Computational Intelligence and Case-based Creativity in Design

Will Byrne, Thorsten Schnier, and R. J. Hendley

University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

Abstract. Creativity is often described as the intelligent misuse or rearrangement of existing knowledge, and is meaningless without the context and associated expectations provided by that knowledge. We outline a model of problem-solving creativity based on the generalisation of cases and discuss how sub-symbolic computational intelligence techniques might be used to implement it. We propose a novel measurement for creativity based on the idea of creative potential or the 'degree of misuse' of existing concepts contained in a case base.

Keywords: creativity, case-based reasoning, computational intelligence, design

1 Introduction

It can be argued that design is essentially problem solving [1], [2], and that creativity is essentially remarkable problem solving. Doing this in an intelligent manner means using and adapting knowledge, and so it seems clear that an artificial problem solving system should be knowledge-driven. Knowledge is clearly an important aspect of creativity: without it people (or other agents or systems) cannot intelligently come up with new solutions or concepts, and there is no context to give creativity meaning and allow surprise.

For frequently encountered, routine design problems there is no need to be creative – the relevant knowledge is obvious and previous solutions can simply be reused, perhaps with some modification if necessary. If existing solutions do not apply directly then more creativity is needed, but a solution must still ultimately be based on the knowledge available. This hints at a possible definition for creativity based on how much manipulation of knowledge is needed to produce a solution.

While creative things are surprising they are not totally independent of more familiar things – indeed they must be related if they are to have any meaning or element of surprise. Many researchers have suggested that a creative solution is essentially a new use or combination of knowledge we already have, and that creativity differs from more routine problem-solving in degree rather than kind. Henri Poincaré argued that creative ideas "reveal unsuspected kinships between other facts well known but wrongly believed to be strangers to one another" [3] and variations on this theme of rearranging existing knowledge have been repeated by others, including Schank with his "intelligent misuse of knowledge" [4] and Koestler with his bisociation theory [5].

In other words, existing knowledge may contain concepts or solutions that might be applicable to the current problem but are obscured by context- or domain-specific details. Creativity involves the ability to recognize this useful knowledge and reuse it in different situations or contexts. The more different the contexts concerned the more creative the result may be considered to be.

This ability to recognise relevant concepts reminds us of *activation functions* or *associative hierarchies* [6], which attempt to capture how we are reminded of particular things given a problem cue. A flat activation function produces more (and more unusual) responses than a spiking one, with creative people said to have flatter activation functions, reinforcing the idea that making non-obvious associations is key in creativity.

In the next sections we will outline a high-level model of case-based creativity and discuss how computational intelligence techniques might be used to implement it.

2 Case-based Creativity

Both creative and routine problem solving are based on the application of existing knowledge to new problems. This has obvious similarities with Case-based Reasoning (CBR), a well known knowledge reuse methodology. In CBR cases can be thought of as "contextualised pieces of knowledge" [7], or concepts wrapped in context-specific details. Cases typically consist of a problem and a solution, although they can include extra information such as results and lessons learned. Solutions can be artefacts or instantiations or, more commonly, methods or processes. If they are artefacts they are usually general rather than concrete. In addition to cases CBR systems may contain other information such as more general knowledge or adaptation rules.

CBR essentially consists of the"4 rs"[8] :

- Retrieve
- Revise
- Reuse
- Retain

Most CBR research focuses on the first 2 steps, which involve problems collectively known as the indexing problem [9]: how to select the cases that will produce the best solutions to the input problem – in other words, if useful knowledge is contained in a case how to make sure it is retrieved. The relationship between retrieving and adapting cases depends on the particular implementation but can be very close – for example, cases may be selected on the basis of adaptability rather than similarity to the input problem.

CBR is based on the assumption that similar problems have similar solutions, which corresponds loosely to routine design or problem solving. However, retrieving a solution from the closest matching case has limitations from a creativity point of view: Too close and the adapted solution will merely be a non-creative variation on an existing one, too far and it may be too difficult to adapt. We need to consider other ways of retrieving a solution, as case solutions containing concepts that can form the basis of a creative solution have corresponding problems that may be (or appear to be) dissimilar to the input problem.

2.1 Misusing knowledge

We want to identify knowledge that is not obviously relevant to the input problem and misuse it in an intelligent or sensible way. It has already been suggested that a straightforward similarity measure is not necessarily the best way to choose which cases to use. Work has been done on retrieving cases on the basis of their adaptability, for example Smyth and Keane's Adaptability Guided Retrieval (AGR) [10]. However this requires some assessment of adaptability, in this case "specially-formulated adaptation knowledge" that can be quickly applied to each case. This knowledge appears to be hand-written for specific domains.

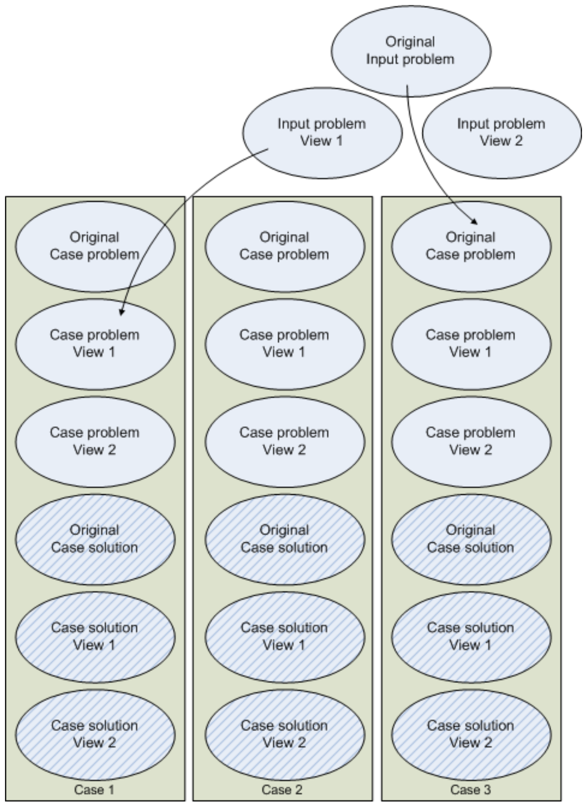
However we have an apparent contradiction: we are suggesting that concepts contained in dissimilar cases may lead to creative solutions, and that similarity measurements will not lead us to them. But this implies that the cases *are* in fact similar or analogous in some way – if they were not the retrieved concept could not lead to a useful solution. We need to uncover the hidden or *potential* similarities.

There are several reasons why cases may not initially appear to be similar to the input problem, even when described in compatible formats. They may be:

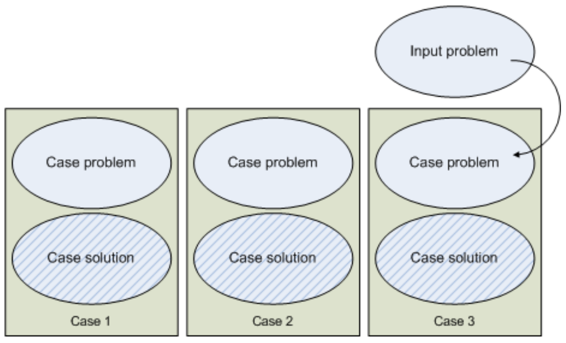
– similar but described differently (e.g. with domain- or context-specific vocabularies);
– described at different levels of abstraction;
– describable in many equivalent (or at least non-contradictory) ways, for example by function (teleology), behaviour or structure.

2.2 Identifying useful concepts

Rather than using cases directly in the retrieval stage, we suggest generating different 'views' of cases and the input problem and assessing these for similarity (Figure 1(a)). By using different views of the problem we can make connections that may not be apparent in the original views, and retrieve concepts that can be incorporated into new solutions. We suggest that the results will be more creative than if we had retrieved a solution through a more obvious similarity (Figure 1(b)). To support this claim we would have the transfer of a concept between two different (or not obviously similar) cases and contexts, and the fact that the solution is not arrived at through simply taking an existing solution and modifying it. The less obviously similar the input problem and 'donor' case problem the more creative the result, and the more work or processing needed to find the common problem. We suggest that some measure of this work represents



(a) Identifying commonalities through similarity of different views



(b) Identifying commonalities through case similarity

Fig. 1. Retrieving useful cases through similarity

creative potential and could provide a means of evaluating for creativity. This fits well with accepted views of creativity as the misuse of knowledge.

There are two broad approaches to identifying commonalities:

- look for connections in case problems only, on the assumption that corresponding solutions will be useful;
- look for connections in both case problems and case solutions.

In the first case we would be looking for a problem common to both the input problem and a case problem but not necessarily explicitly described in either. The corresponding case solution should therefore contain some concept or knowledge useful for the input problem. Rather than having a relatively concrete solution to adapt as in CBR we may have a more abstract solution we need to concretise or specify.

In the second case we may be able to bypass using the case problem as an index and look for connections with solutions directly. For instance, one aspect of the input problem may include the concept of rotation, and we may be able to identify some case solutions that incorporate rotation.

2.3 Case and problem views

Problems and solutions may be viewed in several different ways. We may use different levels of abstraction, different vocabularies, or describe different perspectives such as function, behaviour or structure as in Gero's FBS model [11]. For example, our problem might involve designing a bridge to carry cars across a gorge. By mentioning bridges we have provided a context or bias focused on a span across a gap and are already describing the problem in terms of a known structure. Our case base might contain some cases involving bridges or spans, allowing us to retrieve an existing solution concept and perhaps modify it for our particular application. This is not likely to lead to a particularly creative solution as what we have is still a bridge, and a modified version of an existing bridge at that. However, looking at the desired *function* of the bridge gives a different view: enabling cars to move from one side of an obstacle to another. A behavioural or structural view of a solution such as, say, a record player might enable a connection between movement and a rotating turntable, which may lead to a more creative way of crossing a gorge.

One approach to producing different views of cases and problems is through context-driven or "lazy" generalisation. This may lead to more abstract views with less context- or domain-specific details to obscure potential similarities.

3 Computational Intelligence and Case-based Creativity

To recap, our thesis is that creativity in intelligent design/problem-solving relies on retrieving concepts from cases that might not be identified by straightforward similarity measures, and the less obvious the similarity the more creative the solution will be. We are primarily interested in producing new concepts (the 'creative leap') rather than detailed instantiations.

Given a set of cases and an input problem our goal is to retrieve a concept contained in some exisiting case solution and use it to develop a new one. We are particularly interested in cases where the case problem is apparently dissimilar to the input problem. Our suggestion is that similarities may become apparent at higher levels of abstraction than is given in the original case and problem descriptions. Computational Intelligence (CI) techniques could be used to produce generalised concepts (that is, more abstract concepts that are incorporated in one or more less abstract cases). This process of generalisation could be repeated until a match is found and would apply to both case problems and solutions. The corresponding abstract solution concept would then form the basis of a new solution, which when instantiated would not be merely an adaptation of an existing solution at the same level of detail (Figure 2).

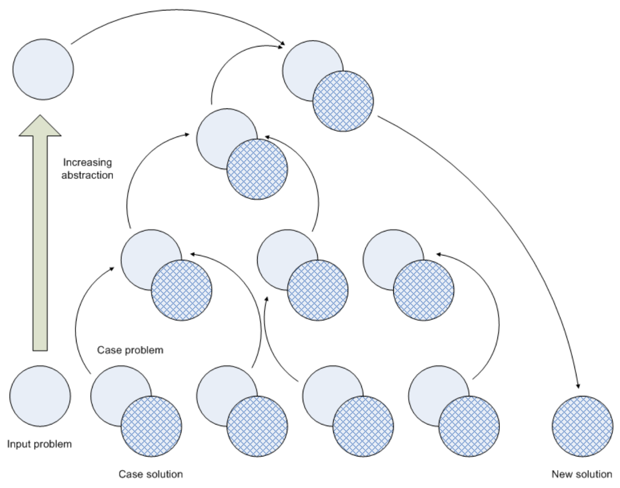


Fig. 2. Case-based creativity through generalisation

We suggest that CI techniques may support the operations and flexible generalisation needed to implement a Case-based Creativity (CBC) system as outlined above. CI techniques are essentially non-symbolic and include evolutionary computation and connectionist and fuzzy systems. These techniques have some useful properties and abilities such as tolerance to noise, clustering, generalisation, pattern matching, and (unsupervised) learning, and do not necessarily depend on their states mapping to valid examples in the input space, or the construction of a model of the space. The advantage of this approach is that we may be able to perform semantically meaningful generalisation without the overheads associated with reasoning over structured, symbolic data.

3.1 Graph representation

The model outlined above is very much symbolic in nature. This is unavoidable for working with problems which are in rich enough domains for creativity to have meaning.

Graphs provide a convenient way to represent structured, symbolic knowledge. Usually knowledge graphs are directed acyclic graphs such as semantic networks or concept nets. They are closely related to ontologies, which can be represented as graphs and provide both a vocabulary and a means of specifying relationships between entities (nodes in the graph). Graphs can also support induction and the definition of new relationships and concepts, useful for a system such as the one discussed. However doing this is computationally very expensive and we are not necessarily interested in the 'correct' formation of new concepts through induction; rather, we are looking for a mechanism to provide pointers to existing solutions whose relevance would not otherwise have been apparent. For this we need only know that two generalised concepts (i.e. higher level concepts produced through the abstraction of lower level ones) would be similar – whether these generalisations can be mapped to valid graphs with respect to some ontology is secondary.

In a case-based system individual cases might consist of a pair of graphs representing a solution and a problem.

3.2 Symbolic data and Computational Intelligence

An obvious issue is that CI techniques are sub-symbolic – that is, they work with real-valued vectors representing things in number space rather than symbol space. As mentioned above we need to use symbolic representations to work with meaningful design problems and creativity, and there is a 'semantic gap' between sub-symbolic and symbolic representations. However, some work has been done on using symbolic knowledge with CI [12] [13], typically involving recurrent neural networks or the recursive application of graphs to SOMs. Other approaches include some kind of preprocessing to reduce dimensionality, e.g. principal component analysis (PCA) or spectral analysis to identify the eigenvalues of a graph which can then be applied to a connectionist system.

If using CI techniques to generalise graphs corresponds to generalising the concepts that they represent, then the results will be semantically meaningful. Both Sparse Distributed Memory models (SDMs) [14] and Self-organising Maps (SOMs) [15] can generalise and retrieve items matching an input. In addition SOMs impose a topology on an input space, which may or may not be useful as we are primarily interested in which cases are similar to a given problem, not with each other. However, a SOM generalises over the input space on which it is trained. This means that weight vectors come to represent examples from that space that it has not necessarily seen but are representative of the topology of the input space. With reference to our high level model we might call this 'horizontal generalisation'. What we are interested in is 'vertical generalisation' to higher levels of abstraction, and how this might be achieved with approaches such as SOMs appears to be an open research question.

3.3 Concretising solutions

Rather than adapting an existing solution we will likely need to concretise an abstract solution. There is an argument to be made that identifying this solution constitutes the 'creative leap' and that the job of producing a more detailed solution or design is secondary and likely to be routine. Again, we only have our existing knowledge with which to work in order to turn an abstract concept into a more detailed solution. We may have the generalised concept we have chosen, the context-specific details from the input problem, specific cases incorporating the generalised solution, the generalised input problem, generalised case problems incorporating the concept, and maybe other cases selected 'non-creatively' (e.g. with a similarity measurement) if available.

4 Other implementation issues

4.1 Building a case base

The aim is to use a publicly available dataset with which to build a case base. This means developing a graph for a problem and another representation for a solution. There are several ontologies available in RDF format that provide the vocabulary and relationships necessary to describe concepts in the relevant domain, and these can easily be used to generate graphs.

There is likely to be a conflict between a rich enough, cross-domain dataset to allow for creativity and practical considerations (e.g. vocabulary size). A domain-specific dataset has the advantage of consistency and a relatively tightly controlled vocabulary, but everything is already in the same context so there is perhaps less scope for creativity.

However there still remains the issue of describing a problem. If possible we want to avoid doing this ourselves to keep the introduction of our own bias to a minimum, but even so it is likely that any publicly available dataset will need a significant amount of preprocessing.

One possibility is to use patent data. Patents are attractive because they cover all domains and are often described in terms of problems and solutions using very specific language. Another advantage is the possibility of comparing the output of the system with actual solutions.

Whichever domain/dataset is chosen it will ideally be human-meaningful; i.e. the outputs are such that humans can assess their creativity.

4.2 Scope

The scope of a system will depend on its intended role in the design process. We have argued that the essence of the creative leap is in the association rather than in the concretisation of the resulting concept. If we envisage the system serving to prompt human designers then highlighting the relevant solutions might be sufficient. However if we wish to automate the evaluation of the results we will need to construct a new case that can be compared to existing ones which, depending on the generality of the retrieved solution, might involve some further generalisation and concretisation.

4.3 Evaluation

We will need to evaluate how well the mapping between concepts and their sub-symbolic representations is preserved during processing in order to evaluate any results for creativity. If we are confident that semantic generalisation is occurring then we may explore the concept of creative potential to evaluate the creativity of results.

5 Summary

We are suggesting that creative potential increases with abstraction, and that creativity is in fact meaningless at the lowest, sub-symbolic or instance level. This suggests that there may be a novel and interesting assessment of creativity based on the amount of processing done on the case base to produce a solution, or at which level of abstraction an isomorphism was found between the new problem and the abstracted case problem. This might be used in combination with graph distance metrics. However we will be interested in the combination of both the problem (context) and solution aspects of a case, which may involve more than one graph.

It is also important that any solutions do actually solve the problem at hand. The need to evaluate performance may limit the choice of domain, and implies that we may need to generate solutions at the sub-symbolic level in order to be able to model or simulate them for evaluation.

References

1. Welch, M.: Analyzing the tacit strategies of novice designers. Research in Science and Technological Education **17**(1) (1999) 19–34
2. Johnsey, R.: The design process- does it exist? International Journal of Technology and Design Education **5**(3) (1995) 199–217
3. Poincaré, H.: The Foundations of Science. Science Press, Lancaster PA (1913)
4. Schank, R.C., Cleary, C.: Making machines creative. In Smith, S., Ward, T.B., Finke, R.A., eds.: The Creative Cognition Approach. MIT Press (1995) 229–247
5. Koestler, A.: The Act of Creation. Picador, London (1964)
6. Gabora, L.: Cognitive mechanisms underlying the creative process. In: C&C '02: Proceedings of the 4th conference on Creativity & cognition, ACM (2002) 126–133
7. Wills, L., Kolodner, J.: Towards more creative case-based design systems. In: AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1), American (1994) 50–55
8. Cunningham, P.: CBR: Strengths and weaknesses. In: IEA/AIE (Vol. 2). (1998) 517–524
9. Kolodner, J.: What is case-based reasoning? In Hofstadter, D., ed.: Case-based Reasoning. Morgan Kauffmann, San Mateo, CA (1993)

10. Smyth, B., Keane, M.T.: Experiments on adaptation-guided retrieval in case-based design. In: ICCBR. (1995) 313–324
11. Gero, J.S., Kannengiesser, U.: The situated function-behaviour-structure framework. Design Studies **25**(4) (2004) 373–391
12. Frasconi, P., Gori, M., Sperduti, A.: A general framework for adaptive processing of data structures. IEEE-NN **9**(5) (1998) 768
13. Hagenbuchner, M., Sperduti, A., Tsoi, A.C.: A self-organizing map for adaptive processing of structured data. Neural Networks, IEEE Transactions on **14**(3) (May 2003) 491–505
14. Kanerva, P.: Sparse Distributed Memory. Sparse Distributed Memory. MIT (1988)
15. Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. IEEE-NN **11**(3) (2000) 586