving a problem. The difference resides in the failure situation. While in the previous situation the failure was in the form of an incorrect solution generated by the system, in this situation the failure occurs when the system is not able to come up with more solutions and thus must ask the user for a solution.

6.2. What are the elements supporting learning?

One element that supports learning in IM-RECIDE is the critique drawn by the user. The user's task is not only to give new problems to the system and wait for the solutions, but also to warn the system when a mistake has been performed. Another supporting element is the type of operations used for generating a new design. This constrains which new solutions can be learned by the system, because not all of the new knowledge (cases) is worth storing in the case library.

6.3. What might be learned?

Two different things are learned by IM-RECIDE. New cases can be stored in the case library, extending the range of problems that the system can solve. This knowledge is used for the generation of new designs, but the learning of failure cases is also allowed. The failure cases represent constraints to the solutions generated by the system.

6.4. Availability of knowledge for learning

There are two different sources of new knowledge. One is the user, which is the best one, but may not be available. The other is the system itself, by the generation of new cases. These new cases are created due to the interaction of the reasoning operators and the case library. Because the user is also constantly learning, the information provided by the user is not stable. One consequence of this instability of information is the possibility of having two contradictory cases in the case library. On one occasion one case was provided to the system, a case that later was proven to be wrong. Another aspect of the knowledge available for learning is whether it is full or partial. Because case-based learning deals with specific knowledge, most of the time this knowledge is partial and incomplete.

6.5. Methods of learning

The method used for learning is case-based learning and storage of new knowledge directly from the user. Both methods are very simple and provide good-quality knowledge.

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press

6.6. Local *versus* global learning

IM-RECIDE was developed to help the designer. It acts like a personal assistant, learning the designer's preferences and solutions. Local versus global learning can be analyzed from a different perspective. The knowledge learned by the system can be specific, like the cases stored in the case library, or general, like the failure cases that can represent constraints.

6.7. Consequences of learning

Bearing in mind the three learning situations that occur in IM-RECIDE, it can be said that in the first one the consequence of learning is an improvement in performance. The time used to solve similar problems is less than the time used to solve the problem for the first time.

In the second learning situation, the consequence is the diminishing of the number of wrong solutions, thus improving the quality of the solutions presented to the user. A side effect of this situation is an increase in the verification time, because more failure cases need to be matched.

The last learning situation improves the system's performance in terms of quality of solutions and the range of problems that can be solved by the system.

7. FINAL REMARKS

In our system, both failure and success trigger learning: failure when the system is not able to find a solution, and success when IM-RECIDE successfully solves a problem. The learning process is supported by the feedback after having completed the design task. Three types of cases can be learned: success, unsplitable, and unmergeable cases. The success cases represent knowledge that will be used for the generation of new designs. Unsplitable and unmergeable cases represent constraints to the new solutions created by IM-RECIDE. The system acquires these new cases through interaction with the user. The types of learned cases influence the system in solving new design problems. However, they also give the system more knowledge, allowing IM-RECIDE to produce more alternative solutions. The failure cases act as validation knowledge, verifying the solutions generated by IM-RECIDE, and improving the overall quality of the design solutions presented to the user.

One limitation of our system is that failure cases are closely related to the splitting and merging operator; in this sense, there are failure situations that are not covered by the cases we used. There are other types of failure cases that we can explore in the future. The case representation knowledge is not being used at its full potential, because we are not considering failure cases with explanations. Experiments are being done with evaluation functions to classify the creativity of new solutions.

REFERENCES

Bento, C., & Costa, E. (1994). A similarity metric for retrieval of cases imperfectly explained. In *Topics in Case-Based Reasoning—Selected Papers from the First European Workshop on Case-Based Reasoning*, Kaiserslautern (Wess, S., Althoff, K.-D., & Richter, M.M., Eds.), pp. 92–105. Springer-Verlag, Heidelberg.

Bento, C., Macedo, L., & Costa, E. (1995). Reasoning with cases imperfectly described and explained. In *Advances in Cased-Based Reasoning—Selected Papers from the Second Workshop on Case-Based Reasoning* (Weiss, S., Althoff, K.-D., & Richter, M.M., Eds.), pp. 45–59. Kaiserslautern Springer-Verlag, Heidelberg.

Gero, J. (1994). Computational models of creative design processes. In *AI in Creativity* (Dartnall, T., Ed.), 269–281. Kluwer, Dordrecht.

Gomes, P., Bento, C., Gago, P., & Costa, E. (1966). Towards a case-based model for creative processes. *Proc. 12th European Conf. Artificial Intelligence*, pp. 122–126. John Wiley & Sons, New York.

Hammond, K. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. *Proc. 5th Nat. Conf. Artificial Intelligence* (AAAI86), pp. 556–560.

Kolodner, J. (1993). *Case-Based Reasoning.* Morgan Kaufman, Los Altos, CA.

Kolodner, J., & Wills, L. (1993). Case-based creative design. *AAAI Spring Symposium on AI+Creativity*, Stanford, CA.

Maher, M., Balachandran, M., & Zhang, D. (1995). *Case-Based Reasoning in Design.* Lawrence Erlbaum Associates, Mahwah, New Jersey.

Tong, C., & Sriram, D., Eds. (1992). *Artificial Intelligence in Engineering Design*, Vol. I. Academic Press, New York.

Paulo Gomes received his B.S. and Ms.C. degrees in Informatics Engineering from the University of Coimbra, Portugal, and is a Ph.D. student at the University of Coimbra. He is currently Assistant Professor in the Informatics Engineering Department of the University of Coimbra. He is a member of the Coimbra Artificial Intelligence research group doing research in case-based design and creative systems.

Carlos Bento received a Ph.D. in Engineering Sciences, Specialty in Informatics, from the University of Coimbra. His thesis subject was reasoning based on cases imperfectly described and explained. He is currently Auxiliary Professor in the Informatics Engineering Department at the University of Coimbra. He is a member of the Coimbra Artificial Intelligence research group, where he is coordinator for the research areas of case-based reasoning and intelligent data analysis. He is also involved in the research area of creative systems.

Pedro Gago received his B.S. degree in Informatics Engineering from the University of Coimbra, Portugal, and is a Ms.C. student at the University of Coimbra. He is currently Assistant Professor in the Superior Institute of Technology and Management of Leiria, Portugal. He is a member of the Coimbra Artificial Intelligence research group doing research in case-based reasoning and knowledge discovery in data bases.

https://doi.org/10.1017/S0890060498122035 Published online by Cambridge University Press