SITUATED CREATIVITY-INSPIRED PROBLEM SOLVING

by

WILL BYRNE

A thesis submitted to the University of Birmingham for the degree of DOCTOR OF PHILOSOPHY

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

December 2014

Abstract

Creativity is a useful attribute for people to have. It allows them to solve unfamiliar problems, introduce novelty to established domains, and to understand and assimilate new information and situations – all things we would like computers to be able to do too. However, these creative attributes do not exist in isolation: they occur in a context in which people tend to solve problems routinely where possible rather than consider non-standard ideas. These more mundane attributes might also be useful for problem-solving computers, for the same reasons they are useful for us. However, they are often ignored in attempts to implement systems capable of producing remarkable outputs.

We explore how the study of both human and computational creativity can inform an approach to help computers to display useful, *complete* problem-solving behaviour similar to our own: that is, robust, flexible and, where possible and appropriate, surprising. We describe a knowledge-based model that incorporates a genetic algorithm with some characteristics of our own approach to knowledge reuse. The model is driven by direct interactions with problem scenarios. Descriptions of the role or appearance of key themes and concepts in literature in functioning problem-solving systems is lacking; we suggest that they appear as artefacts of the operation of our model. We demonstrate that it is capable of solving routine problems flexibly and effectively. We also demonstrate that it can solve problems that would be effectively impossible for a genetic algorithm operating without the benefit of knowledge-driven biasing. Artefacts of the behaviour of the model could, in certain scenarios, lead to the appearance of non-routine or surprising solutions.

This thesis is dedicated to my mother, who has always encouraged me in everything I've done.

Acknowledgements

Many people have provided guidance, support and encouragement, both academic and otherwise. Firstly, I would like to thank my wife, Teresa, and my family. At the University of Birmingham I would like to thank my supervisors, Robert Hendley and Dr Thorsten Schnier, and my thesis group members, Professor Russell Beale and Dr Peter Hancox. I am also grateful to Professor Xin Yao and the Centre of Excellence for Research in Computational Intelligence and Applications for offering me the opportunity and funding to study for a PhD. Lastly, thanks to every colleague and friend who has given their advice and support over the years.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

People constantly encounter scenarios in which they need to bring about desired outcomes, in their routine, everyday activities as well as in fields like engineering, product design, and art. Often these scenarios and outcomes are familiar or routine, but occasionally some more unusual examples appear, and when that happens we sometimes attribute it to creativity. Creativity is what allows us to interpret familiar scenarios in new ways, produce more novel and surprising designs, find solutions to new problems or reveal new directions to explore (Gero and Kazakov, 1996; Boden, 1990). Creative designs and solutions to problems can be more efficient, more distinctive, cheaper, or simply more elegant than existing ones, or they might break entirely new ground. However, not all design and problem-solving is creative, nor does it necessarily need to be – sometimes a familiar, tried and tested design is perfectly acceptable, and perhaps quicker, cheaper and less risky to produce than a ground-breaking new concept (Cropley, 2016; Rosenman, 1997). But the ability to (at least occasionally) do things differently makes us adaptable and so able to operate more resiliently (Runco, 2014), and also produce things our peers both appreciate and are surprised by.

Creativity, then, is a very useful human attribute and, like many human attributes, it would be desirable for computers to have it too, if we want them to be able to do some of the same things we do. For example, we might want to delegate design and problem-solving tasks to them, or we might want them to be better able to fend for themselves as autonomous agents, by coping more flexibly and robustly with unfamiliar scenarios. We might also want them to surprise us by producing artefacts like music, stories or paintings that we find creative, and it is to this end that much work on creativity and computers is concerned. Computational creativity is also closely related to the automated discovery of new scientific and mathematical concepts (Shen, 1990; Lenat, 1979; Colton, 2001). These require the management of inconsistencies and conflicts as well as an ability to recognise interesting artefacts (Colton and Steel, 1999). As a fundamental aspect of human intelligence, creativity is closely linked with the "association of ideas, reminding, perception, analogical thinking, searching a structured problem-space, and reflective self-criticism" we rely on when performing these activities ourselves (Boden, 1998:p 347). The study of creativity is therefore also of interest in the wider study of artificial intelligence, and projects such as Cyc (Lenat et al., 1985) which aims to provide a comprehensive knowledge base to allow computers to perform the necessary rich analogical reasoning.

Traditionally, however, creativity has been thought of as the very opposite of the process and algorithm with which we associate computers – a flash of inspiration or intuition perhaps associated more with the arts than the sciences and found only in people (although more recently problem-solving creativity has been attributed to some animals, notably New Caledonian crows (Weir and Kacelnik, 2006)). This does not bode well for a relationship between creativity and computers, but there is good reason to suppose that creativity is not as intangible as it might first appear.

Ada Lovelace famously said that computers (or at least those represented by Babbage's analytical engine, her only reference at the time) cannot ever produce or originate anything, only do what we program them to do, and Bringsjord et al. (2003) have extended this argument to say that if the programmer can account for a result then it is not creative. However, to take this argument too far is to be in danger of holding computers to a higher standard than we would apply to humans, especially if we interpret it as saying that even computers displaying strong AI[1] could not be creative, even if they were the equal of human intelligence in all other respects.

Meanwhile Thaler (2012) argues that creativity can essentially be found in a neural network operating at a critical level of noise, allowing it to generate outputs that are not simply retrieved memories, but rather 'false' memories of things that *could* be found in the external environment from which the network receives its inputs but that have not actually been encountered. These outputs should represent valid instances in the conceptual space in which the network has been trained, but beyond this critical level of perturbation the system would be expected to produce mainly nonsense. Thaler goes so far as to claim that such a system would be capable of thinking and feeling, negating arguments against computational creativity on grounds of consciousness. Whether or not this is the case, there is a suggestion here that such 'creativity machines' – connectionist systems generating new patterns in response to external stimuli – might be a rebuttal to Lovelace's objections.

Despite the seemingly amorphous nature of creativity, there is evidence that ideas do not appear from nowhere but rather are a product of the interaction of knowledge, experience and context. Coleridge's

[1]Strong artificial intelligence (AI) is typically used to refer to computers that have comparable cognitive abilities to humans, but do not necessarily work the same way. Searle (1990) says that a computer or program displaying strong AI would actually be a mind itself rather than simply a model of one, in contrast to weak AIs which would have the same relationship to the mind as weather models have to the actual weather.

"hooks and eyes" (Boden, 1990:p 29) of memory are echoed in Poincaré's observation that creative ideas "reveal to us unsuspected kinships between other facts, long known, but wrongly believed to be strangers to one another" (Poincaré, 1913:p 386).

1.1 A Note on Design and Problem-solving

Much work on creativity involves design and problem-solving. Design can generally be thought of as problem solving (Welch, 1999; Johnsey, 1995). Chandrasekaran (1990) says it is a search problem with multiple constraints in a large space, while Simon (1995:255-260) described design as the process of generating ideas to accomplish certain goals, as well as a plan for their implementation. How clearly and rigidly these constraints and goals are specified depends on the domain – for engineering design we would expect them to be very clear, for sculpture perhaps less so. Nonetheless, there is evidence that the principles at work are very similar – for example, Weisberg shows several parallels between Edison's development of the kinetoscope and Picasso's work on Guernica (Weisberg, 1988), showing how they were both working towards a pre-defined goal. We shall assume from this point that design is a special case of problem solving, and consider work on creativity in both domains.

1.2 Research Objectives

In this thesis we will ask how the study of computational creativity can inspire an approach that can help computers to better solve problems in the same way we do – that is, robustly, flexibly, and sometimes surprisingly. The context of the work is a knowledge-based system that must reliably solve frequent routine problems as well as rarer non-routine ones, in domains that are perhaps less often associated with creativity – functional, or engineering type problems with objective goals, and perhaps with less opportunities for acceptable, unusual solutions than some other domains. Any consideration of the creative qualities of the results are secondary to this problem-solving capability, although we will go on to discuss the relationship between them.

1.3 Contribution

The contribution of the work is a knowledge-based framework for robust problem-solving that combines aspects of computational creativity, case-based reasoning, and evolutionary computation. The novel aspect of the framework is that its approach is based around *limiting* the use of knowledge rather than expanding it, inspired by some characteristics of the problem-solving behaviour displayed by people – who are, after all, our primary reference for creative behaviour. Any capacity for non-routine problem solving is an artefact of this approach rather than the focus. The work is based on the following observations:

People are situated Their behaviour, both problem-solving and routine, is strongly influenced by their close coupling with their environment (Gero and Kannengiesser, 2004; Withagen et al., 2012; Wilson, 2002; Stojanov and Indurkhya, 2013).

People are knowledge driven Most things people do, creative or otherwise, are based on the intelligent application of knowledge they have acquired (Aamodt and Plaza, 1994; Hofstadter, 2001).

People are often reluctant to consider non-standard scenarios People tend to bring only a small portion of the knowledge that they have available to bear on a given problem, behaving as cognitive misers (Fiske and Taylor, 2013) and biased towards standard ways of dealing with situations they encounter unless they have an incentive for doing otherwise (Bilalić et al., 2008; Hofstadter et al., 1994).

People must frequently perform routine tasks Even creative individuals must frequently solve routine problems quickly and efficiently, with demonstrations of creativity relatively infrequent. The tendency noted above to resist non-standard ways of approaching tasks is in keeping with the context in which people use their knowledge to solve problems, when there is often no benefit to devoting time or resources to considering new solutions where a routine solution is available.

People operate in largely predictable domains Artefacts that people manipulate to solve both routine and non-routine problems generally behave in a consistent, predictable way. That is, the outcome of exploiting an affordance is predictable based on previous experience (Hawkins and Blakeslee, 2004:202).

People sometimes do creative things Despite the above, people do sometimes do surprising, non-routine things.

The prominent role of routine, unimaginative problem-solving means the framework is consistent with the criteria proposed by Wiggins (2012) for an evolutionary imperative – that is, a plausible reason for why behaving in such a way might have been a useful evolutionary trait. The framework is also consistent with another prominent theme in creativity research, which is "the 'intelligent misuse' of the knowledge structures underlying routine cognition" suggested by (Schank and Cleary, 1995:p 230) and echoed in Koestler's bisociation theory (Koestler, 1964). Both of these fall under the broad umbrella of *combinatorial creativity* which we will explore in more detail in Chapter 3.

We describe some experiments and case studies that demonstrate that this approach can solve routine problems robustly, as well as some non-routine problems that would be intractable using other knowledge-based approaches.

Some of the work discussed in this thesis was previously published in Byrne et al. (2008).

1.4 Thesis Structure

Chapters 2 and 3 together comprise a literature review covering, respectively, philosophical and computational approaches to creativity and problem-solving. In Chapter 4 we will introduce a model of a strategy for knowledge reuse supporting both routine and non-routine operation. Chapter 5 will describe some experiments with the model, in two separate domains: an image-based problem-solving domain, and a simulated, physics-engine based agent. Both domains can be used as their own models, meaning that programs or agents operating in them can perform actions based directly on their interaction with the domain, rather than constructing a model of the domain to use to decide on actions. In Chapter 6 we will draw some conclusions and look at future work and research directions.

Chapter 2

THE NATURE OF CREATIVITY

In this chapter we will discuss work on what it is to be creative, frameworks in which the difference between creative and routine can be discussed, and how transitions between them can occur. The focus is on the process by which creative agents produce artefacts rather than on the qualities of those artefacts.

2.1 Conceptual Spaces

Boden (1990) defines creativity as when an idea is had that could not have been had before, and introduces the idea of *conceptual spaces* to both illustrate what it means for a new idea to appear and how this can happen. Broadly speaking, a conceptual space represents a space of possibilities consistent with a known concept – for example, a musical genre, or a type of vehicle. Each point in the space represents a particular instance of this concept, some of which will have actually been instantiated before and many more which have not. Boden argues that while it is relatively easy to find 'unused' points in this space and come up with new examples of the concept the results would be unique, but not necessarily creative or even surprising. This implies that a truly creative idea must come from *outside* the set of possible ideas bounded by this space, and from a *different* space defined by a different set of parameters or constraints. These parameters and constraints could, in turn, be altered or replaced to give another space, containing new possibilities. The limits defined by these spaces are actually very important in creativity; 'thinking outside the box' needs a box to start with.

Gero and Kannengiesser (2012) observe that if a new interpretation of something leads to a new representation of that thing, then a different set of affordances might become apparent than was apparent in a different representation. If we view opening up new sets of these affordances as effectively defining new conceptual spaces (since those affordances constrain the possible outcomes) then we have interpretation-driven association as an enabler of transformational creativity. We should note, though, that Gero was concerned with design affordances, something which it is within the power of the designer to alter, unlike the affordances of physical objects which users can recognise (or not) but not alter. Affordances were defined by Gibson (1979) as possibilities for actions that an agent perceives it can take based on its interactions with its environment. The extent to which the behaviour of an agent is prompted, dictated or invited (Withagen et al., 2012) by those affordances is debated. Since limiting those affordances constrains the possible outcomes, we can view opening up new sets of these affordances as effectively defining new conceptual spaces ("action spaces" in (Gärdenfors, 2007:p 260)).

Boden incorporates conceptual spaces into two of three widely accepted types of creativity: *exploratory* and *transformational*, which are concerned respectively with identifying new points in a space and transforming the space itself. The third type is *combinatorial creativity*, which covers unusual combinations of existing knowledge.

2.2 Creativity and Novelty

It is easy to form something novel that has never been formed before, but it is not necessarily creative to do so. An often-used illustration (Boden, 1990) is language. It is easy to come up with a valid (but probably nonsensical) sentence in English and be reasonably sure that nobody has composed it before. The sentence would be novel, but it would not necessarily be creative because it is a product of a well-defined generative system, in this case the grammatical rules of English.[1] Boden again: "A merely novel idea is one which can be described and/or produced by the same set of generative rules as are other, familiar ideas. A genuinely original, or creative, idea is one which cannot" (Boden, 1990:p 51). So this novel sentence could theoretically have been constructed at any time, it just so happens that it was not. A computer could be programmed to produce countless novel sentences using the rules of English but probably few people would regard it as being creative; it would be working inside a well-known conceptual space. Similarly, computers have been used to produce music, stories and art – all things that can be represented by a set of rules that can be implemented on a computer to produce outputs that are valid, recognisable examples of their type (Schnier, 1999; Ball, 2012; y Pérez and Sharples, 2004). This view of novelty as some measurement of typicality is incorporated in some approaches to evaluating creativity that we will discuss in 2.4.

Creative artefacts or outputs must also be useful or appropriate, in a way that novel things do not necessarily have to be (Beghetto and Kaufman, 2007; Ritchie, 2007). Just what this means depends on the context – as Mednick (1962:p 221) points out,"7,363,474 is quite an original answer to the problem 'How much is 12 + 12?'" but it is not very useful. In this case it is not valid either, but validity can vary

[1]Of course this is not to say that language cannot be *used* creatively - poetry is proof to the contrary, but the creativity here is in the concepts described and their expression through language, rather than a generative grammar.

with context or domain – if all we wanted was a novel sentence, then "12+12 is 7,363,474" would be fine. So we can note that a creative answer or solution needs a creative problem, or a problem with scope for a creative solution – which 12+12 is not, in the mathematical domain at least.

al Rifaie and Bishop (2013) describe an exploration of a pre-defined space using a hybrid particle swarm optimisation (PSO) and stochastic diffusion search (SDS) approach. The domain is line drawings, with the algorithm working from a set of points describing an existing sketch to 'interpret' it in new, novel ways. The idea of the SDS is to allow more exploration of the space than would happen with PSO alone, which would otherwise tend to produce a relatively faithful reproduction of the original sketch. The 'degree of freedom' afforded to the SDS exploration can be varied, but if there is too much freedom the algorithm deviates too far from the original sketch and produces an unrecognisable interpretation – there is too much novelty, and the output has no reference point. In other words, if the system is more, rather than less, constrained the outputs arguably have more value.

2.3 Towards a Description of the Creative Process

The 'genius' view of creativity is that it is inexplicable, irreproducible and strikes a select few like a bolt from the blue. Research into creativity starts with the assumption that this is not the case, and there is a creative process to be described and perhaps quantified. Creativity is, instead, often seen as an extension or special case of normal cognitive processes, which are generally considered to be a legitimate objective for computational study. Weisberg (1988) in particular has supported this view, arguing that the same processes are at work in much more mundane situations, and backs this up with some discussion of some famous examples including Picasso, Edison and Darwin. He acknowledges that this approach contradicts the accounts given by Coleridge, Mozart and Kekulé to describe their own inspiration and creative processes, but presents evidence to show that these accounts are not necessarily reliable. Sternberg (1988) and Minsky are also of the view that there is no substantial difference between ordinary thought and creative thought, with Minsky claiming that we are "unduly intimidated by admiration of our Beethovens and Einsteins" (Minsky, 1982:p 5)

2.3.1 Insight and Intuition

Wallas (1926) identified 5 stages in a rather high-level view of the process: *Preparation*, *incubation*, *intimation*, *illumination*, and *verification.* (He later relegated intimation to a sub-stage, making a tidier 4 stages.) This is a high-level view and does not shed much light on the underlying processes, although this level of detail is still used by researchers interested in things like encouraging creativity in business environments or brainstorming sessions. The language is sometimes almost circular – for instance, ‘illumination’ is basically the ‘Eureka’ moment, so saying that part of the creative process is having unexplained creative insights is not particularly helpful. Nevertheless, this is a step towards creativity as an identifiable, and therefore potentially reproducible and manageable, process or ability. Work at this higher level also considers the role of concepts such as motivation and reward. Motivation in particular is undoubtedly important – many famous examples of creativity involve people immersed in their work, although motivation is nothing without an opportunity to exploit it.

Hélie and Sun (2010) build on Wallas’s work concerning incubation and insight with the EII (Explicit-Implicit Interaction) theory, which is based on the principle that both implicit and explicit processes are involved in most tasks, knowledge is represented and processed redundantly and simultaneously in both, and conclusions from each are integrated. Explicit processes work on knowledge which is “easier to access and verbalize, said to be often symbolic, crisper, and more flexible”, while implicit knowledge is “harder to verbalize, often subsymbolic, and often more specific, more vague, and noisier” (Hélie and Sun, 2010:p 997). Implicit knowledge is associative, and in the CLARION model which instantiates EII this type of knowledge is distributed across nodes in a neural network forming the bottom layer of the model. Explicit knowledge is represented in individual nodes in the top layer of the model; edges between these nodes represent explicit rules about their relationships. CLARION can be used to simulate and explain human data in standard psychology paradigms such as lexical decisions, free recall, and problem solving. The ‘insight problem’ tested was to provide an explanation for a short paragraph:

> A dealer of antique coins got an offer to buy a beautiful bronze coin. The coin had an emperor’s head on one side and the date 544 B.C. stamped on the other. The dealer examined the coin, but instead of buying it, he called the police. Why? (Hélie and Sun, 2010:p 65)

Features of the problem (coin date, dealer’s decision etc.) were represented by two nodes in the top layer, representing the options (good/bad) for each. Each feature was also represented in the bottom layer, with more or fewer nodes assigned according to the assumed importance of the feature. The bottom layer is trained with a set of 8 stimuli, one for each ‘abstract explanation’ represented in the top layer. New stimuli representing previously unseen variations on these explanations, chosen to be inconsistent with the rules for good solutions encoded in the top layer, are presented to both layers. Resulting activation vectors represent hypothesis, with confidences based on their statistical distribution. A solution is judged to have been found if this confidence is above a pre-set threshold; otherwise the hypothesis is fed back as a new stimulus. CLARION is shown to perform similarly to humans under certain conditions using a process argued to be analogous to abductive reasoning (Hélie and Sun, 2010). The creative ability of CLARION is argued to be derived from implicit data processing.

Sloman (1996) suggests that the rule-based processing of explicit knowledge can suppress and overrule associative processing, but that the latter "always has its opinion heard and, because of its speed and efficiency, often precedes and thus neutralizes the rule-based response" (Sloman, 1996:p 15).

Duch claims that intuition is closely linked to the ability to evaluate similarities and the number of patterns stored in long term memory, noting that these are characteristics of connectionist models like neural networks. (*Insight* is treated as qualitatively different from intuition; it has more context, often that of having been stymied by a particular problem.) Duch also claims that "Solutions of complex problems . . . combine systematic search with intuitive recognition based on partial observations" (Duch, 2007:p 4). In this view, creativity is a product of rich associative networks, combined with some filtering of resultant concepts. As such, it is a product of ordinary cognitive processes and so should be "amenable to computational modelling" (Duch, 2007:p 10). He proposes *knowledge atoms* as discrete items of internalised, qualitative knowledge that can support rapid, associative assessments of problems or scenarios without the need for more complex evaluations.

2.4 Evaluating Creativity

Another perspective on the difference between creativity and novelty is how easily we can describe artefacts or the outputs of a would-be creative agent.

Maher (2010) suggests that creative artefacts can be evaluated according to three criteria – *novelty*, *value*, and *unexpectedness* – regardless of how they were created. Maher's measure of novelty is some distance metric from others in the same class. Value is linked to the performance of the artefact, while unexpectedness is to do with how the artefact differs from the next expected artefact, where we have some sort of prediction based on a recent series of artefacts produced previously. Ritchie (2007) argues that to evaluate the creativity of a system we first need to decide what factors we are going to observe and what their relationship to any potential creativity could be. He also argues that our reference for creative behaviour must be people, as to do otherwise would be rather circular. Rather more contentious is his suggestion that evaluation of computational creativity should *not* include any consideration of the algorithms or processes that drive it, on the basis that we have no real knowledge of how people do creative things and yet still manage to decide whether or not they are being creative. Ritchie argues that if we are to try and assess creativity empirically, then we can only do this the same way we assess human creativity.

Ritchie considers systems or programs operating to produce artefacts in domains already defined and operated in by humans, who are generally able to consistently recognise and rate new artefacts appearing in those domains. He focusses on three criteria for creativity: *novelty*, *quality* and *typicality*. Typicality is some measurement of the degree to which an artefact is a member of its class – e.g. a joke. The existence of the artefact class also implies some expectations are set, which can perhaps later lead to surprise if they are not met. Quality is a measure of how *good* the artefact is – for example, how funny the joke is. Ritchie gets around the difficulties in assessing these criteria by making it clear that quality and typicality can be thought of as primitives supplied by human evaluators. By contrast, novelty is a construct of two factors: *untypicality*, measured as per typicality, and *innovation*, which is any observed capacity for a program to produce outputs that are "different" to those upon which it is based. Implicit in this is that a computer is creative if people think it is, at least to some extent.

Ritchie also introduces the idea of an *inspiring set*, which is a subset of the *basic items* that the system could produce. For example, the set for a program for generating simple jokes could be the set of "finite sequences of words and punctuation symbols" (Ritchie, 2007:p 74). The inspiring set is a knowledge base that drives the computational process, and also plays a role in its evaluation. Ritchie does not include any discussion of whether the inspiring set could come from a *different* domain or artefact class (as it might in the misuse of knowledge) although he does allow for empty inspiring sets. In Morris et al. (2012) there is some discussion of different inspiring sets and how they might support a kind of cross-domain transfer, but the work is restricted to different inspiring sets in the same broad domain of cooking.

Ritchie provides a set of criteria to assess the outputs of a system. These criteria are functions which operate on the outputs to produce scores for properties of the outputs – for example, the ratio of untypical but high-valued outputs to typical ones. These functions use a set of threshold parameters, including *threshold to achieve high typicality*, *limit of untypicality*, and *threshold to achieve good quality.* These criteria can be used to produce a table of scores to rate a system for creativity. There were 14 criteria, and so 14 scores, in Ritchie's original paper (Ritchie, 2001). This approach allows users of the framework to choose criteria and alter parameters to effectively come up with their own definitions of creativity.

Colton takes a different view from Ritchie, claiming that the way an artefact-generating system works must be taken into account when assessing its creativity. This follows from his observation that "when consumers of paintings assess them, they do not strictly separate the process and the artefact" (Colton, 2008:p 1). Sometimes they *do* make this separation, "projecting" creativity onto the painter according to their assessment of the work, while on other occasions the assessment might be all about the process, with little value given to the resulting artefacts. Colton, anecdotally, describes a scenario in which more-or-less identical works could be produced by different methods, with knowledge of these methods potentially heavily influencing a consumer's perception of their creativity. He introduces the *creative tripod* to describe three qualities a creative computational system should have, and the relationships between them. These qualities are *imagination*, *skill*, and *appreciation.* Each leg can in turn have 3 sections, one each for the system itself, its programmer, and a consumer of the artefacts it produces. Appreciation is tricky: it requires the computer to have some notion of the value of its own output, but Colton goes on to suggest that this can be effectively some heuristic or classifier, perhaps encoded in a fitness function for an evolutionary art program. This seems to be setting the bar for what might pass as appreciation by a computer rather low, especially if we want to use it in the same context as the appreciation displayed by the programmer of the system and the human consumers of the outputs, as Colton does by including them in the tripod. Reducing human appreciation to relatively simple classification seems problematic, and Colton's own work in fact supports this. Colton suggests that appreciation could take the form of some internal implementation of Ritchie's framework discussed earlier in this chapter.

Colton observes that the importance given to process is different for different domains – lower for mathematics or science domains, and higher for art. This is captured in his own work on HR, which automatically generates mathematical theorems (Colton, 2001), and the Painting Fool, which is a system to produce "painterly renditions" (Colton, 2012:p 18).

Colton has also described the *FACE* and *IDEA* descriptive models (Pease and Colton, 2011; Colton et al., 2011), as part of *computational creativity theory*. This is an attempt to put computational creativity on a similar footing to machine learning and provide a "rigorous, computationally detailed and plausible account of how creation can be done" (Colton et al., 2011:p 90). It is also a reaction to what is seen as an over-abundance of evaluation approaches inspired by the Turing test, which are charged with encouraging front-end tweaking at the expense of background and contextual aspects of creative acts. The pursuit of outputs or artefacts that could pass for something produced by a human is seen as merely encouraging Chinese Room-style handle turning (Chalmers, 1992). In fact Pease and Colton (2011) state that as far as computational creativity is concerned it is "untenable to apply any defensible version of the Turing Test" (Pease and Colton, 2011:p 1).

The FACE model aims to capture both the process and the output of a 'creative act' by representing it with a tuple of at least one instance of 8 types of 'generative act'. These generative acts are essentially split into four types of output (*concepts*, *expressions of concepts*, *aesthetic measures*, and *items of framing information*) each with a corresponding method for generating them. A creative act need only contain one of these. FACE supports comparisons between creative systems in a few ways, for example a system that produces a concept and an expression of that concept is less creative than one that also produces an aesthetic to assess them against.

The tuples comprising the FACE model are incorporated in the IDEA model, which is designed to address the typical lack of "a-priori notion[s] of right and wrong" (Colton et al., 2011:p 92) in creative systems – that is, simple metrics for how good a solution, output or artefact is. IDEA swaps metrics of value for metrics of "impact of creations" (Colton et al., 2011:p 92). It assumes a system that will perform creative acts (as described by FACE) and is designed to produce outputs not predicted by the programmer. During development the programmer assesses the system and makes changes. Once done, the outputs of the system are assessed by an 'ideal audience', capable of giving two ratings to the resulting creative acts: *well-being* is a value between -1 and 1, representing strong dislike and liking respectively, with 0 being indifferent, while *cognitive effort* is a value between 0 and 1 indicating how much time the assessor was prepared to invest in understanding the output.

A distance measure is introduced to take account of both the process and the output from two FACE tuples, and thresholds are proposed for things that are too similar to known creative acts (including those introduced, or known about, by the programmer) and things that are too dissimilar and cannot reasonably be compared.

The ratings of the ideal audience can be used to calculate scores for *disgust*, *popularity*, *indifference*, *divisiveness*, and *provocation* (or how thought-provoking the artefact is).

These two models are intended to allow a degree of insight into the processes behind a creative act, including those that effectively represent the hand of the programmer or creator of the candidate creative system.

2.5 Reuse and Misuse of Knowledge

Design and problem-solving necessarily depends on the appropriate use of knowledge – this follows from our earlier discussion of problem solving: when we solve problems we generally do not randomly generate and evaluate solutions (although we discuss some techniques in 2.6 for incorporating randomness into the creative process.) Rather, we think about the problem in terms of our knowledge and previous experience. If all design is knowledge-based (Rosenman, 1997), then creative design is perhaps distinguished by how we use that knowledge, which means how we choose it and adapt or combine it. Misuse suggests that it is being used or adapted for use in some other context than that with which it is usually associated. One driving factor behind this could be the design problem or scenario itself – as Rosenman puts it, "The lesser the knowledge about existing relationships between the requirements and the form to satisfy those requirements, the more a design problem tends towards creative design" (Rosenman, 1997:p 643). The relationship between knowledge used, the scenarios or contexts with which it is usually associated, and the problem or scenario in which it is currently being applied may be where Schank's misuse and Poincaré's "unsuspected kinships" are found. However, the line between reuse and misuse is not easy to draw. We might say the purpose of a saw is to cut things, and using it for another purpose (e.g. as a musical instrument) is misuse. But is using a saw to cut a cake misuse or regular use? The function and the useful aspect of behaviour are the same, but the context is different. Or, we could use the same behaviour for a different purpose or function – for instance, sawing wood to produce sawdust rather than smaller pieces of wood.

Another example of function and misuse can be found in the two strings problem described by Maier (1933). Two strings are hanging from the ceiling, too far apart to be held at the same time, and the problem is to tie their ends together. There are various objects to hand including a pair of pliers. The solution is to use the pliers as a pendulum weight on one string, setting it swinging and making it possible to hold both at once. Like saws, pliers are strongly associated with a particular function, but that function is irrelevant to this particular solution. This experiment suggests that people tend to hew closely to these familiar associations at the expense of other ones.

When it does occur, this distortion of purpose or familiar use can surprise us, as well as stymie us. Familiarity suggests expectations or preconceptions, and if those preconceptions are challenged we are surprised. This familiarity can be with the usual purpose or behaviour of an artefact we are using, or perhaps with the sort of design we would expect to find in a familiar scenario. Design and problem-solving are about producing desired behaviour or effects, and creative designs are surprising if they challenge our preconceptions of what those effects should be or how they might be produced. Of course not all creativity is necessarily about the misuse of physical objects as in the saw example or the two strings problem, but they provide good illustrations of the concepts.

2.5.1 Adaptation and Degrees of Misuse

In the examples in the previous section, there was no adaptation of the structure of the saw or the pliers, and no new behaviours were exploited – the tools were simply being used for a different purpose than usual. However, adapting an artefact or design does not necessarily lead to more surprising misuse. Recipes are favourite examples in knowledge reuse literature (Aamodt and Plaza (1994); Schank et al. (1999); Kolodner (1991, 1992)). Following an existing recipe for, say, blueberry pancakes is straightforward reuse. Adapting it to make blackberry pancakes is arguably closer to misuse than reuse but the end result is not very different to the original one, and it is hard to imagine many people finding this creative. If we adapted the recipe to make muffins we might be getting closer to misuse, and if it could be adapted to help in, say, chemicals manufacturing then we might be closer still. The link between the original function of the recipe and its new function has arguably been broken rather than simply stretched a little.

The misuse of knowledge is consistent with ideas of some sort of creative insight, if the insight is knowledge-based – for example, to perform some action or exploit some affordances with some reason to think that this will lead to some desired outcome. This implies that some sort of similarity between the current problem and some knowledge must be identified. It seems that this recognition of similarity to some existing knowledge is key to problem-solving, both creative and routine. This is what is going on when we solve all sorts of everyday problems (Kolodner, 1992): we reuse the knowledge that worked last time, and the similarity between 'last time' and 'this time' is usually obvious. Solutions to these routine problems are likely to be unsurprising; if we simply do what we did last time the outcome will be predictable and familiar.

If, somehow, we identify one of Poincaré's "unexpected kinships" then instead of doing what we did last time, we might produce some more surprising results through the misuse of knowledge. In order to do this, we must identify and exploit some useful relationship between that knowledge and the current scenario that is not usually considered when we simply reuse knowledge.

The fact that solutions of this type are remarkable rather than commonplace suggests that this is easier said than done. *Mental set* can lead to domain experts being less flexible when it comes to considering knowledge outside of their domain by fixating on more familiar options (Wiley, 1998); this is also known as the Einstellung effect (Bilalić et al., 2008). Perhaps creativity also involves the ability (or a greater ability) to overcome this, splicing contexts to produce outputs that are out of context and so more surprising. That is, just as reusing things from different contexts to the current one might lead to a result that might be perceived as being more creative than choosing something from the same context, so the act of moving between or across contexts is itself part of the creative process. However, mental set is not intrinsically a bad thing, and might even be preferable much of the time. As Hofstadter et al. (1994) have it: "Time and cognitive resources being limited, it is vital to resist non-standard ways of looking at situations without strong pressure to do so. You don't check the street sign at the corner, every time you go outdoors, to reassure yourself that your street's name hasn't been changed." (Hofstadter et al., 1994:p 63) This is consistent with the idea that creativity arises in scenarios when it is necessary, or at least not disadvantageous, to do something different or unfamiliar.

But while such a scenario might be necessary, it is not sufficient. The right knowledge must also be available. As Mednick (1962) puts it, "It should be clear that an individual without the requisite elements in his response repertoire will not be able to combine them so as to arrive at a creative solution. An architect who does not know of the existence of a new material can hardly be expected to use it creatively." (Mednick, 1962:p 222). This suggests that circumstance and opportunity may also have a role to play in creativity. Circumstance may also be what separates an insight that an individual has not previously made (*P-creativity* (Boden, 1990)) from one that nobody else is known to have made before (*H-creativity*). In this regard, instances of P-creativity are just as important to the study of creativity, if not more so than H-creativity. The interesting thing is how an individual does it, which is independent of whether it was done before.

However, this is not to suggest that creativity is simply a product of necessity, opportunity or resources, nor that famously creative individuals might not have some unusual abilities, either in kind or degree, that could enhance these processes. Gabora touches on this idea with her 'Beer Can theory of Creativity', when she wonders whether some people might arrange their memories and knowledge in such a way that richer, more diverse relationships between them are possible, allowing more creative use, or misuse, of that knowledge. In her beer can analogy, the plastic rings holding a six-pack together highlight that how items are connected, or associated, is as important as their individual properties. The richer the web of associations between items, the greater the flexibility of reuse and the potential for this to lead to misuse. As Gabora puts it, "to *adapt* the idea to a new context, in order to *evolve* it in new directions, it must have been stored in memory in a way that implicitly identifies its relationships to *other* ideas." (Gabora, 2002b:p 147)

2.5.2 Association and Bisociation

The ability to recognise potentially useful knowledge is key to misusing it. This means that, when appropriate, we need to be able to recognise associations between knowledge or ideas beyond the ones we would normally recognise and operate with. Gabora's beer can idea discusses how some people might find this easier than others (Gabora, 2002a).

Associative hierarchies (Mednick, 1962) describe how flexibly individuals can make associations between different items, concepts, or knowledge. Creative individuals are thought to have flatter hierarchies while less creative individuals have steeper ones. This means that for a given stimulus, starting point or cue, people with flatter hierarchies can form more associations than those with steeper ones. Experimentally this can be measured by counting how many words individuals can come up with in response to a key word – the more associated words, the flatter the hierarchy and the more (potentially) creative that individual is. In other words, a person with a flatter hierarchy will be reminded of a wider range of things, and so make a richer set of associations, and it is this diversity that might support a creative insight. The results of this insight might be surprising to those who have not made the same associations.

Rich associations and some sort of focus are becoming recurring themes, with this richness perhaps arising from the way knowledge is stored and accessed and being driven by contextualised interactions. Gabora also recognises the importance of some sort of filtering or throttling of 'arbitrary' associations in order to focus on 'meaningful' ones, and makes an interesting analogy with genes (Gabora, 1997). Linkage equilibrium (the emergence of stable combinations of genes over arbitrary ones) reminds us that the associations that we make between different concepts are as much a product of context as any intrinsic properties.

Grace et al. (2012:p 195) observe that "the construction of a new relationship between two concepts or ideas, is . . . at the heart of many creative endeavours." They suggest that to associate two things we need to represent them in such a way that a mapping between them that represents the association can be supported. The development of representations and mappings are very closely coupled. They use interpretation-driven association, which means taking two things and altering them until they can be interpreted in such a way that an association can be recognised between them. (This has some parallels with the ‘creative explaining’ proposed by Schank and Kass (1990) and Moorman and Ram (1994).) Interestingly, they are interested in evaluating the creativity of the associations themselves, rather than the properties of any resulting outcomes or artefacts. This is consistent with the view that creativity involves some understanding or appreciation of how something was done. They go on to suggest a sort of metric of utility, in which how creative an association is can be related to how many affordances it supports. They use ‘free associations’ to describe those associations that are ‘goal-agnostic’, i.e. not constrained by necessarily having to produce anything useful. An important part of this idea of utility is that an association which might support further, sequential associations has more value than one that does not. This is slightly different to the idea that the creative utility of an association is related to the effect that it enables directly, or the ‘quality’ of that association.

2.6 Procedural Creativity

Other methods and mechanisms for generating novel outputs in domains often associated with creativity have been explored, by specifying or identifying processes which can, effectively, explore search spaces on behalf of the user. However, these processes cannot typically take all the credit for any interesting properties of their outputs: they often operate on components pre-defined by domain experts, or have their outputs modified either by built-in heuristics or by the direct intervention of the user. Music lends itself well to procedural, algorithmic or generative creativity, and artists have been exploring these possibilities for centuries. In the 18th century, dice games were a popular way of ‘composing’ music, with a roll of the dice (or some other way of choosing a random number) determining the sequential selection of pre-defined ‘snippets’ of music that could be combined to produce a composition. It was recognised that the value of musically ‘correct’ output was greater than that of simply varied output and so the components of the systems were carefully chosen (Cope, 2000). In a dice game version attributed to Mozart called the *Musikalisches Würfelspiel*, the snippets consist of 11 different versions of each bar of a minuet that can be combined in any order (Collins and d’Escriván, 2007). This kind of approach is stochastic, meaning that the output is different every time the game is played.

More generative approaches are more deterministic, and will produce similar outputs according to their rules every time they are run. Eno (1996) and others have been exploring the possibilities for generating complex musical output from relatively simple sets of rules since the 1970s. Many approaches are based on observations about the structured nature of music, and the potential of applying structures (or means of generating them) found in other domains. Biles (1999) attempts to identify the difference between merely translating a structure or sequence into sound, and composing music that humans might appreciate. Using a program (PGA-1) based around the Fibonacci sequence as an example, he argues that a relatively literal interpretation of an algorithmically-generated structure is unlikely to demonstrate the balance between expectation and surprise which people appreciate in music. Consequently, PGA-1 has its sequences modified by a human composer based on heuristics for producing 'engaging' music.

(Smith et al., 2012:p 160) describe a system which composes music "inspired by non-musical audio signals" – for example, bird song or speeches. Elements of these signals are mapped to a sequence of notes which mimic the original signal, with some musical heuristics applied to tidy up the results. An interesting aspect of this work is that it could incorporate some opportunism or serendipity – if the system were embodied in a situated agent, then the inputs could be provided by anything it happens to encounter.

More recently, Iamus is a musical program that evolves its own style, and is perhaps unique in that human musicians are willing to perform its output. Iamus uses an evolutionary algorithm, but uses indirect encoding rather than the direct encoding of more traditional approaches. This means that a small genotype can map to a larger and more complex phenotype, in a process inspired by developmental biology. Quintana et al. (2013:p 100) claim that this allows such systems to be more "disruptive", and perform an analogue of brainstorming. Iamus learns to compose in a similar way to humans, rather than learning how to produce examples of a particular type. Humans are not directly involved in the evaluation of candidates, although the fitness function consists of a large set of rules developed in collaboration with human musicians. This is similar to the heuristics applied by the human user of PGA-1, in that it exploits learned, shared responses that human listeners tend to exhibit rather than relying on generated structures alone. Ball (2012) notes that the audience has a role in making music – a composition needs to resonate with a listener's experiences and associations before it is appreciated, and music is usually composed with appreciation in mind. This interaction can be biased by preconceptions about who, or what, composed it.

Oblique strategies (Eno and Schmidt, 1978) are a rather different, non-mathematical approach to generating music or other outcomes. They are an aid or prompt to human creativity, rather than a means of generating creative outcomes directly. Originally developed in the 1970s, the strategies take the form of a set of cards, each bearing a short strategy, aphorism or observation that can be used as a kind of creative catalyst. A card can be drawn at random from the 'deck', and, when considered by the user in their current context, provide inspiration for some lateral thinking. Some example strategies are

Only a part, not the whole

Nobility of intentions

Use something nearby as a model

Other strategies for inspiration include the set of surrealist techniques used by artists, for example automatic drawing (Carr, 2003). This involves subconscious, involuntary movements of a pen or brush rather than a conscious expression of shapes or lines. The extent to which these approaches provide a useful starting point or are an end in themselves is subjective.

2.7 Natural Creativity

Natural processes, most notably evolution, are capable of producing outcomes or artefacts that might strike observers as surprising, attractive, or as representing novel solutions to problems. In other words, they might pass a creativity Turing test, although Bentley (1999) argues that, despite this, evolution does not explore search spaces in "innovative [or] efficient" (Bentley, 1999:p 4) ways, making it hard to consider it creative. In addition, although evolution can be considered to be knowledge-based (since all candidate 'solutions' are based on previous ones) it does not transfer knowledge between domains. Evolution is not limited by mental set or contextual bias, and in Section 2.5.1 we suggested that the ability to overcome these things is a key part of creativity. However, implicit in our discussion of knowledge-based creativity is some internal model driving the selection, adaptation and evaluation of knowledge. This is lacking in evolution as well as natural systems displaying emergent behaviour, or the appearance of global complex forms from local interactions (Cariani, 1990).

2.8 Summary

Existing literature paints a faceted but cohesive picture of creativity which supports the idea that it could be reproduced, or at least mimicked, by computers, in line with the debate around other areas related to artificial intelligence. Some recurring themes in how creativity can be conceptualised are apparent. Conceptual spaces provide boxes for ideas to occur both inside and outside of, allowing us to think of creativity and novelty in terms of new and existing, well-understood spaces respectively. They also let us consider how existing spaces can be explored or transformed, or how new ones can be created. Spaces can also be defined by sets of affordances (action spaces) or parameters, which can be influenced by how agents perceive and interact with their environment.

Another common theme is the difference between the reuse and misuse of knowledge, and our preference for the former over the latter. Misuse can represent the exploitation of "unsuspected kinships" and can be seen in the use of familiar objects performing unusual functions, as discussed in Section 2.5 and illustrated by Maier's strings experiment.

However, we also encounter the idea that creative artefacts must be valid, useful or appropriate – mere novelty is not enough, while too much novelty can lead to artefacts being harder to appreciate. Similarity is important to allow us to recognise when something is different from what has come before, but also to provide a context to understand and evaluate it.

Frameworks for evaluation provide an outputs-focussed perspective on what it is to be creative, with different approaches either taking process into account or disregarding it. If process is not deemed to be important, by using some generative or procedural approaches we might end up with a system apparently behaving creatively but in fact simply applying some preprogrammed rules. The argument about *being* creative and *appearing* to be creative will always apply to a greater or lesser degree to any computer or agent that has been programmed to behave a certain way, just as it does to many other areas of artificial intelligence. That said, pre-loading a set of heuristics is closer to an expert system, and while such a system could decide what is likely to be deemed creative in a given domain or context (which includes the preferences of the people or processes doing the evaluation) this would be a very limited sort of creativity: it could not generalise to different domains, or use it to solve new, immediate problems it might encounter. It would be incapable of any sort of creative insight and while it might pass a sort of Turing test, there is no obvious relationship with how people are thought to 'do' creativity, which is of more interest to us if we are interested in a capacity for behaviour like Boden's P-creativity for computers.

Cognitively, creativity is widely agreed to be an extension of normal processes, and therefore based around the same mechanisms with differences in degree rather than kind. In this sense we might think of creativity as a construct or artefact of those processes when they occur in remarkable scenarios or lead to remarkable outcomes. Human problem solving and design are knowledge-based, and the intelligent, directed misuse of knowledge appears as both a characteristic of some creative artefacts and outputs and a way of producing them. Misusing knowledge can cross-pollinate ideas, confound expectations, and generate new conceptual spaces to explore, with context provided by the regular, routine use of that knowledge to help understand them. However, we have also seen that regular use of knowledge is often adequate, if not actually preferable. In fact, people have a strong tendency to operate routinely where possible as cognitive misers, although they can evidently also behave creatively despite this tendency. Without this generally routine behaviour there would be no context for creativity.

Associative processing of implicit knowledge is found to have an important role in insight and intuition, which can work in tandem with more explicit, rule or symbolic based reasoning, and while implicit knowledge can be overruled by explicit knowledge it can also neutralise it. We have discussed the idea that people might have different intrinsic capabilities or attributes in this regard, as well as be subject to different external influences or drivers like problem scenarios they encounter and different or more extensive resources like experience and knowledge they can bring to bear.

However, the differences between computers and people mean that what works for people might not work for computers, even if they were performing the same sort of tasks under the same sort of circumstances or contexts. Likewise a remarkable insight made by a person might be less remarkable for a computer which organises and accesses its knowledge differently.

According to Hofstadter et al. (1994), creative people display an ability for selecting knowledge in unfamiliar scenarios such that what is retrieved is "typically a small set of concepts that "fit like a glove," without a host of extraneous and irrelevant concepts being consciously activated or considered." (Hofstadter et al., 1994:p 37) Compare this with a computer which could, in principle, search through its entire memory and evaluate every piece of knowledge it has against a problem, or perhaps generate a solution from scratch by exploring a much larger search space than a person could, especially considering their tendency to consider only a small part of their knowledge. These approaches might give us better or more novel solutions than we would get by constraining them to be cognitive misers like us, although likely at the cost of efficiency. However, the context for some tasks we might want computers to perform for us might favour fast, familiar, predictable solutions over slower, more novel ones.

Chapter 3

COMPUTATIONAL CREATIVITY

In this chapter we discuss some existing work systems, frameworks and representations supporting problem solving, reuse and adaptation of knowledge, and analogy. Not all the work described claims to be producing, or even concerned with, creativity, but is relevant in the light of the work discussed in Chapter 2.

3.1 Exploratory and Transformational Creativity

Much of the work discussed in Chapter 2 involves generating new outputs by performing some operations on some existing concepts. These are consistent with the conceptual spaces encountered in Chapter 2, where we observed that creativity is not found in an existing conceptual space, like the one defined by the rules of a musical genre (although novelty might be). Rather, it is found in altering a conceptual space, or generating a new one, to allow new possibilities.

A conceptual space can be manipulated using a variety of techniques, for example mutating the values of parameters defining a space. Gero has looked at mutation as a means of introducing variation into a design prototype or schema by changing the values of parameters, stating that a designed artefact may be "broadly interpreted" (Gero, 1990:p 31) in terms of groups of variables describing its function, structure and behaviour. The relationship between these properties of artefacts has been explored by several researchers, with Gero notably using it to explore design and creativity with his Function-Behaviour-Structure (FBS) framework (Gero, 1990). In FBS the *structure* or physical make-up of an artefact dictates its *behaviour*, which then allows it to perform its *function*. Function, or purpose, depends on some subset or aspect of behaviour but cannot be directly inferred from structure, which means we might find it easier to describe an object or note some of its properties or behaviours than guess what it is *for*. For example, the properties of a Frisbee mean that it can be used as a plate or bowl, but that is not its function. The *no-function-in-structure* principle (De Kleer and Brown, 1986) lays out the idea that the function or teleology of an artefact is largely context-dependent, and that structure and behaviour can change while function does not.

The FBS framework provides a useful way to consider various aspects of the selection, generation and adaptation of both abstract and more concrete structures. It is also useful when considering what it means for things to be similar, which is a recurring element of work on both the reuse and the misuse of knowledge. The framework is most obviously suited to physical objects but the principle holds for less tangible things.

Gero also identifies a set of processes that form relationships and transformations between function, structure and behaviour:

- Formulation (maps function to projected behaviour)
- Synthesis (maps projected behaviour to structure)
- Analysis (analyses behaviour actually produced by the structure)
- Evaluation (compares the actual and projected behaviours)
- Documentation (produces a design description)

Gero has also explored the role of *situatedness* in problem solving, recognising that "situatedness is concerned with locating everything in a context so that the decisions that are taken are a function of both the situation and the way the situation is constructed or interpreted" (Gero, 1998:p 168).

Gero's situated function-behaviour-structure framework expands on the FBS approach discussed above by recognising that problem solving (or at least designing) occurs in dynamic, rather than static environments, as noted in Chapter 1. Situated FBS aimed to move away from the pre-encoded knowledge that the original FBS framework implied, and towards a framework in which an agent's knowledge is "grounded" in its experiences of interacting with its environment.(Gero and Kannengiesser, 2004:p 374) In this framework, the environment in which the design agent is situated allows the agent to incorporate feedback: it sees the effects of its actions and refines its view of the problem accordingly. Situated FBS relies on the interactions between three different but interdependent *worlds*: The external world (the 'real' world in which the agent is situated), the interpreted world (the real world as the agent perceives it), and the expected world (an 'updated' version of the interpreted world, as the agent predicts it will be after performing certain actions). The agent interacts with the external world to try to remake it in the image of its expected world. Gero assumes a constructive memory, which is analogous to a process of interpretation in which external data can produce internal representations by pushing from the outside, and interpretation can be driven by existing concepts pushing from the inside (or 'pulling'). The key idea here is that the agent's internal representation of its situation "wires itself up" (Gero and Kannengiesser, 2004:p 379) through interactions between the real world and the interpreted world. Situated FBS incorporates *focus*, corresponding to pursuing a particular approach or concept by generating behaviour and structure, and relies on constructive memory to provide domain knowledge by producing extra variables for function, behaviour and structure. 'Situatedness' is effectively distributed across the interactions between the constructive memory, the internal model of the world the agent maintains, and the real, or external world.

The main role of situatedness in this approach seems to be as a feedback mechanism to tune solutions, by providing domain knowledge based on previous knowledge – although this can also lead to the 'pulling' of new concepts (changes to the interpreted, internal world). While this is necessary for creative design and problem solving (and, for that matter, routine design) it does not seem to be sufficient – the role of situatedness is responsive rather than proactive.

Gero has proposed *design prototypes* as a standard, reusable schema for describing a design concept (Gero, 1990). This allows a design or concept to be represented in such a way that the space it describes can be altered computationally. Behaviour and function can also be represented to a certain extent by schema (Gero et al., 2005; Finke, 1996; Aamodt and Plaza, 1994). Gero considers *routine* and *non-routine* design in terms of schema (Gero, 1996). Routine design involves following a known and well-defined schema, while non-routine design involves altering the schema so that it produces different results. Gero and others have looked at some ways in which alterations to a schema might be introduced in a computational model (Gero, 1994; Saunders and Gero, 2000), outlining three mechanisms through which schema can be altered by the addition or substitution of variables: *combination*, *mutation* and *analogy*. Combination is the addition of variables from another artefact. Analogy is similar except that the variables or sets of variables are chosen through the identification of analogies between the current artefact and a source artefact. Mutation is the alteration of a variable by changing it to some other value.

In addition to the transformations mentioned in 3.1. Gero describes three additional *reformulation processes* that can change the design space by altering one each of the original function, structure or behaviour variables (Gero and Kannengiesser, 2004), potentially facilitating transformational creativity. Each of these can be potentially influenced by a design or problem-solving agent's current situation, via the introduction of new examples of either structure, behaviour or function into the design space. Gero also explores the application of schema to validating the appearance of new things. He outlines a process in which new designs are evaluated to see if an existing schema can be found to describe or understand them – if not, and no new schema can be produced, then the design is rejected. This has parallels with Boden's conceptual spaces in that new things must exist in some known or describable space if we are to be able to understand and accept them.

Gero also discusses the importance of *emergence*, which can apply to structure, behaviour or function. This is not the same emergence as in emergent behaviour in complex systems (Cariani, 1990). Rather, what is emerging (or perhaps more accurately being revealed or recognised) is some ‘new’ property that an artefact may have already that is not captured by the current schema being used to describe it. If that schema is replaced by another, the emergent properties become tangible. On a similar theme, Finke’s Geneplore model (from ‘generation’ and ‘explore’) (Finke, 1996) describes how ‘preinventive’ structures or images are generated. Preinventive forms are emergent forms or structures that appear from combining existing structures or forms but are waiting for a function to be ascribed to them. They are ‘preinventive’ because they have not been purposely developed for a particular task. Emergence in this context might correspond to the misuse of knowledge, in that knowledge of the behaviour of an artefact in a particular context (and function) can be applied in a different context (to perform a different function). This different function can be a new one or perhaps an existing one that the artefact is not usually associated with. We are also reminded of the representational affordances discussed in Section 2.1. Finke lists some examples of generative processes such as memory retrieval, association, mental transformation and synthesis, and analogical transfer. Again, these processes are taking place at a relatively high level compared to some we will look at later.

These approaches are consistent with both exploratory and transformational creativity. Exploratory creativity involves finding a previously ‘unused’ point in an existing and possibly already well-explored space (Boden, 1990). The space can be defined by a generative grammar or be an n-dimensional parameter space. However it is defined, the constraints of the space do not change and a new artefact is represented by a new permutation of parameters. The language examples mentioned previously would come under this definition, and as we observed earlier artefacts produced this way are perhaps more likely to be novel than creative. Boden considers exploratory creativity to be more tractable for computers than the transformational and combinatorial approaches to creativity we will look at later, although she does note that defining the space in the first place requires a lot of domain knowledge (Boden, 1994).

Kötter and Berthold describe a method for recognising missing concepts in graph-based knowledge networks. That is, concepts that are supported by things that are encoded in the graph but are not explicitly encoded themselves. They contend that to ‘discover’ these missing concepts is to make an insight into connections between node that are not explicitly connected. These connections are “hidden in the noise of the integrated data” (Kötter and Berthold, 2012:p 230). They use a statistical approach to identify concept nets in large corpora that have not been explicitly identified, although they do not seem to assign labels to those concepts. This approach is based around a threshold for a measure of the relationship between vertices and edges.

In transformational creativity, the space itself is changed so that it is possible for new things to exist within it – things that could not have existed in it previously. This can be done by altering the parameters defining the space, perhaps by swapping or substituting parameters from other spaces. Boden considers transformational creativity to be more powerful than both exploratory and combinatorial creativity because it allows new ideas to arise that were previously *impossible* rather than ones that are just *improbable*.

Wiggins (2006) also defines some techniques by which a creative agent might perform transformational creativity, on his way to showing that transformational creativity can in fact be considered as meta-level exploratory creativity – that is, a strategy for exploratory creativity working on the representations of conceptual spaces rather than just representations of things *in* those spaces. He suggests separate rule sets to define and traverse a space, with the former being objective and the latter varying – there might be multiple traversal rule sets for a given definition rule set. There is also a set of rules covering evaluation, which work in combination with the traversal rules. Wiggins also shows that, according to this approach, it may be possible to distinguish "what is in *principle* possible in a creative domain from what is *actually* possible according to the properties of a given creator" (Wiggins, 2006:p 453). This is consistent with the idea touched upon in Section 2.5.1 that not all people necessarily have the same 'creative potential' in a given scenario – their approach is influenced by the knowledge they possess and their experiences with it, or the contexts in which they have used it.

3.1.1 Constraint Satisfaction

We have already mentioned limits, constraints, and their role in 'thinking outside the box'. Constraint satisfaction is an area of interest in artificial intelligence domains such as planning. Problems can be defined in terms of sets of parameters and corresponding constraints, with solutions consisting of values for each parameter that satisfies the corresponding constraint. However, some problems may be *over-constrained*: there may be no solutions which satisfy the original constraints, but changing them may allow a solution that is 'good enough' and still useful. Constraints can be modified or *relaxed*, which changes the original problem. This is different from *removing* the constraint completely, and thereby allowing a variable to have any value from its domain; in relaxation new constraint(s) effectively replace the existing one. Constraints can be soft, meaning there is less of a penalty if they are adapted or not met than for a harder one. Constraint satisfaction problems (CSPs) for which a full solution is not possible are known as partial CSPs. Concepts of hard and soft constraints and constraint relaxation can be mapped to conceptual spaces. Changing or relaxing a constraint can change the space of possibilities. Of course, choosing which constraints to relax is important. *Circumscription* is similar to the closed world assumption found in ontologies, in which things that are not known to be true are considered to be false (McCarthy, 1980). For example if a problem describes 3 blocks we would not base a solution around any extra blocks – although the problem does not explicitly state that there are no more, there is no reason to think that there are. Circumscription was developed to get around the need for knowledge representations to account for every conceivable condition, known as the *framing problem.* It allows us to assume that things that are not explicitly stated can be disregarded. It also allows us to assume that, when an action is performed, only the things explicitly affected by that action are affected.

3.2 Combinatorial Creativity

Combinatorial creativity is the process of combining existing knowledge in new ways, and we use it here as a mechanism which might then inform or drive exploratory or transformational creativity. If 'active' knowledge defines a conceptual space within which outcomes for a particular scenario must be generated, then combining knowledge in new ways can generate new spaces, and consequently, new outcomes. The knowledge being combined might be concepts or domains, and this is essentially the *bisociation* put forward by Koestler (1964) when he suggests that the roots of creativity lie in knowledge or concepts the mind already has – they are just combined in new and unusual ways. He coined the term 'bisociation' to describe this and to distinguish between routine thinking, requiring only a single plane, and creative thinking, requiring more than one plane. A plane in this context is a frame of reference, so a link is made between creativity and combining thinking from different frames of reference, implicitly recognising the importance of analogy in creativity, since some similarity or compatibility between these planes must exist in order for this to be useful. Lawson observes that "no amount of thinking on [plane] P^1" (Lawson, 2003:p 100) can reach a target located on a different plane, while Gero calls creativity "the production of an unexpected result through the confluence of two schema" (Gero, 1996:p 436). Underlying this combination of existing knowledge and ideas is the ability to (at some level) make analogies or identify similarities. However, Boden does not consider combinatorial creativity to be capable of producing surprising new ideas – at least compared to transformational creativity, which she regards as the most powerful form of creativity.

3.2.1 Analogy

Identifying or making an analogy is an opportunity for combinatorial creativity, by transferring properties from one thing to another. Lawson (2003) identifies analogy as one way that Koestler's planes might be linked, going on to describe a neural model of analogical reasoning. Goel (1997:p 63) states that "analogical design involves reminding and transfer of elements of a solution for one design problem to the solution for another design problem". What these elements are can vary according to the problem; they could be a particular arrangement of components or the shape of a single component. Analogies can concern functionality or behaviour as well as structure or appearance. However, as Goel points out, analogy does not have to be confined to helping combinatorial creativity by selecting elements of existing solutions to use in a new solution. It can also be used to help identify and reformulate problems, assess solutions and anticipate potential problems with candidate solutions. Using analogical reasoning also helps the problem-solving process by making more cognitive resources available for more novel aspects of a particular scenario (Mair et al., 2009).

Analogies are typically classed as either intra- or cross-domain, with much research being focussed on the latter. From a creativity point of view cross-domain analogies are likely to be more interesting, in that they are likely to lead to more novel (and theretofore surprising) combinations of knowledge and scenario, if the domains are "semantically distant" (Shen and Lai, 2014). Hey et al. (2008:p 285) argue that "potential for creative problem solving is most noticeable when two domains being compared are very different on the surface". Similarly, O'Donoghue and Keane (2012) suggest that the difference between creative and 'ordinary' analogies is in the 'distance' between the domains being linked, with the mechanism being the same – this is consistent with other observations about the relationship of the creative process to more routine activities (Stojanov and Indurkhya, 2013). Their model of creative analogies has three phases: *retrieval, mapping* and *inference validation.* This last is concerned with whether any new inferences about the target that follow by analogy with the source are correct. This does not preclude them from conflicting with other, existing inferences – indeed such conflict might give rise to the surprise and confounded expectations that are often part of creativity. The retrieval phase concentrates on structural similarity of graphs describing domain structure rather than semantic similarity, in the hope that this will avoid "semantic narrowness" (O'Donoghue and Keane, 2012:p 18) and drive a more diverse (yet still constrained) selection.

Analogies are typically found or recognised between two existing concepts or cases, but Baydin et al. (2012) have looked at how, given a base or source case, an analogous case can be generated as well as the analogy between it and the source. Their domain is semantic networks, which are graphs representing the relationships between concepts, with the edges describing the relationships between concepts at the nodes. They have expanded on Clement's observation that analogies need not necessarily be recognised between a given case and another already selected from a knowledge base (Clement, 1988). They used WordNet (Miller et al., 1990) and ConceptNet (Liu and Singh, 2004) as knowledge bases to generate analogies for the relationships between the objects in the solar system. The knowledge bases are used to ensure that only valid networks are generated. Since there is no selection of target cases there is no reuse of knowledge in the sense that we have discussed earlier. However this approach still gives new relationships between the elements making up the new semantic network and those in the problem network.

Gentner and Markman (1997) argue that analogy has much in common with similarity although they are often considered to be distinct, with analogy "clever [and] sophisticated" compared to similarity's "brute perceptual process" (Gentner and Markman, 1997:p 45). They argue that both exist in the same space of similarities, with analogy occurring where attributional similarity is low but relational similarity is high. An example of attributional similarity is physical similarity with attributes such as colour, number of edges, and so on. This type of similarity does not have the predictive power of relational similarity, which is concerned with the relationships and bindings between different elements of the things being compared. This is useful because making the analogy supposes that behaviour produced by these relations is transferable from source to target, allowing predictions of behaviour in the target. The structure mapping theory described by Falkenhainer et al. (1989) is extended from analogy to similarity.

Stojanov and Indurkhya (2013), however, argue that analogies based on surface or perceptual similarity can be *more* powerful for creative problem-solving than those based on existing conceptualisations, which can limit the space of potential solutions.

3.2.1.1 Copycat

Copycat (Hofstadter et al., 1994) is an attempt to reproduce the 'mental fluidity' that humans display in making analogies. Copycat's domain is letter strings which are transformed into different strings through analogy with other, reference transformations. A two-part example from Hofstadter et al. (1994:p 32) is:

> Suppose the letter-string abc were changed to abd; how would you change the letter-string yk in"the same way"?
>
> Suppose the letter-string aabc were changed to aabd; how would you change the letter-string ykk in"the same way"?

Copycat is designed so that it is, in principle, generalisable to other domains, but an interesting claim is that it has some *understanding* of entities in its domain that other, less abstract systems might not have of theirs. It is based on the idea of *conceptual slippage*, which is itself an analogue for recognising similarities between applicable concepts. In conceptual slippage, the concept of changing the last letter in the string for its alphabetical successor can 'slip' into that of changing the first letter for its alphabetical predecessor. This slipping is driven by some pressure, which in this case is that the solution to the first part of the example does not appear to be a very good solution to the second part. This reminds us of our observations in Chapter 2 that an important element of creativity is the need to be creative in the first place. Hofstadter recognises that this conceptual slippage must in some way be focused, guided and selective.

Copycat's architecture has three main elements: a long term memory containing a network relating all of the system's concepts (the *Slipnet*), a short term or working memory containing instances of concepts (the *Workspace*), and a collection of agents waiting to execute tasks (the *Coderack*).

The Slipnet is context-dependent: the edges between concept nodes represent their 'conceptual proximity', which determines how easy it is for one to slip into the other. This proximity is updated according to Copycat's current 'understanding' of the situation. Each concept node also has a weight describing its 'conceptual depth', which describes how abstract it is, or how hard it is to identify in the situation. Hofstadter's claim is that the 'deeper' a node is that is associated with a problem the more significant it might be, and so potentially more likely to have a large effect on the solution. This is a good fit with the idea that the more abstract an analogy or similarity between two concepts or problems, the more different their 'default contexts' might be and hence the more likely the solution is to tend towards the creative. However, the deeper the conceptual depth the harder it is for conceptual slipping to occur. Again, this fits with previously mentioned ideas: that the processes underlying creativity will typically take the line of least resistance, that creativity reluctantly comes into play when routine will not do, and that creative behaviour differs from routine in degree rather than type.

Another important feature of the Slipnet is that concepts are not actually represented by a simple node but rather distributed across an activation radius which encompasses multiple nodes (reminding us somewhat of CLARION). However, each concept must still have a discrete core to allow slippage between concepts to occur. Hofstadter claims that concepts in Copycat are therefore *emergent* rather than explicitly defined, although this is more like the 'refocussing' emergence discussed in Section 3.1 since they are still composed of pre-defined elements.

The workspace initially contains all the raw data from the problem, and is where agents from the Coderack work to provide descriptions and structures using concepts from the Slipnet. The way the agents work on things in the workspace is determined by their *salience*, which is effectively a measure of the extent to which their descriptions and structures are based on highly active concepts in the Slipnet. Objects in the workspace are probabilistically selected in pairs and evaluated for similarities which can be concretised as bonds, allowing them to be combined into higher-level structures called *groups*. These groups are themselves evaluated for similarity in the same way as their constituent objects were. Similarities between the two structures representing the opposing parts of the original problem are themselves instantiated as bridges, which combine to form a *viewpoint* of the problem. This viewpoint is emergent in that it is generated by agents whose actions influence those of other agents without any common overall goal. Again, viewpoints can effectively compete for dominance, leading to what Hofstadter calls "revolutions", which correspond to potentially radical reassessments of the problem – or, perhaps, creative leaps.

The agents (or *codelets*) in the Coderack are independent and have limited functionality, but are ultimately driven by the state of both the Slipnet and the Workspace so are indirectly working towards the same goal. Each agent has a probability that it will get a chance to work in the Workspace, based on its relationship with the state of the system. So agents working on a currently strong viewpoint have a high probability of being selected quickly. There are different types of agents, which Hofstadter claims represent different pressures or drivers found in problem-solving: *bottom-up* agents are open to any relationship, description or structure they can recognise and so represent general problem-solving. *Top-down* agents, on the other hand, represent specific aspects of the current problem and situation. Copycat, then, is intended to be a model of the flexible reuse of knowledge discussed earlier.

3.2.2 Metaphor

Metaphor implicitly recognises similarities between its source and target, and can be thought of as an expression of an analogy. Like a good analogy, a good metaphor is based on the application of some salient properties of the source to the target. While a novel or ingenious metaphor can be an example of creativity in itself, the recognition of a metaphor-supporting relationship between two concepts or artefacts might be useful for combinatorial creativity, as an indication that two concepts have at least one transferable property which could perhaps be exploited to solve a problem. Veale and Hao (2007) describe a system called Sardonicus which can generate new metaphors from a case base of metaphors and similes drawn from the web. This case base is generated by generating an extensive list of antonymous adjectives and inserting them into a search for terms including:

as [ADJ] as a *

where 'ADJ' is an adjective and * is the wildcard that allows this search to return metaphors featuring a wide range of nouns. Both nouns and adjectives can be associated with several potential sources and targets respectively. For a given noun, for example, the system can identify several adjectival properties, while for a given adjective it can find multiple nouns with which it has been associated. The use of the web allows the system to effectively incorporate 'real-life' knowledge of the kind not typically included in formal knowledge-representation repositories designed to be manipulated by computers. Metaphors generated by Sardonicus can be validated by searching for them again on the web. If they are found they are valid; if not they are either novel or invalid, where an invalid metaphor would be one that fails to convey a meaningful relationship between target and source. The system can generate valid but novel metaphors by relying on categorisation. For instance, it can apply adjectival properties it knows to be associated with a noun representing a generic concept to a more specific example of that concept.

Hey et al. (2008) suggest that metaphor is a dominant tool in the early stages of the design process. They distinguish metaphor from analogy in the context of design by noting that while analogies are usually concerned with relational or structural similarities, metaphors can be richer and encompass *appearance* similarities as well. A key difference is claimed to be which elements are mapped and how they are used in the design process.

3.2.3 Conceptual Blending

Conceptual blending is a cognitive process that is closely related to analogy and might support some aspects of combinatorial creativity. Fauconnier and Turner (1998) suggest that blending, or conceptual integration, drives analogy and that although blends are obvious in remarkable cases, blending is actually a "workaday" process that generally operates unnoticed. It involves the merger of at least two input spaces into a 'blended space' (Li et al., 2012). In other words, the mapping of elements of concepts contained in one space onto elements of concepts contained in a different space, for example a butcher's cleaver mapping onto a surgeon's scalpel. Work on blending has tended to focus on combination of concepts or spaces, rather than on how these are selected for blending, when the process should end, and so on. Goals and context can be used to drive the selection of input spaces, the further selection of particular concepts within those spaces, and stopping criteria for what Li et al. (2012:p 9) call 'blend elaboration'. Goals and context are elements of situatedness and we have already recognised how important this, as well as constraining the type of blends we are willing to consider (Hofstadter et al., 1994), can be for creativity. However, they do not consider the role of standard mappings from context to solutions, or how conceptual blending might be related to mental set. They do, however, recognise the need to avoid a 'combinatorial explosion' of blended concepts.

According to Fauconnier and Turner (1998), conceptual integration involves projection between structures, allowing viewpoints, frames and scenes to be connected. Projections are usually partial, to allow the construction of spaces with structure not found in any of the input domains; this is "emergent" structure as discussed in Section 3.1, but the value of the blended space is that it is still connected conceptually to those domains. Fauconnier and Turner present a taxonomy of networks of connected spaces, which can include input spaces, generic spaces, and blended spaces. The degree to which concepts in blended spaces are constrained by the inputs spaces or frames varies in these networks. Blending is susceptible to bias from routine knowledge and scenarios.

Costello and Keane (2000) investigate the effect of constraining the combination of concepts. They describe a process designed to mimic both the efficiency and creativity with which people produce novel compound phrases. Constraints in the Constraint-guided Conceptual Combination (C^3) model are *diagnosticity, plausibility* and *informativeness.* The model is for interpreting noun-noun compound phrases, not generating them. People use ‘discourse context’ to generate conceptual combinations of these phrases – that is, possible interpretations of any noun are narrowed down by the context in which they are encountered (although they can also provide their own context for any following words). There are different ways in which people interpret the relationship between the nouns, they argue, some more common than others, which drive their interpretation of the phrase. The constraints are based on ‘pragmatics’, which are a set of assumptions a listener might make about what someone is telling them. These assumptions are that the speaker believes the listener will know what they are talking about, that the two words chosen are the best ones available, and that these two words are both necessary and sufficient – that is, any candidate concept that could be either conveyed by fewer words or needs more to describe it unambiguously is likely incorrect. These map to the plausibility, diagnosticity, and informativeness constraints respectively. (In this context, ‘diagnostic’ means that the phrase conveys the characteristics or properties that best define the concept and so best allow the listener to recognise it.) If these constraints are met as the listener constructs their interpretation of the phrase, it is likely to be the one intended by the speaker. In this work, ‘creativity’ is treated as synonymous with ‘diversity’. Nonetheless it echoes themes of appropriateness and constraints present in much work on creativity.

3.2.4 Cross-Domain Transfer

Morris et al. (2012) describe an ‘automated chef’ that can take things from different inspiring sets (as defined by Ritchie in (Ritchie, 2007)). They observe that creating hierarchical taxonomies allows for transfers high up in the hierarchy that are manifested as more striking combinations lower down (reminding us of the ‘conceptual depth’ discussed earlier in this chapter). This is essentially the same as observing that how surprising juxtapositions of things are depends heavily on context and the degree to which those things are generalised. For example, finding a motorbike where we would expect to find a car is surprising if the context is cars, less so if it is ‘motorised wheeled transport’. As an aside, the advantage of using recipes is that less tacit knowledge is needed to assemble solutions or examples – there is little or no syntax or rules involved, elements can be swapped in and out or removed without having to worry about how they need to be handled when the recipe is actually carried out. Solutions are described at a high level and the lower level work is handed off to the cook; Morris simplifies this further by making all the recipes ‘one-pot’.

3.2.5 Case-Based Reasoning

Case-based reasoning (CBR) is a methodology for solving problems with existing knowledge, by applying solutions from selected cases from a case base to new problems. Useful cases are chosen based on the similarity of the new problem and the case problem, and the case solution is then adapted to fit the new problem. This approach is widely thought to be similar to the human problem-solving process (Schank, 1983; Aamodt and Plaza, 1994; Carbonell, 1993; Gentner, 1983) and has obvious relevance in a discussion of routine and creative problem-solving. CBR is based on what is known as the *similarity assumption* (Smyth and Keane, 1998) or *CBR assumption*, which is essentially that similar problems have similar solutions. How similarity is calculated depends on the domain and case representations, although it can be as simple as Euclidean distance. Selection can be based on "apparent, syntactic similarity only" (Aamodt and Plaza, 1994:p 5); indeed, one of the strengths of CBR is that it can work when there is no domain model. However, in addition to the perhaps unstructured knowledge contained in the cases a CBR implementation may have other knowledge resources or domain models to call on to help it select and adapt cases. CBR is a lazy methodology in that the cases are only evaluated when a new problem is presented to the case base or case memory. This also means that new cases or experience can be added very easily and no re-training is needed. Aamodt and Plaza (1994) summarise the CBR approach as:

- RETRIEVE the most similar case(s),
- REUSE the case(s) to attempt to solve the problem,
- REVISE the proposed solution if necessary, and
- RETAIN the new solution as a part of a new case.

The fourth step means that CBR systems can incorporate some learning capability. This might include some mechanism to form new relationships between cases and problems. CBR can be thought of in terms of analogy; Aamodt and Plaza (1994) describe analogical reasoning as, essentially, CBR that can work across domains – in other words, CBR with a wider scope, which we mentioned above as being important for a creative case-based system and also explored in Section 3.2.1. They go on to point out that cross-domain analogies bring up the *mapping problem*, which is "finding a way to transfer, or map, the solution of an identified analogue (called source or base) to the present problem (called target)." (Aamodt and Plaza, 1994:p 5)

As a methodology, CBR can have a variety of implementations and Aamodt and Plaza (1994) also provide a useful summary of different approaches which vary largely according to how concepts contained within cases are stored and accessed. For example, in *Instance-based reasoning* we might expect to see a larger number of relatively simple cases (perhaps consisting of feature vectors) with little or no domain knowledge. In *Exemplar-based reasoning* concepts are defined by a set of their exemplars, and new problems are classified against them. *Analogical reasoning* (which we will discuss in more detail later) focuses on cross-domain matching of cases and problems, which has implications for both the selection and modification of cases.

CBR grew out of Schank's work on dynamic memory (Schank, 1983). A dynamic memory contains memory organisation packets (MOPs), which represent sequences of events that can themselves consist of re-usable components that can be shared with other MOPs. MOPs are similar to frames, which are data structures that also contain some contextual information, originally developed by Minsky in an attempt to make AI techniques more applicable to real world problems (Minsky, 1975). Frames, and the frame-systems in which they exist, are an attempt to make concepts more generalizable. Minsky intended that they represent 'stereotyped knowledge', but with some capacity for flexible reuse by altering some of their elements or slots at lower levels, leaving the immutable, higher level elements intact – the things that are always true or applicable for that particular concept. We might think of them as having flexible examples of Coleridge's hooks surrounding their key elements. A dynamic memory also contains explanation patterns which are like cases for explaining events – previous explanations for events fitting a certain profile are likely to be good places to start for events with similar profiles. In CBR, cases are "contextualised piece[s] of experience" (Bergmann et al., 2005:p 1) containing a problem and a solution, and as such are usually specific rather than generalised – that is, they only describe a specific episode, rather than a generalised description of episodes of that type. However, in (Aamodt and Plaza, 1994) as well as these 'concrete experiences' there can be generalised cases based on sets of similar ones, in either a flat or hierarchical index structure.

3.2.5.1 Representation

In CBR, general domain knowledge is not represented explicitly; there may be no model of the domain that a case occurred in, just a record of some past events. Bergmann et al. (2005) summarize three basic approaches to representing cases: *feature vectors*, *structured representations* (for example object-oriented representations and frames), and *textual representations.* Feature vectors have n dimensions describing some feature space, and similarity between them is calculated using some distance metric – although this similarity does not guarantee usefulness in all domains. Object-oriented representations are an extension of object-oriented programming; cases consist of collections of objects with attributes described by key-value pairs. Finally, textual representations are used in domains which lend themselves to textual descriptions. Similarity in the latter case is typically calculated using term frequencies, and textual representations may be mapped to more structured representations (Weber et al., 2005). Both textual and feature vector representations are potentially susceptible to noise, as well as issues with small differences leading to non-transferable solutions (although it is hard to think of many representations a computer could employ that would not have issues with the latter).

3.2.5.2 Selection

To exploit the similarity assumption mentioned in the previous section a CBR system must first ascertain which problems in its case base are similar to the problem it is currently facing. The issues surrounding the identification of relevant features and the subsequent selection and retrieval of cases in CBR are collectively known as the *indexing problem* (Maher and Gomez de Silva Garza, 1997).

However, indexing which is optimised for solving problems by reusing solutions from cases containing similar problems has its limitations. It aims for familiar reuse of knowledge, and is unlikely to also support more interesting or surprising misuse of knowledge. In addition, we might encounter a problem which is found to be very similar to one or more existing cases, but that has some wrinkle which prevents solutions from those cases being easily adapted. For example, if our problem is to allow people to cross a ravine, we might recall a similar problem in which we used a bridge. But if our new problem has a wrinkle – perhaps the ravine is too deep and wide for a bridge, or the budget does not allow it – then this solution is unworkable. In this scenario, the case base might still contain *other* cases that could be adapted (or misused) to solve the problem with more surprising results, but which might not be retrieved using similarity.

3.2.5.3 Adaptation

To access cases that might support a non-routine solution we need a different approach to selection than one based on the similarity assumption and designed around routine solutions. Smyth and Keane (1998) remind us that the similarity assumption only holds as long as the features used to calculate similarity are representative of the domain, and that some deeper domain knowledge is needed to apply any resulting cases or knowledge to a new problem or context. This 'adaptation knowledge' tells us whether a case can be modified to provide a workable solution. They go on to describe *Adaptation Guided Retrieval* (AGR) which involves some measurement of adaptability. In the Deja Vu system described in (Smyth and Keane, 1995) this takes the form of 'scripts' that can be quickly applied to each case, and these scripts appear to be hand-written for specific domains. For many CBR systems this is not really an issue as the domain is typically relatively narrow, constrained and well suited to representing problem elements as building blocks that can be assembled in a variety of valid combinations. For real-world systems this is more of a problem, and even more so for systems incorporating cases from different domains.

Whether we should discard the similarity assumption in favour of an AGR-like approach depends on what we expect of our creative system. For routine problem-solving (where the problem is similar enough to a case that the solution can be easily applied) the two are likely to lead to the same outcome – if there is a very similar case in memory it is likely that it is also the one that is best suited for adaptation. From a performance point of view AGR is likely to be more expensive, and hard to justify if the similarity assumption can generally be relied upon unless we are specifically interested in more unusual solutions.

Wilke and Bergmann (1998) summarize approaches to adaptation under two broad types: *transformational* and *generative* adaptation. Transformational adaptation includes substitutional adaptation (where new parameter values are substituted for existing ones) and structural adaptation (where parameters themselves can be changed). These approaches typically rely on a set of adaptation operators and rules, based on substantial domain knowledge. Whether they are appropriate depends on the domain and also the representation of the problem. As we go from higher to lower levels of representation, structural adaptation may become more likely to break a solution than improve it.

In generative adaptation, the *derivation* of the case solution is applied to the new problem, rather than the solution itself. This approach means that as the derivation is reproduced, changes can be made at the point at which it begins to break down in the new context. This means that extra information about the derivation of the solution must be stored in the cases. Transformational and generative adaptation may be thought of as top-down and bottom-up approaches respectively.

Wilke also describes *hierarchical* adaptation, in which the case base stores cases at multiple different levels of abstraction, with more conventional 'concrete' cases at the bottom of the hierarchy. Initially adaptation occurs at the highest level of abstraction, and becomes progressively more detailed. A drawback with this approach is the need for some training over the cases. On the other hand it does allow for some pruning of the case base, which helps to avoid being swamped with too many similar cases. PARIS (Bergmann and Wilke, 1995) is a CBR system incorporating hierarchical adaptation. The domain is engineering process planning, with the case base initially consisting of a set of concrete cases in a company archive. Planning domains consist of a set of states and operators to allow transformations from one state to another, described in this case using the STRIPS framework (Fikes and Nilsson, 1972). There are assumed to be different domains for abstract and concrete cases, and each domain has a different representation language. Concrete cases can have more than one abstraction or abstract case. PARIS uses two mappings to associate abstract and concrete cases: a *sequential mapping* and a *state mapping*. The latter requires some general domain knowledge. An algorithm generates abstract cases from concrete ones. PARIS was found to be capable of solving more problems in a given time limit by using abstracted cases than by using concrete cases, and (perhaps more interestingly from a creativity point of view) showed a much greater 'flexibility of reuse' – that is, an abstracted solution was found to be applicable to a much greater percentage of the remaining case problems in the case base than a concrete case. This approach can be considered as analogous to breaking mental set by avoiding fixation on local or low-level details or features.

In common with many CBR systems, PARIS operates in a relatively narrow and explicitly constrained domain. Solutions are plans or sequence of operations. Each operation is discrete and does not perform differently depending on its position in the sequence. This is not representative of many real world domains.

3.2.5.4 Abstraction and Generalisation

In machine learning, abstraction means the creation of a new concept from a set of instances, whereas generalisation means the extension of an existing concept to cover a broader set of instances. Both can support the identification of similarities between concepts (including cases and problems) in the absence of initial obvious similarities. Here we are using 'concept' to describe an idea or abstraction that exists at a higher level than the potentially disparate cases that might exemplify, incorporate or instantiate it. In order to flexibly select cases, we might like a CBR implementation to be able to generalise and recognise that, for example, cars and motorbikes are both motorized transport. We might also like it to flexibly create an abstraction of a case by extracting the relevant features or properties, and then see if other cases fit this abstraction. In other words, we would like our system to be able to work with concepts that might not be previously defined. The degree to which the difference between abstraction and generalisation is meaningful and important might depend on how much domain knowledge is incorporated into the system already, perhaps in the form of rules for selection and abstraction.

3.2.5.5 Creative JULIA

Creative JULIA was designed to address three of the main elements of CBR: memory search and retrieval, evaluation of alternatives, and updating a problem specification. Kolodner intended it to address both the deficiencies with CBR and the creative process in general, noting that one aim was to incorporate a trait she noticed in her own problem-solving process – a willingness to "consider odd proposals for what they might contribute rather than disregarding them outright because they wouldn't work" (Kolodner, 1994:p 5). The approach outlined by Kolodner is:

1. Retrieve a set of cases (based on the original problem specification)
2. For each case:
    - Evaluate the corresponding solution
    - Update the problem solution
    - Update the problem specification
3. Repeat until a satisfactory solution is found.

An interesting difference with conventional CBR is the second step, in which the problem specification is expanded on to allow the retrieval of a wider range of results. Additional constraints that do not contradict the original ones are added, or existing constraints are removed. Kolodner went on to note that many CBR systems actually skip *situation assessment* – that is, assessing which features of the current situation or context are relevant – and assume that there is sufficient information contained in the problem. In doing so they miss out on the potential for a broader interpretation of the problem and a wider scope for solutions. As people typically have to solve problems that are very closely coupled with their immediate environments (Withagen et al., 2012; Wilson, 2002) we might reasonably expect that situational assessment might be very useful to drive the interpretation, comparison, selection and reuse of knowledge. Again this echoes the themes of mental set and situational or contextual bias discussed throughout this chapter.

3.2.5.6 Fuzzy CBR

Fuzzy values correspond to the vaguer, less precise values we tend to give things in real life. For example, when people describe an object as 'warm' or 'hot' the corresponding temperatures of the object might be anywhere within two overlapping ranges – say, between 18 and 30 degrees and between 25 and 35 degrees respectively. In fuzzy logic (Zadeh, 1994), 'hot' and 'warm' are *linguistic variables*, each of which maps the temperature of the object to a value between 0 and 1. Depending on the mapping functions, a temperature of 30 degrees might have membership values of 1, and 0.65 for 'hot' and 'warm' respectively.

In fuzzy CBR (Subbotin and Voskoglou, 2011; Bonissone and Cheetham, 2001; Bonissone and De Mantaras) cases are described with fuzzy rather than crisp attributes, allowing for more flexible selection.

These fuzzy values can only be applied to numeric variables (Jeng and Liang, 1995), not symbolic ones. However, *type-2* fuzzy systems can account for uncertainty in the rules used to operate on numeric values. If these rules are based on the input of experts, uncertainty might arise if words used in them have different meanings to different people. Or different experts might have different ideas about what those rules actually do – that is, the *consequents* of rules might be different according to different sources (Mendel, 2000).

A type-2 fuzzy system, then, allows the degree of membership of a type-1 fuzzy set to itself be treated fuzzily.

Bonissone and De Mantaras also take a more general approach by noting that at the level of cases themselves there is partial set membership, since many cases in a case base will be neither totally useless nor totally useful.

An analogous approach might be to abstract concepts or structures until we identify some commonalities or analogies at a higher level. The ability to perform fuzzy CBR, then, would be a big step towards our goal of the flexible reuse of knowledge by computer, bringing them closer to how people reuse knowledge.

3.2.6 Computational Memory Models

Our memories allow us to be reminded of things in sometimes surprising ways, providing a mechanism to drive the selection of knowledge to misuse. Distributed, associative memories are widely thought to be good analogues of human memory (Kanerva, 1988; Anwar and Franklin, 2003; Olshausen et al., 1993). They can handle noisy data and degrade gracefully, just like our own memory. These memories are content-addressable, meaning that rather than storing and retrieving items in single locations using unique memory addresses (as in standard RAM) items are used as their own addresses for storage and retrieval. This means that items can be stored without using indexes or categories, which could potentially limit opportunities for their reuse. In this respect this type of memory is much 'messier' than RAM but it does have some very useful properties. Items presented to the memory as cues do not have to *exactly* match items in memory – they must only be similar. So memories from unrelated or slightly different experiences could be retrieved by a certain cue, as well as items that match it very closely.

Distributed memories also allow memories to interfere with each other, just as they do in humans. Since similar memories are stored in some of the same places, the item stored at a particular location will be modified by each new item stored there. No one item will delete and overwrite the rest, but rather the item stored there will become a composite of all of them. If the items are very similar, a distributed memory will essentially converge to a prototype common to all of them, even if this is something the memory has not actually seen before or which does not actually exist, reminding us of the 'creativity machines' mentioned in Chapter 1. These characteristics allow associative memories to be 'reminded' of things, and mimic some of the characteristics of human memory. If they are given several slightly different partial examples of, say, an image or figure, they can also return a composite, more complete image formed by combining these slightly different stored examples (Kanerva, 1988). As well as this ability to retrieve items given only fragments as inputs, they can generalise across similar inputs, prioritise more recent, frequent inputs over older ones, and be relatively resistant to local damage.

However, as pointed out by Gabora (2002a) earlier in this chapter, we can have too much of a good thing. While it would not be very useful if a content-addressable memory needed a cue to be identical to an item in memory for that item to be retrieved, neither would it be very useful if the memory returned too wide or rich a range of results – the process querying the memory would get swamped and function less effectively, just as people would find it hard to function if remembering or reminding involved recalling too rich a range of things from memory. As Hofstadter et al. (1994:p 64) puts it: "In general, unlikely ideas (or even ideas slightly past one's defaults) ought not continually occur to people [. . .] a person to whom this happens is classified as crazy or crackpot."

The Sparse Distributed Memory (SDM) model proposed by Kanerva (1988) supports the mechanisms identified above. Kanerva's model exploits the properties of very large dimensional vector spaces, particularly regarding the distribution of similar points within the space. Addresses are points in the space. Input vectors to be stored in the memory are compared to vectors representing randomly initialised addresses, and stored within those that are similar to within a certain tolerance. Similarity is determined using the Hamming distance between vectors. This 'storage tolerance' has some similarities with the steepness of an associative hierarchy, as discussed in Section 2.5.2. If it is tight, only very similar things will be associated or recalled for a given cue, corresponding to a steep hierarchy. If the tolerance is larger, the hierarchy will be flatter and a wider range of items will be retrieved, although potentially with the problems associated with too broad a range of retrieval mentioned above. The memory is said to be sparse because, like our own memory, it has the capacity to store only a small sample of all the possible inputs. SDM is generally used as a content-addressable associative memory – that is, the inputs and the items stored in memory are the same. It can support auto-associative and hetero-associative retrieval of items. Auto-associative retrieval means that presenting a fragment of an item as an input can retrieve the whole of that item from memory, while hetero-associative retrieval means that a *different* item would be returned. Implementations typically return a single, composite item for a given cue, but if we are interested in misuse, or combinatorial creativity, as part of a creative system it might be more useful to return all matching items individually for consideration.

Kanerva intended his model to be able to capture associative links between items, as illustrated with the following example from Kanerva (1988:p 2):

> Why are fire engines painted red? Firemen's suspenders are red, too. Two and two are four. Four times three is twelve. Twelve inches in a foot. A foot is a ruler. Queen Mary is a ruler. Queen Mary sailed the sea. The sea has sharks. Sharks have fins. The Russians conquered the Finns. The Russians' color is red. Fire engines are always rushin'. So that's why they're painted red!

This example plays on homonyms to link apparently unrelated concepts, but it is a good example of Coleridge's "hooks and eyes of memory" and it is easy to see how it could map to associative hierarchies and flashes of inspiration.

However, the retrieval of a rich set of items that might support the misuse of knowledge (assuming the system were not overwhelmed by them) would be the result of a rich set of inputs, rather than artefacts of the internal operation of the model. Small differences in vectors may represent correspondingly small differences in the things they represent, depending on the mapping between them, meaning we would need different inputs to retrieve items of a different type rather than degree.

Other interesting memory models include Kosko's Bi-directional Associative Memory (BAM) (Kosko, 1988) and Adaptive Resonance Theory (ART) (Grossberg, 1976; Carpenter et al., 1991). BAM uses a feedback network to increase the accuracy of associations between stored pattern pairs – that is, to make sure that the retrieved patterns most closely resemble the inputs. When an input is presented to the memory, an output is generated and bounced back to the input through the transposed weights to produce a new value for the input which should better match a stored input pattern. This is bounced back to produce a new output, closer to the matching stored output, and so on until the system converges to a stored input and output pattern pair.

ART is essentially a clustering algorithm for binary vectors (although ART 2 dealt with continuously-valued inputs). A new prototype is constructed for an input if it does not match a cluster to within a certain degree of accuracy. The data stored is a set of prototypes, one for each cluster. ART addresses the stability-plasticity problem (Carpenter and Grossberg, 2010) – that is, it does not have to forget something in order to learn something new. In ART, each node in the output layer has an associated weight vector, representing the weights of its connections with all the nodes in the input layer. This vector represents the item learned or stored in that node. In the first cycle, the output node with the closest match to the input 'wins' – the system is designed so that even if more than one item matches, inhibitors mean that only one will be active. The winning node sends its vector (or 'exemplar', effectively the pattern that is actually being stored) back to the input layer. This vector and the original input vector are ANDed to produce a comparison vector. The comparison vector and the input are sent to a reset layer and their similarity is calculated, with the result being compared to the vigilance threshold. If the similarity is greater than the threshold, there is a match. If not, the winning node in the output layer is disabled and the process is repeated. The input vector 'resonates' between the two layers until a match is found, or, if none is found, a new pattern is acknowledged and allocated to an unassigned or 'empty' node. There are clear parallels here with abstraction and generalisation.

These models are limited to relatively simple input vectors, although these could represent symbolic inputs. Austin (1996) describes work on using associative memory to perform fast rule matching, with symbolic rules being represented as binary vectors after converting each operator and value into a vector token. An example of a rule is:

$$NAME : JOHNANDALIVE : TRUE- > AGE : 33$$

Austin acknowledges that a limitation of connectionist implementations is that they are very susceptible to the order of inputs, i.e. the elements in the inputs vectors. Even though rule preconditions are commutative, if the order of preconditions as presented to the memory were different from that stored the relevant rule would not be retrieved. In addition, inputs to associative memory models are typically fixed length. Austin gets around this by superimposing all the matrices representing rule preconditions into a single matrix, which is then converted to a 1-dimensional vector for input. The system can handle rule conflicts (when more than 1 rule matches the input conditions) and can deal with partial matches.

3.3 Creativity, Explanations and Understanding

Schank and Kass (1990) and Moorman and Ram (1994) make the point that understanding or explaining problems is an important part of solving them. The latter make the distinction between the *generative creativity* we have been discussing for most of this chapter and *explanatory creativity.* Understanding in this sense is in some ways similar to finding analogies or reformulating problems: it involves constructing a model or perspective in which inputs or problem inputs make sense. When we find analogies in combinatorial creativity we are searching for ways in which a new problem (or part of it) makes sense in the context of some existing knowledge. Moorman and Ram claim that creative understanding has occurred if novel artefacts have been comprehended in a useful way.

3.3.1 ISAAC

Moorman and Ram (1994) describe the CUP (creative understanding process) algorithm, which consists of 4 stages:

1. Memory retrieval
2. If 1 fails - attempt *incorporation*
3. If 2 fails - attempt *baseless analogy*
4. If 3 fails - reformulate the problem

This cycle continues until a useful concept is produced or the current working concept becomes too bizarre to be useful. *Incorporation* involves identifying relationships between concepts found in memory and the ‘problem’ concept. *Baseless analogy* involves using existing domain knowledge to generate a new concept with which an analogy can be made, if one cannot be found. CUP is incorporated into ISAAC (Integrated Story Analysis And Creativity), a system for developing creative understanding in reading which is capable of resolving conflicts between artefacts it retrieves from memories and artefacts it encounters in stories. While ISAAC only worked with one story and involved a fairly limited, predefined knowledge structure with various categories to which things could be assigned, the 4 stages described above (and their sub-processes) are a good fit with the broad CBR approach that has characterised our discussion of the relationships between knowledge, problem-solving and creativity to this point. The goal of understanding the novel concepts in a story is effectively the same as understanding how an artefact or structure can be useful in a given problem context, and doing so might be thought of as making a creative leap or flash of insight.

3.3.2 SWALE

SWALE (Schank and Leake, 1989) is a case-based program designed to creatively explain things it has not come across before, which it does by drawing comparisons with things it *has* seen before (i.e. things it has in memory). It retrieves the relevant match (decided by an algorithm) and modifies it to come up with an explanation. Items in memory are stored as explanation patterns (XPs) representing the indices of a particular concept. For example, SWALE comes up with possible explanations for the unexpected death of the racehorse of the same name, based on XPs in its knowledge base representing other unexpected deaths with known causes. It does this partly by recognising the fact that there are anomalies in the story of the racehorse and looking for similar anomalies in its cases.

3.4 TRIZ

TRIZ, or the theory of inventive problem solving, is a set of principles and tools aimed at supporting the solution of non-routine, or creative problems (Webb, 2002). At the heart of TRIZ is the notion of *contradictions* which must be resolved – for example, the need to make something lighter while also making it stronger. A given problem is abstracted until it can be viewed as a set of pairs of these contradictions, which are arranged in a *Matrix of Contradictions.* Each row/column index has a subset of *inventive principles* which are suggestions for approaches to resolving the contradiction. The principles and contradictions have been distilled from the study of millions of patents. The concept of an *ideal final result*, or IFR, is another core element of TRIZ. The IFR represents a solution in which all of the problem goals are met without compromise, and all constraints are removed. The value of the IFR is in removing bias and preconceptions, and allowing a less restricted search space to be considered.

TRIZ has parallels with case-based reasoning, in its assumption that every new problem will turn out to be similar in some way to a previously encountered problem in a (very large) knowledge base, and also in its reliance on recognising those similarities. Like CBR, it relies on previous cases, except that where CBR (in principle if not necessarily implementation) lazily assesses each case at run time and can learn, TRIZ relies on a fixed set of principles distilled from those cases in advance.

TRIZ does not claim that 'inventive' is necessarily synonymous with creative. However, despite the rather mechanical approach TRIZ is, in a sense, misusing knowledge, although in this case the difference between misuse and reuse has been flattened out by the prior generation of the matrix and the inventive principles. In other words, the line between misuse and reuse has been blurred by maintaining a lookup table of how misuse can occur. More recent versions of TRIZ have included techniques deliberately aimed at introducing novelty – effectively these are extensions of the inventive principles. TRIZ is a tool for people rather than computers but its principles could be encoded in a computer or agent.

TRIZ was originally focussed on engineering-type problems but has since been expanded by researchers and commercial organisations to cover different design domains.

3.5 Models of Creative Cognition

Wiggins (2012) describes a model of regular cognition out of which creative cognition might emerge, or at least creative inspiration. He defines 5 criteria that a system claimed to be a model of human cognition must meet: *Falsifiability*, an *evolutionary context*, *learning capability*, *production capability*, and *reflection.*

Falsifiability simply states that the system should not produce behaviour demonstrably different from that of people. *Evolutionary context* says there should be an argument for an evolutionary advantage the model would confer, since this is presumably how people came to have the capabilities they do, and they are our reference. *Learning capability* suggests that an agent should be able to learn more about its domain. Clearly there must be some *Production capability* – that is, the agent can actually *do* things. Lastly, Wiggins specifies *Reflection* – the system must have some awareness of its own behaviour, and some way of explaining it.

A key point of the work is that the agent contains and maintains a reliable model of the domain, and uses it to predict and compare rather than simply to respond. This prediction and comparison give a platform for expectation, although this is not the same expectation that gives rise to surprise on the part of those assessing outcomes. The approach proposed by Wiggins is an extension of the global workspace theory proposed by Baars (2005), which posits a set of expert systems called 'generators' working in parallel and competing for access to a *global workspace*, which holds only that information of which an agent is aware at any given time. It also contains one 'thing', which is what supports its interactions with its environment. The generators are effectively competing to provide this 'thing'. A problem with this model (according to Baars himself) is that there is a threshold for access to the workspace that must be crossed in order to put a particular thing in it. The global workspace itself should decide this, but in order to be considered a particular generator would first need to have access to the workspace – which is what it is trying to gain. Wiggins proposes an approach for managing access to the workspace that would get round this difficulty, and also rather neatly provide a mechanism for creative inspiration. Wiggins uses a metaphor of 'loudness': predictions made by multiple generators are louder than those made by individual ones, and the loudest ones win out. Wiggins uses Shannon's concept of information entropy (Shannon, 1948) to provide a mechanism by which an agent might hesitate or be cautious in situations where it cannot confidently predict what will happen next. Entropy is maximised when all outcomes are equally likely, and minimised when only one is certain. Likelihood is matched to expectation, so unexpected things are considered to be very unlikely. (However it is left unclear how this ultimately manifests itself in more careful, yet still (presumably) purposeful, behaviour of the kind that the approach is designed to support, distinguishing it from mere responsive models.) The generators shout louder in the presence of something unexpected, providing a way for unexpected things to be noticed. Wiggins tweaks the mechanism in such a way that there is a bias against both very likely and very unexpected predictions, speculating that this might explain how agents and organisms avoid getting overwhelmed by choice.

Wiggins makes the step to supporting creativity by looking at situations when there is effectively no competition from generators that demand attention – there are no sensory stimuli to drive the 'volume', and outputs that would normally be drowned out have a chance to reach the workspace. However this is not elaborated on, in particular the fact that there is still presumably a need for competition – if there is still babbling, albeit quieter babbling, then something still needs to be heard above the rest if the 'one thing in the workspace at a time' constraint still applies. The path from a random entry to the workspace to a creative outcome or artefact is also not explored, although Wiggins points out early that the model only covers inspiration, and the creative reasoning necessary to transform inspiration to artefact is outside the scope of the work.

Wiggins concludes that in this model, creativity could be a side effect of what is primarily an "essential survival mechanism" (Wiggins, 2012:p 186), in an evolutionary context.

3.6 Summary

A key theme identified in this chapter are the transfer or projection of attributes between two domains, to enable new inferences or possibilities in the target domain. Another is the identification, retrieval and adaptation of existing knowledge in order to solve problems. Much work in this section can be thought of in terms of work discussed in Chapter 2, especially combinatorial and transformational creativity, and how to specify and explore conceptual spaces and relate items of knowledge. Combinatorial creativity has at its core the selection and adaptation of knowledge, and work on analogy and metaphor is relevant here. It might be seen as a mechanism to drive transformational creativity (which Wiggins argues can itself be considered as meta-level exploratory creativity).

Much of the work in this chapter is consistent with the process of combinatorial creativity, and issues such as maintaining diversity while at the same time limiting the negative effects of considering a wide range of options at once. Hofstadter's Copycat is a notable example of work addressing several aspects of combinatorial creativity and flexible reuse, including intelligent selection, injection, adaptation and combination of existing knowledge. According to Hofstadter, "the awakening from dormancy" (Hofstadter et al., 1994:p 36) of a relatively small number of prior concepts is key. Many of these concepts are encountered in Case-Based Reasoning (CBR). Although CBR might appear to be best suited to more routine problem-solving than creative, the issues of selection, representation and adaptation of knowledge that it raises have wider implications which seem likely to be crucial to creativity. In Chapter 2 we saw that creative behaviour is generally considered to differ from routine in degree rather than kind, and this degree could be depend on the flexibility of each of the '4 Rs' mentioned in Section 3.2.5.

However, the similarity assumption – that similar problems have similar solutions – underpins the idea of selecting the most similar case for reuse and revision of case knowledge in CBR. This assumption might have its limits if what we want is to *misuse* knowledge in the manner discussed in Chapter 2. Smyth's Adaptation-Guided Retrieval (AGR) shows that similarity and adaptability (or potential for reuse, and therefore also misuse) are not necessarily the same, while the PARIS system showed that abstracted cases showed a greater flexibility of reuse than concrete ones. Maier's two strings experiment mentioned in Section 2.5 is consistent with these concepts and also with the concept of emergence discussed in Section 3.1.

The concept of generating solutions from the bottom up rather than by adaptation from the top down is encountered in Copycat and Wilke and Bergmann's generative adaptation; both of these are still knowledge driven.

Memory models can display some of the same behaviours as human memories, dealing with issues of similarity, retrieval, and situatedness, all of which are relevant to solving problems by using existing knowledge. However, there are limitations on what might be represented and stored in an actual associative memory implementation, making them better suited to recognising surface or perceptual similarity than relational similarity, when relationships and mappings between different elements of the things being compared have been identified. Gentner and Markman argue that analogy occurs where relational similarity is high and surface or attributional similarity is low. However, Stojanov argues that analogies based on surface or perceptual similarity can be more powerful because they are not based on existing conceptualisations.

We discussed situatedness but see relatively little consideration of these concepts in the context of wider problem-solving behaviour; the generation of creative outputs is assumed to be the end rather than the means. Wiggins suggests that as well as the ability to produce outputs, a model of human cognition (of which creativity is a part) should provide a plausible evolutionary context to account for how this is done. This suggests that the characteristics of human problem-solving should be considered in the contexts in which it occurs. This is consistent with observations in Chapter 2 that people may encounter scenarios in which non-routine problem solving behaviour should be discouraged, and do in fact tend to display a bias towards routine solutions.

Chapter 4

A MODEL OF OPPORTUNISTIC, ROBUST KNOWLEDGE-BASED PROBLEM SOLVING

In this chapter we describe a model that could help a computer to solve problems robustly by biasing its search based on knowledge from cases selected in response to new problem scenarios. The model is designed to solve problems as routinely as possible, but artefacts of its operation might occasionally provide opportunities to solve them non-routinely. This would be an example of the misuse of knowledge, and we suggest it would be behaviour with some parallels with some examples of creativity, though the model is not claimed to be a model of human creativity.

4.1 Model Rationale

Our approach is inspired by the characteristics of human problem-solving behaviour as discussed in Chapter 1. The context of the model is similar to the one in which people generally operate, where 'good enough', efficient, and robust problem-solving is usually preferable to doing extra work to produce surprising, unpredictable outcomes. We use 'robust' here in the sense that the same knowledge can be used to solve problems the same way it has been previously, even when problems are not encountered in exactly the same way.

We have observed in Section 1.3 that people occasionally display creative behaviour but generally do not – in fact, they tend to do the opposite. However, like people, a computer would benefit from some capacity to produce a beneficial outcome when an appropriate solution is lacking, or perhaps insufficient.

Inspired by the discussion in Section 2.1, the model is based around the idea that affordances associated with objects identified in a problem scenario can define a conceptual space, which both constrains and focusses the nature of solutions that can be considered. Objects are identified based on using the world as its own model, inspired by the discussion of the importance of associative assessments of problems in Section 2.3.1. A knowledge base contains cases, containing episodic knowledge describing how objects of different types were manipulated to generate solutions, and so affordances associated with identified objects of the same type can be extracted from these cases. This means that an agent or system can act based directly in response to its interactions with, and perception of, the world in which it operates. Solutions are effectively being built from the bottom up around these affordances, rather than performing actions after generating a model of a scenario.

The model relies on the assumption that case knowledge involving objects of a particular type will be strongly biased towards using those objects consistently, or routinely. In other words, the affordances of a given type of object that are routinely exploited in existing cases can be assumed to be the relevant ones when objects of that type are encountered in new scenarios. This is analogous to assuming that if a pair of pliers is found in a problem scenario, the solution will involve gripping something with the pliers, if that is what pliers are always used for in other scenarios. A set of assumptions like this can significantly reduce the search space, which would otherwise include all the other things that *could* be done with a pair of pliers – for example, they could be used as a pendulum weight, as in Maier's strings problem described in Section 2.5. Considering all of these possibilities is inefficient if using them to grip will solve the problem.

The more objects in a scenario have distinct, routine uses the more the search space can be constrained and the more efficiently and routinely problems can be solved. This is useful behaviour if the model encounters routine problems frequently and non-routine ones rarely (as people do) and generating non-routine solutions is generally not the main aim. However, if objects are not used consistently or distinctly and biases are weaker then more possibilities must be considered, the search space is less constrained, and problem solving is less efficient.

A given problem scenario might also activate other, apparently superfluous knowledge. This can be a consequence of using the world as its own model if the world is noisy and cases can be recalled using partial matches. Or, it could be an artefact of the mapping process from scenario to cases. This knowledge cannot be applied routinely if the objects in the activated cases are not found in the scenario, but it could be evaluated for potential non-routine use, if the opportunity should arise.

This would involve the cross-pollination of knowledge between cases, by applying knowledge (i.e. affordances) associated with case object types to *different* object types in the scenario.

Opportunities for unconventional solutions, then, arise when the subset provided by the knowledge activation process cannot be conventionally applied to produce a solution, or such a solution is insufficient. As suggested in Chapters 2 and 3, unconventional solutions might be more interesting or surprising, and if the model should produce any *despite* its bias towards conventional ones then its behaviour will share some characteristics with that displayed by people when they solve problems creatively. A set of mappings from activated knowledge to the scenario defines a conceptual space. A process similar to combinatorial creativity is occurring if this involves the transfer of affordances from items in a case to items in the problem scenario in an unfamiliar way. However, analogies are not *explicitly* recognised in the model. We might expect that opportunities to successfully misuse knowledge in this way are rare.

The model is consistent with ideas and themes discussed in Chapters 2 and 3, including conceptual spaces, combinatorial and transformational creativity, mental set, the importance of associative or implicit processing, and the idea that creative (or at least non-routine) problem-solving behaviour differs from the more mundane in degree rather than type. However, a key point of the model is that the unconventional application of knowledge should generally *not* be required. The model, and the assumptions on which it is based, aims to ensure that any available knowledge that can be used conventionally to produce a solution is promoted wherever possible.

4.2 High-Level Description

The discussion in Section 4.1 implies two main processes. The first selects or activates a subset of cases from a case base based on some interaction with, or perception of, a problem scenario. The second process attempts to generate a solution based on and constrained by that knowledge. There is no interaction between these processes beyond the one-way exchange of cases, and the second process is limited to solutions that can be constructed using knowledge contained in these cases – the rest of the knowledge base is unavailable to it.

The first process is associative and should support partial matches (Coyne, 1990). Useful cases are identified based on perceived similarity with the problem scenario rather than by reasoning over some model of the problem or domain. This is likely to lead to cases being activated that contain knowledge that is not necessarily relevant to the current problem, with no indication of which knowledge might be redundant. The implementation of this process will be domain-specific.

To implement the second process we use a genetic algorithm (Goldberg and Holland, 1988). Genetic algorithms (GAs) are search heuristics that take their inspiration from evolution, in common with other evolutionary computation techniques. We are interested in combining 'snippets' of knowledge from selected cases, working under the assumption that those snippets have significantly constrained the search space such that we expect to reach a routine solution quickly. A GA allows us to easily bias the exploration of a search space by effectively fixing parts of the chromosomes of candidate solutions using these snippets. As well as exploring the space of possibilities for adapting a given subset of available snippets, it can be configured with a meta-population to search for optimal subsets. This is useful as we have no *a priori* knowledge of which snippets might be useful. GAs also avoid the need for domain knowledge and models to drive the adaptation of cases; as we have seen in Chapters 2 and 3, such models are difficult to implement.

The role of the GA in the model is to effectively replace tacit knowledge and handle the adaptation of cases into the new scenario, rather than to act as the primary means of searching the whole of the search space. It attempts to use the knowledge conservatively or conventionally by generating, where possible, solutions in which knowledge is used the same way as it is in the cases from which it is sourced. Knowledge snippets extracted from cases are treated as *assumptions* about what operations should be performed on one or more objects in the scenario. A *lead* is made up of a set of these assumptions, and can contain knowledge from multiple cases. Cases are weighted according to the strength of their perceived association with the scenario. Using these weights, cases are selected probabilistically to contribute to leads. Leads are themselves similarly weighted based on the aggregated weights of their contributing cases, which represent the confidence that they contain knowledge that will support a routine solution. A lead is complete when each object in the problem scenario has been assigned some parameters based on their corresponding assumptions. If no assumptions have been assigned to an object, parameters are randomly generated. The second process does not know which assumptions that it can extract from a particular case might be useful, or even why that case was selected.

For each lead, a population of candidate solutions consistent with the lead assumptions is generated. When the GA is run, a lead is selected probabilistically based on its confidence weight to provide a candidate for evaluation. After a certain period, lead weights are re-calculated to represent their actual performance, rather than their confidence. If a routine solution is available, it should be performing well and so be allocated more resources, leading to a rapid routine solution. When all candidates in a lead's population have been evaluated a new population is bred. At intervals, the worst performing leads are replaced with new leads, which can be either variations on existing leads or generated from cases.

If the case base contains a problem very similar to the new one that can be solved routinely then it is to be expected that the solution will be found very quickly.

A high-level view of the model is shown in Figure 4.1.

In summary, the key attributes of the model are:

- The world is used as its own model

- The second process has no direct influence on the knowledge that goes in the subset – it must work

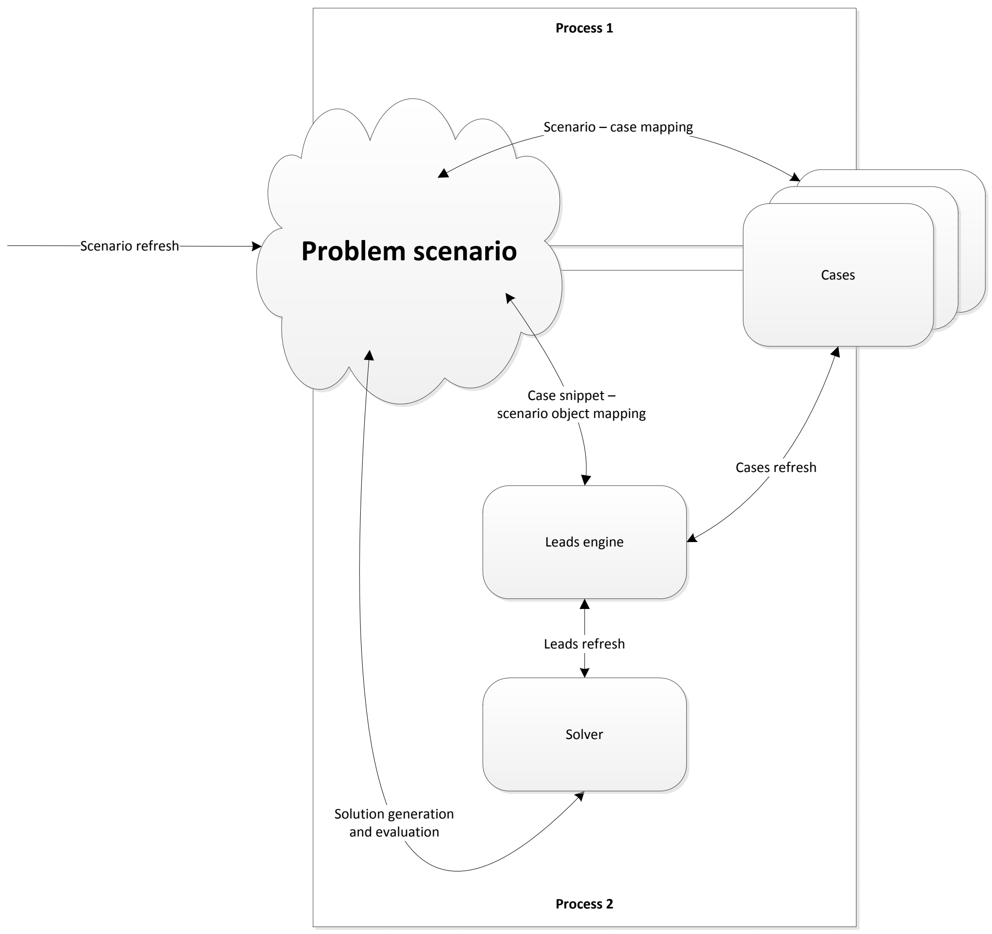


Figure 4.1: A model of problem solving showing how there is no direct communication between process one and process two.

with the materials it is given

- The second process is essentially performing the role of applying explicit knowledge to make concrete some insight that has been made implicitly (Hélie and Sun, 2010)
- Convention – the system aims to use available materials as routinely as possible (e.g. using a spoon as a spoon)
- The system operates in a context where similar problem scenarios tend to involve exploiting similar affordances identified in them – use of object types is consistent
- The system encounters routine problems much more frequently than non-routine ones
- The system has a consistent means of mapping scenarios to useful cases in its knowledge base (as some of the highest ranked ones). So for a routine problem the set of materials is not likely to be very diverse from a creativity point of view, which is what we want if we are mostly interested in robust, routine solutions

The model was implemented in two domains, which support different scenarios allowing us to illustrate and explore different properties of the model. The first domain allows us to implement each process in the model and quantitatively evaluate its performance in different scenarios. It also lets us compare the model to a GA operating without the benefit of knowledge biasing. The second domain allows us to qualitatively demonstrate that the model is capable of solving problems in very difficult domains with intractable search spaces. In these domains, solving problems using a GA without the benefit of knowledge biasing effectively reduces to simple exhaustive search.

4.3 Domain 1: Image Projection

This domain supports a range of problems that can be solved with relative ease by a regular GA with no knowledge base. Evaluating candidate solutions in this domain is relatively straightforward and fast, so generations of large populations can be developed quickly. The fitness function is also straightforward and allows easy progress to optima.

Problems in this domain involve arranging objects in 3D space in such a way that, when projected to a 2D image via a perspective camera, the resulting image matches a target image. Figure 4.2 shows objects arranged in 3D space, and Figure 4.3 shows the 2D projection of those same objects. A problem scenario therefore consists of a set of these objects to be arranged and a target image, and is considered solved if the projected image matches the target to within a certain tolerance.

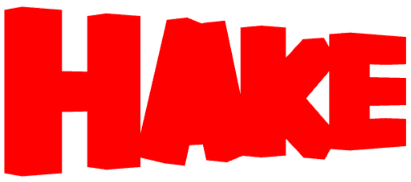


Figure 4.2: Example showing perspective in 3D domain

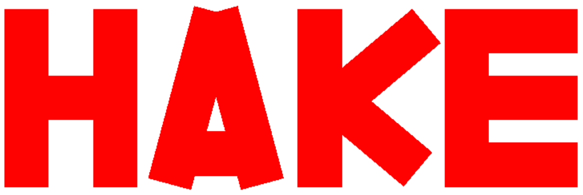


Figure 4.3: Example projected image in 2D domain

The image projection problems in this domain also allows for objects to perform different roles in robust ways. Simply orienting objects differently in relation to other objects allows them to effectively perform different functions, corresponding to different affordances. There is no need to encode tacit or adaptation knowledge in the model to enable different uses of the object. The problems in this domain can also be presented in such a way that we can use the 'world as its own model' approach discussed earlier: the image to be reproduced can be used directly to associatively select cases with images with matching features. Partial matches are also supported.

Note that features inspired by letters of the alphabet were used as they support the easy generation of cases using a ready-made collection of easily distinguishable and implemented features – the resulting 'words' and any similarities between them have no semantic significance. This implementation used Microsoft WPF (Windows Presentation Framework) 3D, and the open source AForge.NET library (http://www.aforgenet.com/framework/) for image processing.

4.3.1 Domain Implementation and Representation

Objects are all cuboids from a small set of 7 predefined types, with their positions and orientations specified by the following parameters:

- x-position *the position of the object along the x axis. Can be positive or negative*
- y-position *the position of the object along the y axis. Can be positive or negative*
- z-position *the position of the object along the z axis. Can be positive or negative*

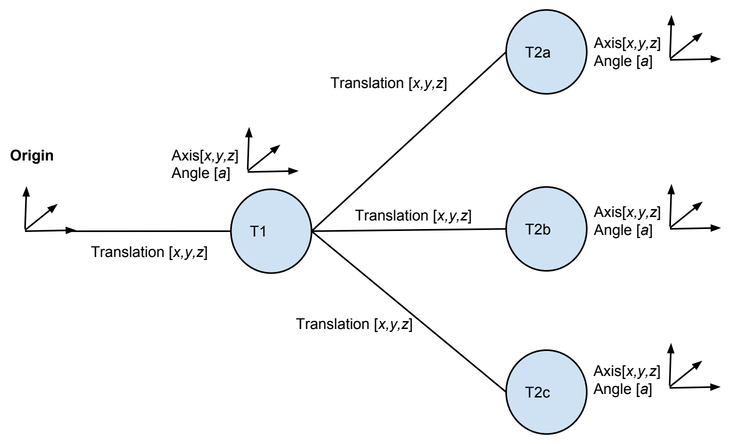
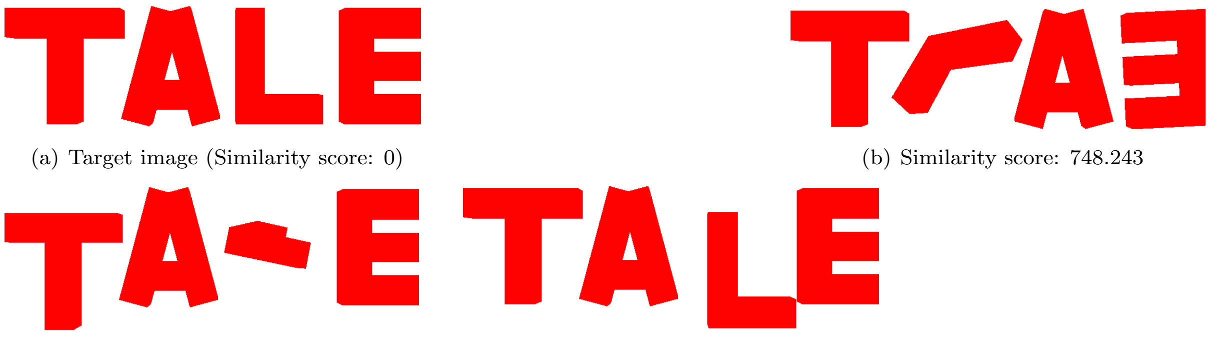


Figure 4.4: Assumption about the relative position of 3 objects in a 3D scene

- x-axis *the x component of the axis of rotation. Can be positive or negative*
- y-axis *the y component of the axis of rotation. Can be positive or negative*
- z-axis *the z component of the axis of rotation. Can be positive or negative*
- rotation-angle *the angle in degrees the object should be rotated around the axis of rotation. Can be positive or negative*

A solution consists of sets of these parameters together with the type of object to which they should be applied, such that every object in the problem scenario is accounted for. There is an optional reference object that the parameters are relative to – the origin of the 3D space is the default, but positions and orientations can be calculated with respect to another object. This allows assumptions to concern the relative positions of a set of objects, which is more flexible than assumptions about the absolute positions of individual objects. For example, an assumption might be that an object of type 1 should have a certain position and orientation relative to an object of type 2, effectively defining a composite object that can be operated on by the genetic algorithm while preserving the assumption. For example, for a problem for which the case shown in Figure 4.3 has been retrieved, one assumption might be that 3 elements of type *T2* are arranged relative to an element of type *T1* as described in Figure 4.4. This would correspond to the 'E' feature in Figure 4.3.

Solutions are themselves contained in cases, which consist of the solution as well as the image it produced. The image allows the associative comparison of the case with the current problem. The knowledge base consists of a set of these cases. Similar to CBR, there is no organisation or training of the case base – each new problem is assessed against it as it is encountered.



(a) Target image (Similarity score: 0)

(b) Similarity score: 748.243

(c) Similarity score: 445.305

(d) Similarity score: 53.901

Figure 4.5: Examples of images with similarity scores, where 4.5(a) is the target image

4.3.2 Selection

This section describes an implementation of the first process described in Section 4.2. A feature matching approach using the AForge.NET framework is used to select cases using a measure of similarity between the target image and solution images contained in the cases. The AForge BlobCounter class is used to identify sets of distinct features in both images. For each feature identified in the target image, we find the best matching feature from the case image and score it according to its similarity and relative position. Feature similarity is determined by combining a score for how much of the candidate feature falls outside the boundary of the target feature, and how much of a 'skeleton' of the target feature is contained within the the candidate feature. Each case image is given an overall score for similarity; if two images are identical the score is 0. Figure 4.5 shows some example image similarity scores.

All cases that score within an arbitrary threshold of 900 are added to the subset. A higher threshold value allows greater potential diversity of selected cases, which may in turn allow greater potential for non-routine solutions but at the expense of efficiency of routine solutions. Selected cases in the subset have their similarity scores normalised, giving weights that represent a confidence that they contain the most relevant knowledge for a conventional solution.

4.3.3 Knowledge Application

In this domain, an assumption from a case consists of a set of parameters for one or more of the 3D objects from the case problem scenario, describing the relative position and orientation of those objects in 3D space, as described in Section 4.3.1.

This assumption effectively allows such a set of objects to be treated as a single object with a single set of parameters on which the genetic algorithm is free to operate, while preserving the parameters for the individual objects. If the T1 element from Section 4.3.1 is rotated around the X axis, the 3 T2 elements are rotated with it to maintain their relative position.

In keeping with the model described in Section 4.2 assumptions are as conservative as possible, meaning that operations applied to the same type of objects wherever possible. If the operations in an assumption can be faithfully reproduced on available objects of the same type then that assumption is added to the lead and both the objects and the operations from the case become unavailable for new assumptions. Assumptions are repeatedly made until either:

- all the objects in the problem scenario are accounted for,
- no new conventional assumptions can be made from the subset of cases,

or

- all the operations in all the cases have already been used.

If there are still objects unaccounted for and no unused operations from cases, random assumptions are generated for them. These assumptions can refer to one or more objects. This process is shown in Figure 4.6.

4.3.4 Genetic Algorithm Parameters

The genetic algorithm operates on a meta-population of leads, each of which has its own population of candidate solutions that implement it. 10 leads are generated initially, and for each one a population of 500 candidate solutions is generated. A candidate is generated by altering each of the parameters in an assumption (as described in Section 4.3.3) to a random value with a probability of 0.8. 100 candidates are evaluated in turn for each lead, after which lead weights are replaced with their best candidate score. This is to allow each lead to be assigned a score representative of its actual performance, after which point roulette selection is used to choose a lead to provide a candidate for evaluation. Roulette selection is used as it is strongly biased towards better-performing leads (desirable for fast routine problem-solving) while still allowing weaker leads to occasionally be evaluated. This means more resources are allocated to the best performing solutions, which should correspond to the leads with the highest confidences for routine problems. Evaluation uses the same similarity metric described in Section 4.3.2. Parameters for the genetic algorithm were chosen on the basis of good performance in testing.

When all leads have had all their candidates evaluated, we keep the best performing top 10% or top 5, whichever is greater. For each of these we breed a new population of size 1000, using elitism by keeping the best 20. New candidates are created by crossover from two parents, selected using tournament selection with a tournament size of 10% of the population size to promote diversity within leads. Each candidate is represented as a list of assumptions. Since every candidate for a lead is based around the

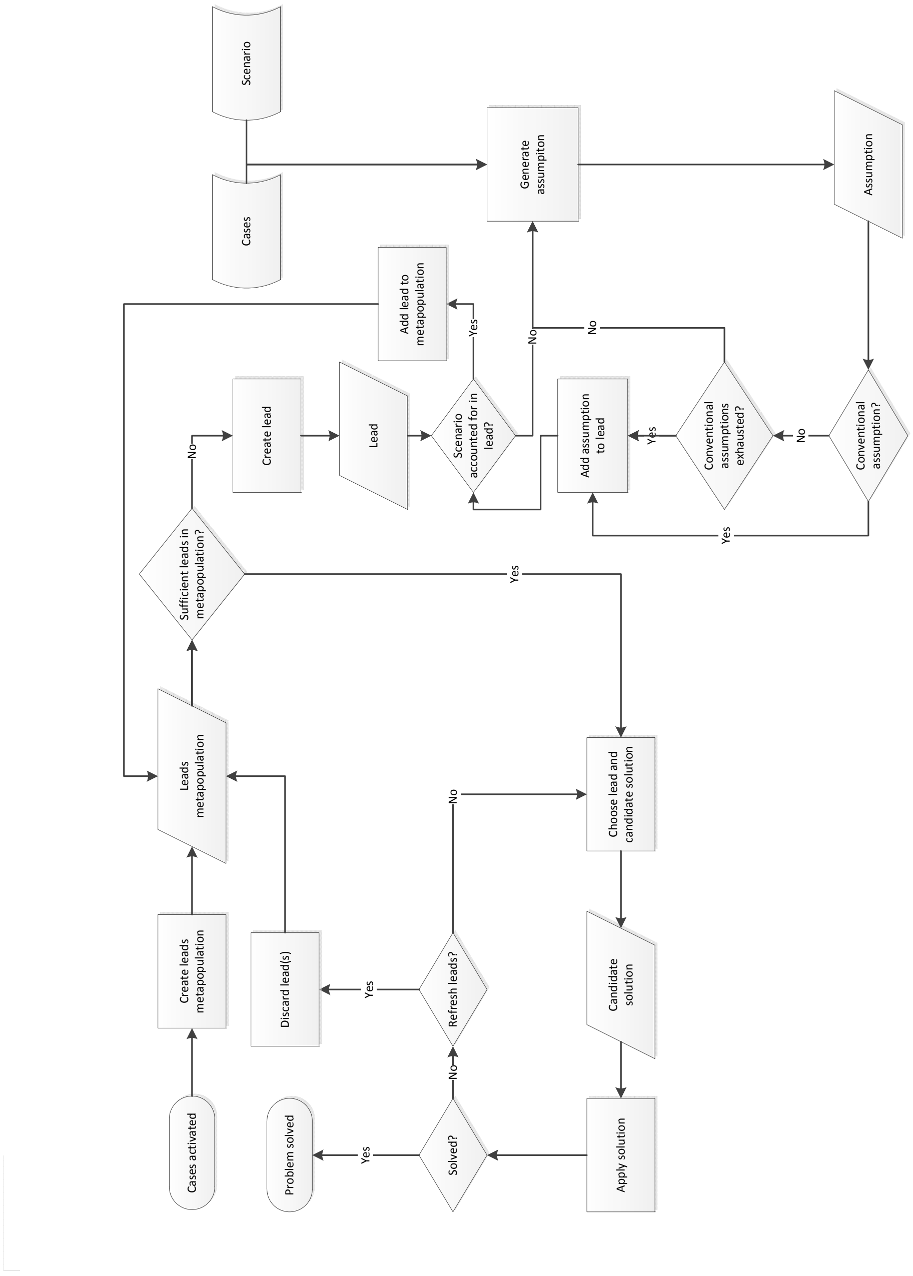


Figure 4.6: A flowchart showing how leads and assumptions are generated from cases.

same assumptions, lists have a fixed length. Single-point crossover occurs at a random point in the list, and both children are included in the new population.

50% of the candidates in the new population are then randomly selected for mutation. For each assumption in that candidate, parameters are replaced with random ones with a probability of 0.2. For efficiency we also check that new positions are in the field of view of the camera; if they are not, new parameters are generated.

When a new population is generated, the worst performing lead in the meta-population is discarded and replaced with a new one. This can be a brand new lead or a mutation of an existing one, selected by roulette. Existing leads are mutated by removing some assumptions with a probability of $1/n$, where n is the number of assumptions in the lead. Objects associated with those assumptions are now available for new assumptions, which can come from a new case-based assumption, an existing lead, or be randomly generated. The probability of random generation is 0.02; if this is not done the probability of a new case-based assumption or the transfer of an existing one is 0.5.

These new leads have new populations created as before. Once again, new leads have 100 automatic evaluations before reverting to roulette selection to select leads to provide candidates. This process continues until the problem is solved to within an acceptable limit.

4.4 Domain 2: Agents in Physics World

This domain is chosen to assess problems that are very difficult for standard GAs to solve. There is little opportunity for a gradient to be exploited – landscapes are largely flat with very steep cliffs to good solutions. In addition, evaluating candidates is very expensive and so only a limited number of evaluations are possible.

The domain is a simulated 3D physics-enabled world, in which problems consist of arranging a set of blocks to match a target state. Blocks are moved by autonomous agents situated in the world, and so solutions consist of behaviours for the agents that result in the desired outcome. Problems are considered solved if some calculation of the position of all the blocks is within a certain threshold. Problems are designed such that agents must independently perform actions in a particular sequence to arrive at a solution, with their behaviour being driven by their interactions with objects and other agents in the simulated world. Candidate solutions must therefore be allowed sufficient time for these interactions to play out, whereas in domain 1 a candidate can be evaluated in a fraction of a second.

Agents operating in the world have a limited amount of energy, which is depleted as they move around. When energy reaches zero an agent is 'dead' and takes no further part in the simulation (except possibly as an obstacle to other agents). Energy can be replenished by returning to a charger or consuming depots, which can be dropped by agents. To drop a depot an agent must first have visited a charger, and only one depot can be dropped per charger visit. Visiting the charger adds 55 to an agent's energy level, and dropped depots contain 65 units of energy. Therefore, a solution must feature behaviour to stop agents running out of energy as well as the necessary behaviour to move the blocks into the target state. This can involve cooperation between agents.

Figure 4.7 describes a foraging problem, in which 2 agents must collect a group of blocks from their original starting positions and deposit them at a different point some distance away. The agents start with energy levels of 35 and the distances between the charger, blocks, and target location are such that without replenishing their energy agents will 'die' well before the task is completed. Each candidate is allowed 25 seconds (running at 5x real time) to complete the task, although this will be cut short if all agents run out of energy. The problem is solved when the average distance of all blocks from the target location is less than 5.

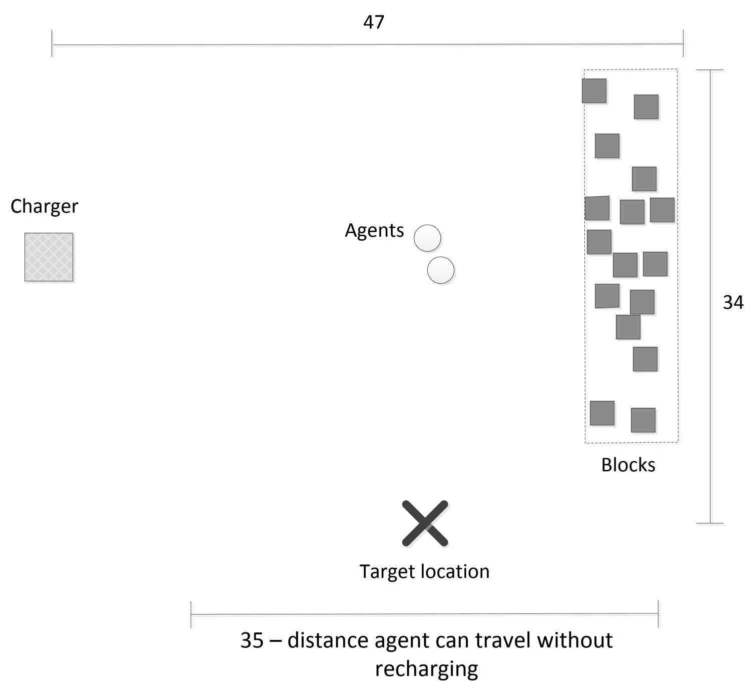
This domain is implemented using Microsoft Robotics Developer Studio (MRDS), which includes a simulation environment with a built-in physics engine.

4.4.1 Domain Implementation and Representation

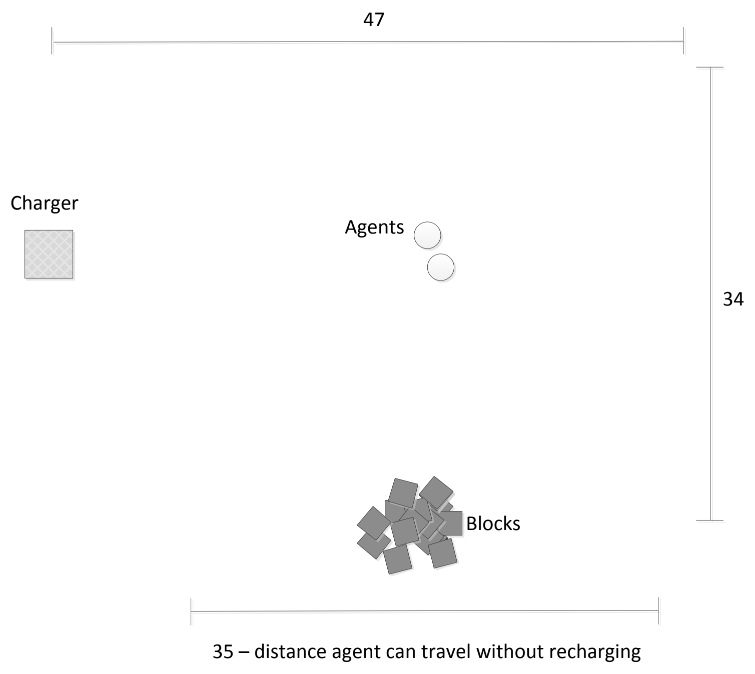
Each agent acts independently according to behaviours encoded in a tree, which consists of branches made up of *terminals* and *functions*. Terminals describe actions that can be performed, and functions evaluate the individual agent's state to determine whether a particular terminal action should or can be performed. Figure 4.8 shows an example behaviour tree.

The set of terminals used in this implementation is described further in Appendix A. There is a separate behaviour tree for each agent. A candidate solution to a problem, therefore, consists of a set of trees sufficient to account for the behaviour of every agent in the problem scenario. The tree representation allows modular snippets of behaviour to be swapped between genotypes using a genetic programming approach (Koza, 1992) which, apart from this difference in representation, can be considered to be equivalent to using a genetic algorithm.

The behaviour of each agent is determined by evaluating or 'walking' its tree, depth-first, every 140 milliseconds to determine the action the agent should perform, based on its state at that time. Agent state includes its current position relative to the scenario and the other objects in it as well as, for example, whether it is currently carrying an object or not. Agent actions can take the form of a 'doing' action (such as picking something up or dropping it) or a 'moving' action (such as moving to the nearest block). An agent can interact with an object when it is close enough that their bounding spheres intersect. For moving actions, the vectors specifying the agent's current position and the target location suggested by the action can be used to calculate a vector that is applied directly to the agent as a velocity to



(a) Start state



(b) Goal state

Figure 4.7: Start state and goal state for the foraging problem, showing the distances involved and the maximum distance the agents can travel without recharging. We see that the agents cannot complete the task without recharging.

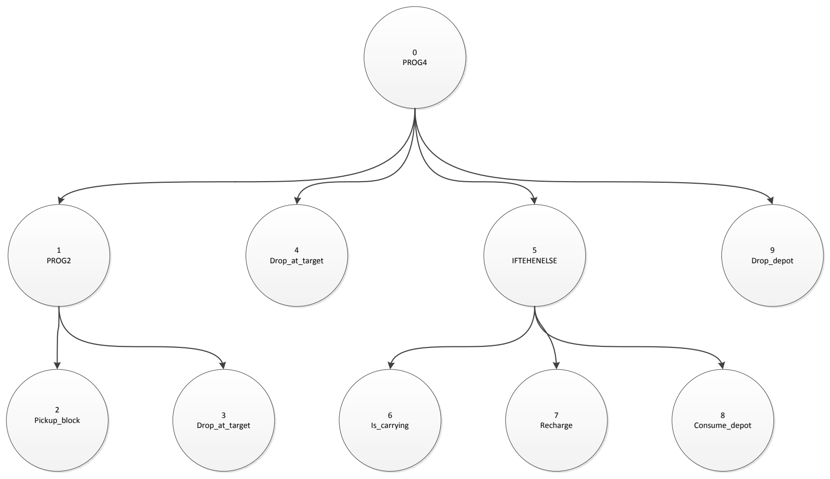


Figure 4.8: Example of an agent behaviour tree, showing the order of the depth-first evaluation of nodes

generate its speed and direction of travel. The length of this vector also represents the energy the agent is expending to move to the target location.

Cases contain a behaviour tree for each agent; all case scenarios feature the same set of agents. Cases also include the problem scenario as a scene consisting of a set of objects with dimensions and positions specified as vectors.

4.4.2 Knowledge Selection and Application

As in domain 1, we use the concept of leads to explore different sets of assumptions about the form solutions should take. In this domain, an assumption is a snippet of an agent behaviour tree taken from a solution in an activated case. As in domain 1, there is no *a priori* knowledge about which parts of the tree might prove to be useful in the new scenario. Snippets determine how an agent will behave in a particular set of circumstances, and can be combined into trees of arbitrary size that determine its overall behaviour. In this domain, then, a lead consists of a set of snippets, and a candidate solution for a lead is a tree incorporating those snippets.

As in domain 1, cases would be selected based on associative similarity of their problem scenarios to the current problem, with each case assigned a weight which is then used to probabilistically decide from which cases to take assumptions. Again, leads have weights reflecting the weights of the cases from which they have 'sampled' assumptions.

4.4.3 Genetic Algorithm Parameters

A population of size 250 is generated for each lead. For each evaluation cycle, a lead is chosen by roulette according to its weight or confidence score to provide a candidate for evaluation.

When all the candidates for a lead have been evaluated, a new population is bred. Crossover is used to generate new offspring with a probability of 0.9. Two parents are selected using tournament selection, with a tournament size of 10. Since we are using relatively small populations, there is a check to ensure that the parents are different. With a probability of 0.65 we check that at least one of the parents has displayed at least one of the desired behaviours for that lead, checked when that candidate was evaluated. Crossover is done by choosing a random node in the tree for each candidate to be combined and swapping the children of that node between the trees.

If crossover does not occur, a candidate is generated by mutating an existing candidate instead, chosen using tournament selection as above. Mutation is done by randomly generating a new tree and performing crossover with the chosen parent as described above.

If all leads have had at least 30 cycles or evaluations then, at every interval of 50 cycles, lead weights are recalculated, altering the probability of selection for evaluation. If the best-performing lead is still the one indicated by its confidence weights, then we decay the weights of each lead according to how many cycles it has had using the following formula:

$weight = weight \times e^{-1 \times (number-cycles/12500)}$

This is to ensure that the weights of leads that have had more cycles decay faster than those that have not, reflecting diminishing confidence in high-weighted leads as they fail to quickly arrive at a solution and reducing the selection bias shown towards them.

If, however, another lead has achieved the current best result, lead weights are recalculated according to the improvement they have shown. This is represented by the size of the step to their current best result. The aim of this is to favour leads that are actually performing well rather than those that were predicted to perform well.

4.5 Summary

We have described a model that incorporates some of the same characteristics we have identified in human problem-solving behaviour. These include the ability to solve routine problems efficiently through reluctance to consider more unusual solutions, and some capacity to solve non-routine problems *despite* this reluctance. The model operates by disregarding as much of the search space as possible in favour of a much smaller one constrained by the most routine knowledge it has available. This knowledge is activated in response to interactions with the problem scenario. We suggest that this is useful where there is no model of the domain, and might allow optimisation techniques to solve relatively easily solved problems (with small, cheaply-evaluated search spaces) more effectively, and potentially find solutions to some effectively unsolvable problems (those with very large search spaces, little or no gradient, and expensive fitness evaluations). The model can be thought of as conservative in that it attempts to use knowledge as conventionally as possible rather than exploring new ways of using it.

In this model, a routine solution means that the application of knowledge to parts of the scenario is routine, not necessarily that the outcome is quantitatively the same as the outcome from a previous case (although it is likely to be *qualitatively* similar). A problem is solved non-routinely if some other knowledge is used and applied differently than it is in the case that contains it. Knowledge is applied as strictly as possible in the model. Applying knowledge more loosely will be considered only if there is no knowledge that can be applied strictly, or if a comprehensive effort to use it has not solved the problem – that is, if a problem turns out to be non-routine.

We suggest that this approach is well suited to many tasks that we might want computers or robots to perform on our behalf, if the most routine solution is generally preferable but a computer or agent could benefit from some extra capability to generate non-routine ones. None of the components in the model have to be very intelligent, but the smarter the second process is the better the model might work, with the caveat that however smart it is it can only work in the conceptual spaces defined by the knowledge it has activated. Similarly, the richer the case base and mapping mechanism (i.e. the more flexible and robust) the better.

This model is consistent with the three ways of obtaining creative solutions described by Mednick (1962): serendipity, similarity and mediation. These describe, respectively, serendipity in encountering stimuli that activate elements that can lead to unusual (yet appropriate) associations, how those stimuli might activate particular elements, and how common elements can bring any associative elements into contiguity with each other.

Chapter 5

EVALUATION

In this chapter we describe some experiments to evaluate our model, using the two domains described in Chapter 4.

5.1 Objectives of Evaluation

5.1.1 Quantitative Comparison of Performance on Solvable Problems

Our first objective is to quantitatively compare the model's performance to that of a standard GA in different scenarios that reflect variations in the assumptions the model relies on. These assumptions are described in Chapter 4, and influence the extent to which the model can constrain the search space, and its advantage over a standard GA.

5.1.1.1 Experiment Design and Rationale

To compare the performance of the model in different scenarios with that of a regular GA, we must consider problems that allow us to perform multiple evaluations. This means that evaluating a candidate solution must be fast and the fitness landscape must be tractable. We use the domain described in Section 4.3, and we measure the performance of the model as we vary the 'quality' of the knowledge available in the case base in two ways: *amount* and *consistency.*

The *amount* of knowledge refers to the proportion of objects found in a problem scenario that are featured in cases. If problems contain some objects not featured in case knowledge the model will assign random values to those objects. The lower the amount of knowledge, the less the search space is constrained.

We measure and compare performance in four scenarios, each with a progressively smaller proportion of objects found in cases. In the final scenario no objects appear in cases, and the model behaves as a regular GA with all candidates initialised with random values and no biasing of the algorithm. This provides a benchmark for comparison.

Since evaluating a candidate is consistent in all scenarios, our metric for comparison is the number of candidate evaluations needed to arrive at a solution for each scenario. This evaluation of the effect of quantity of knowledge is described in Section 5.2.

Consistency of knowledge refers to the consensus in cases about how a particular type of object should be used; the effect that any uncertainty or a lack of consensus might have on the model is discussed in Section 4.2. To evaluate performance as knowledge consistency varies we consider two scenarios. In both scenarios, every object featured in a problem is featured in a case. In the first scenario, problem scenarios include both a set of unique objects and a set of objects that appear multiple times, with a case base in which unique objects are used consistently. In other words, in every case in which a unique object is featured it is used the same way. In the second scenario, problems involve only generic objects and a case base in which objects of that type are used inconsistently. In other words, in the second scenario there is no bias in cases to indicate a routine use of any of the objects in the problem scenario. To evaluate the effect of knowledge consistency on performance, we measure the rate at which the model is able to solve problems for each scenario. This is measured and plotted as the number of problems that can be solved and the number of cycles needed to reach those solutions. The evaluation of these scenarios is described in Section 5.3.

5.1.2 Qualitative Demonstration of Performance on Unsolvable problems

Our second objective is to demonstrate that, using our model, we are able to solve very difficult problems that a standard GA cannot. This is the case in domains where evaluation of each candidate solution is very expensive, and the fitness landscape is largely flat, with very steep cliffs to good solutions. This means that for a standard GA finding solutions effectively involves exhaustive search, and so these problems can only be solved using some additional mechanism, such as our model.

5.1.2.1 Experiment Design and Rationale

We use the domain described in Section 4.4. Since we cannot perform a large number of evaluations we cannot take a comparative, quantitative approach, so we aim instead to qualitatively describe how the model can find solutions to such problems.

Certain behaviours are necessary to solve the problem, for example picking up and dropping blocks and dropping and consuming depots. The second, lead-generating process operates independently of the first, case-providing process, allowing us to provide a mock-up of the first process in this domain. We

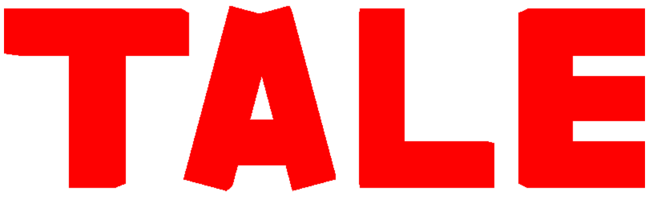


Figure 5.1: Target case

provide a set of trees which are known to generate these behaviours, and use them as cases from which the model can draw its assumptions. We then demonstrate that by using this knowledge to constrain the search space it considers, the model is capable of arriving at a solution. This evaluation is described in Section 5.4.

5.2 Experiment 1: Image Projection Domain

The ability of the model to solve problems routinely and robustly depends on the assumption that most problems encountered will involve objects that have been encountered before and which are used consistently. In this experiment, we compare the performance of the model in scenarios where not all objects have necessarily been encountered before, in which case applicable knowledge is not available in the case base.

5.2.1 Methodology

Cases are generated such that the assumptions for the model mentioned in Section 4.1 are upheld; specifically, different object types are always used the same way even when problem scenarios are different. The problem used to evaluate the model is a new combination of some of these features, shown in Figure 5.1. This arrangement includes:

4 x type1 dimensions (3,1,1)

2 x type2 dimensions (1,4,1)

1 x type3 dimensions (1,4,1)

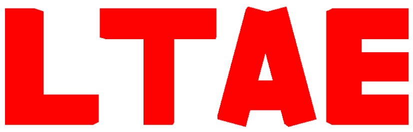
1 x type4 dimensions (2.5,1,1)

1 x type5 dimensions (1,4,1)

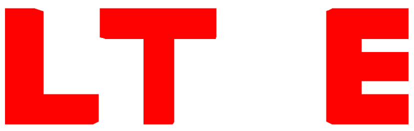
1 x type6 dimensions (1,2,1)

1 x type7 Composite: vertical element: (1,1,1) horizontal element: (4,1,1)

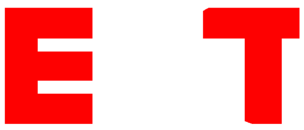
In the first scenario, the model is allowed to assess every case in a case base consisting of cases which all feature each of the objects available in the problem scenario. This means the model has access to routine knowledge that can be applied to each object in the problem scenario to reach a routine solution. Then,



(a) Example of a 100% complete case



(b) Example of a 75% complete case



(c) Example of a 50% complete case

Figure 5.2: Examples of cases of varying completeness

in subsequent scenarios, the model is restricted to a smaller set of cases that each contain knowledge concerning only some of the objects in the problem scenario – roughly 75% and 50%. Examples of a case from each scenario are shown in Figure 5.2. Assumptions are made from cases as described in Section 4.3.3 and consist of the arrangement of a set of objects: one object chosen at random from the scenario and any objects that intersect it.

In the last scenario, the model is run with no cases, behaving as a simple genetic algorithm with candidate solutions initialised with purely random values and no biases beyond the fitness function. Only one lead is used as there are no assumptions being made. Performance of the model is evaluated by measuring the number of candidate solution evaluations needed to reach a result that matches the target image with a similarity score of 50 or less.

5.2.2 Results and Conclusions

In Figure 5.3 we see that performance improves the more case knowledge is available. All scenarios where knowledge is available outperform the no-knowledge scenario. The rate of improvement in the 50% and 75% scenarios is less consistent than in the 100% scenario, and both converge to a much shallower approach to the solution (as is typically seen with genetic algorithms). However, the 75% scenario spends longer in this state than the 50% scenario despite an earlier advantage.

We conclude that any knowledge that can be used to bias the search space and constrain the genetic algorithm confers a significant advantage. By constructing solutions from the bottom up around assumptions, the model is able to robustly reuse knowledge from cases, even when there is no case containing an exact match. The process that generates solutions is able to do this without communicating with the process that selects the cases.

The similarity of the performance in the 50% and 75% scenarios can be explained by considering how assumptions are made. This influences the number of independent objects whose positions and ordinations must be optimised by the genetic algorithm, and consequently the size of the genotype and search space and the performance of the genetic algorithm. Table 5.1 shows the number of independent objects the genetic algorithm effectively has to work with in each scenario. In this experiment, we see that the 50% and 75% cases differ by only 1 independent object and so the difference between their respective search spaces is not as great as with the 100% scenario.

5.3 Experiment 2: Image Projection Domain

The ability of the model to solve problems routinely and robustly depends on the assumption that most problems encountered will involve objects that are consistently treated the same way. In this experiment,

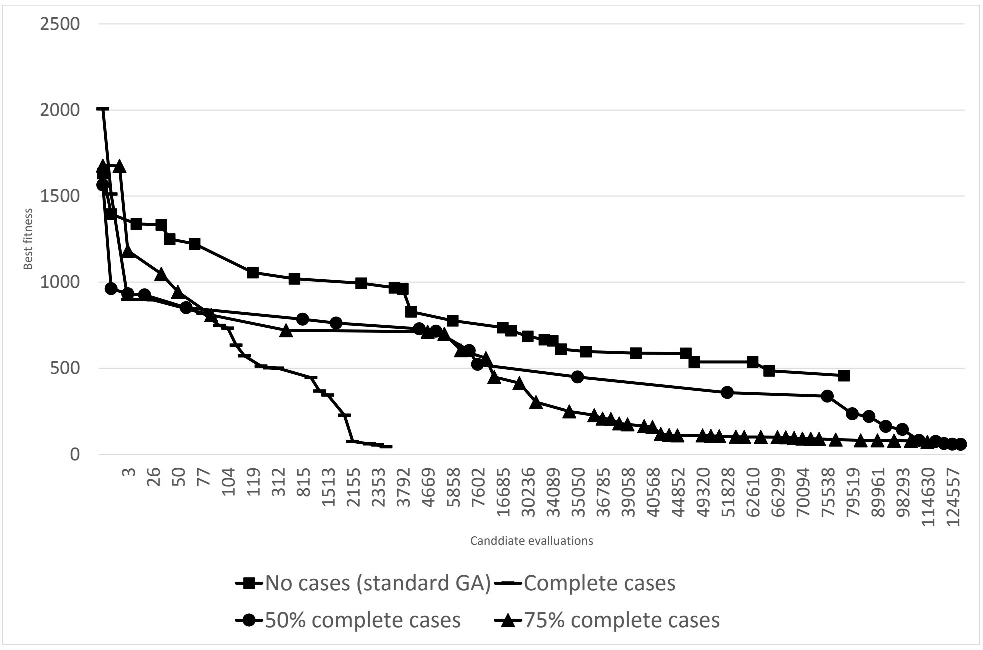


Figure 5.3: Graph showing the improvements in performance of a genetic algorithm by making assumptions based on cases selected for similarity with the problem

Table 5.1: Objects in problem scenario covered by knowledge

Scenario	Objects with knowledge	Independent objects
100%	11	4
75%	8	6
50%	6	7
0%	0	11

we compare the performance of the model in two scenarios: one where this assumption is justified, and one where it is not.

5.3.1 Methodology

We generate a case base of 100 cases in two scenarios. In the first scenario, the cases are based on random combinations of 7 'letter' features, as described in Section 4.3. Some examples are shown in Figure 5.4. Features are composed of objects from a set of 11 options. The features are defined in the same way across all cases. Unlike experiment 1, not all cases contain the same features, and the ability of the model to work with partial matches means that some selected cases might contain features not contained in the target image. In the second scenario, cases are built from a new library of 7 arrangements of objects, or features, which are again used in different random combinations. The two libraries differ in that features

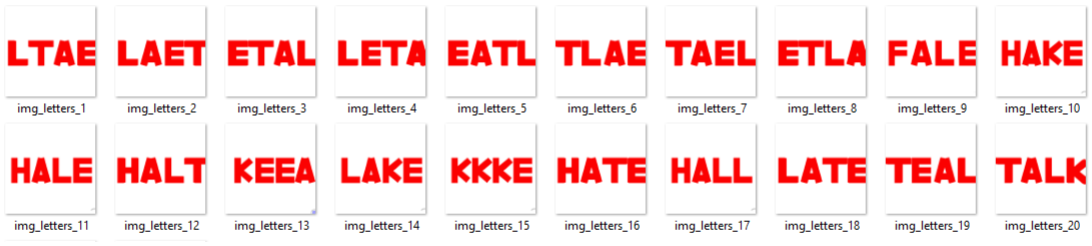


Figure 5.4: Randomly generated case examples.

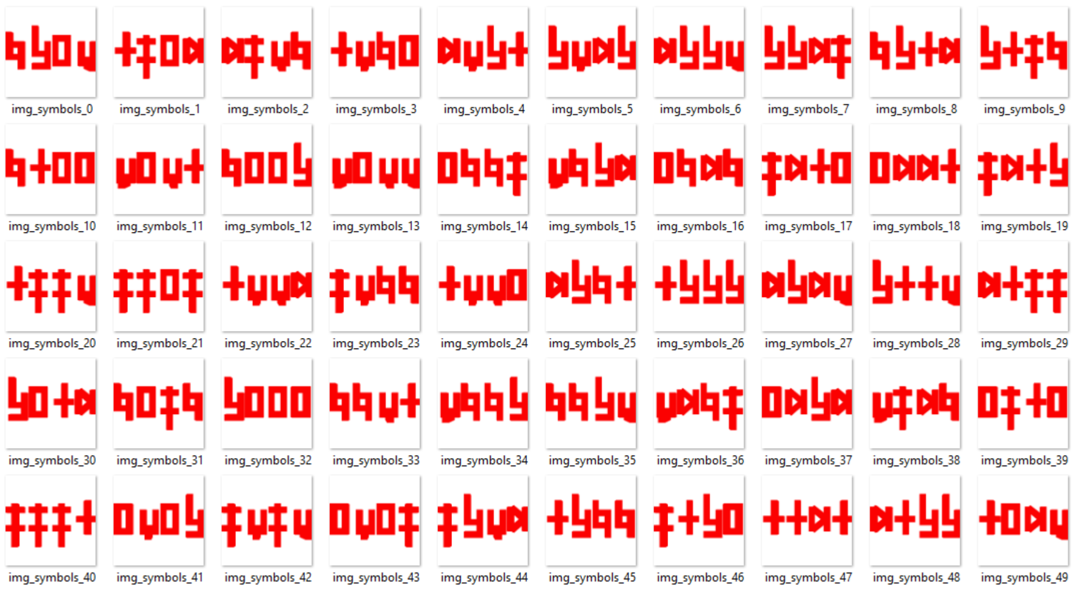


Figure 5.5: Randomly generated cases using generic objects of 2 types that can always be instantiated in a problem scenario.

in the first one do not all require the same number of objects, and each feature requires at least one specific object unique to that feature. By contrast, each feature in the second library requires 4 objects – 2 each of the same 2 types, and each case then consists of a total of 16 objects with 8 of each type. Some example cases from the second library are shown in Figure 5.5.

For the first case base, we selected 10 cases at random to serve as problems and recorded the number of cycles needed to arrive at a solution for each. These cases were excluded from case selection, meaning that no case exactly matched the problem. This process was repeated for the second case base.

5.3.2 Results and Conclusions

We see that performance is significantly better in scenario 1 than scenario 2. Figure 5.6 shows the average number of candidate evaluations needed to solve 10 problems in each scenario; the first scenario is 14 times faster. Figure 5.7 shows the average rate of improvement for each scenario over 66000 candidate evaluations. Initially they are very similar but the rate of improvement for the second scenario begins to decrease after a best result of around 400.

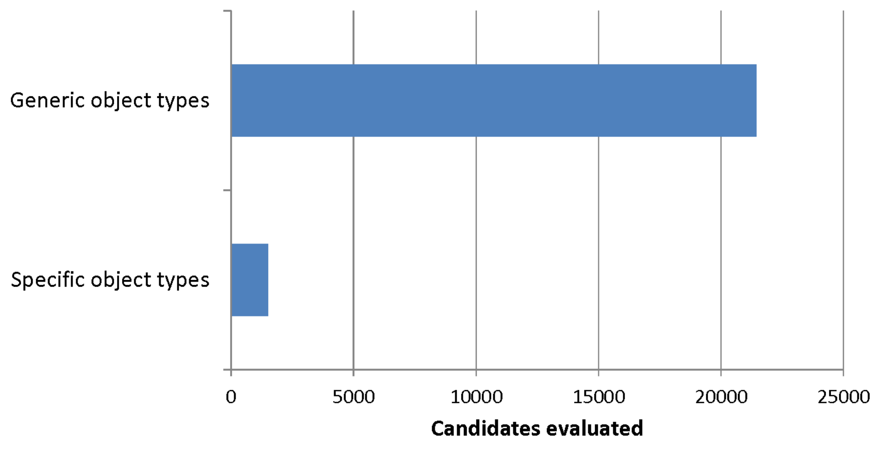


Figure 5.6: Number of candidate evaluations needed to solve problems involving specific and generic object types, averaged over 10 problems each.

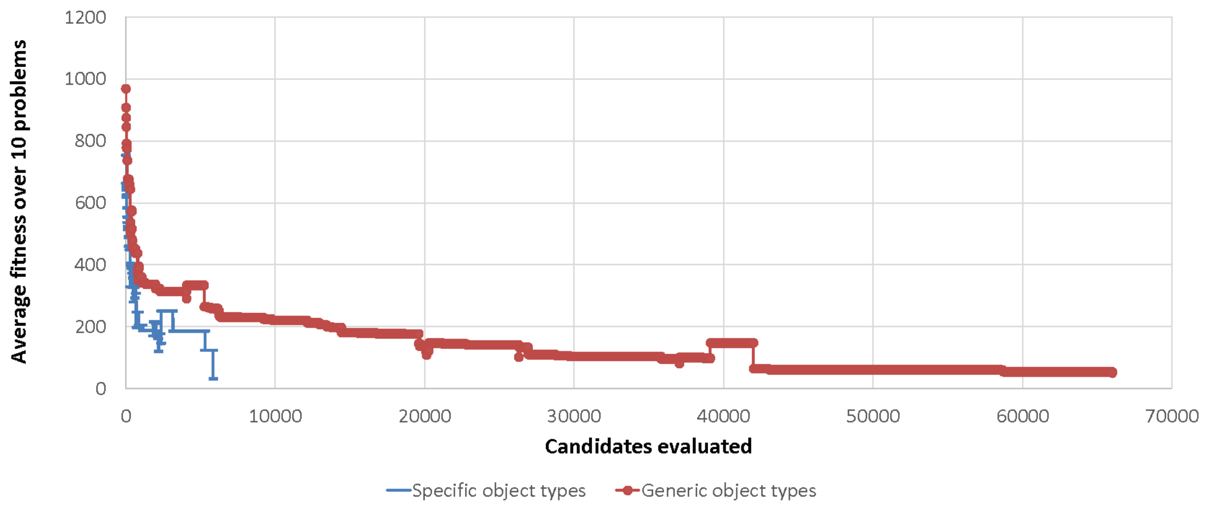


Figure 5.7: Rate of improvement in domain 1 with problems involving specific and generic object types, averaged over 10 problems each.

By building solutions from the bottom up around solutions, the model is able to flexibly and robustly reuse case knowledge to solve routine problems, even when cases can be retrieved using partial matches and consequently can contain superfluous knowledge that cannot be conventionally applied. It does this with no model of the domain or any *a priori* knowledge of what aspects of case knowledge is likely to be useful.

In scenario 1, problems involve some objects that are consistently used the same way in cases in which they are featured. In problems in scenario 2, no objects are used in only one way in cases in which they feature. This means that in scenario 2 the model can conventionally apply any feature from any selected case, regardless of which assumptions have already been made and which objects are consequently unavailable for subsequent assumptions. In scenario 1 it cannot, meaning that the search space covered by the leads is much more constrained, accounting for the improvement in performance. When this happens, the model is countering the lack of *a priori* knowledge about which knowledge from cases might be useful by exploiting the consistent knowledge associated with particular types of object. We conclude that the model works best with problems where at least some objects can serve as 'attractors' in the search space.

5.4 Experiment 3: Agents in Physics World Domain

5.4.1 Methodology

We assume that we have generated leads based on cases retrieved according to scenario similarity (skipping some of the steps in experiments 1 and 2). Each lead is designed to favour different types of behaviour applicable to the foraging problem, and is given a confidence indicating how many of those behaviours it constrains:

1. exchange-recharge-pickup-drop (confidence 0.6)
2. exchange-recharge (confidence 0.15)
3. pickup-drop (confidence 0.75)

The leads were designed to contain behaviours involved in one known viable solution to the problem. In this solution, one agent repeatedly visits the charger and drops depots at a midpoint. The second agent consumes the depots on its journeys back and forward moving the blocks and so never needs to visit the charger. The first lead is designed to be the most comprehensive and includes more of the behaviours needed to reach this solution.

Initial populations for each lead are generated by combining trees from a knowledge base that are known to show those behaviours when evaluated in the simulated domain. These trees were generated randomly and added to the knowledge base if they displayed any of the behaviours shown above when evaluated in the domain – they do not represent solutions to actual problems. This approach was taken because specifying the necessary functions and terminals to produce certain behaviours and actions does not necessarily result in those actions being performed, depending on the scenario.

5.4.2 Results and Conclusions

Two solutions to the problem were found, produced by two different leads as shown in Figure 5.8 and Figure 5.9. Figure 5.8 shows that the *exchange-recharge-pickup-drop* lead quickly shows promising results and begins to dominate the allocation of cycles, despite the *pickup-drop* lead having the higher initial weighting. However Figure 5.9 shows that the *exchange-recharge* lead arrives at a solution despite initially being allocated significantly fewer cycles than the more heavily-favoured options.

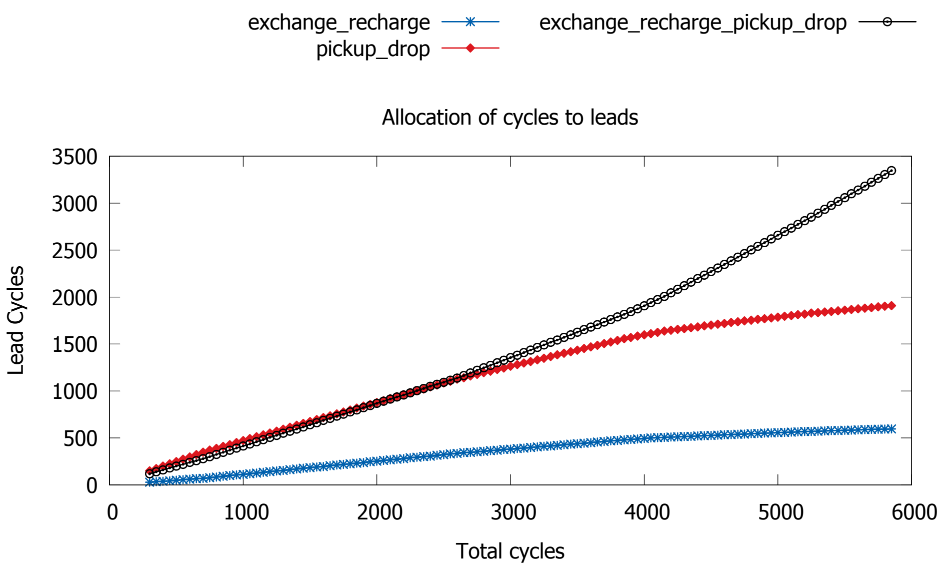
Limitations in this domain mean we can only draw limited conclusions. However, we see that biasing the search space with knowledge does allow the model to reach a solution. The affordances offered by problems in this domain are much less constrained than in the domain used in experiments 1 and 2, and the dynamic nature of the agents with their environment means that behaviours can emerge that, although they do not necessarily help to reach a solution, do not impede it either. Since the genotype can be of arbitrary length, behaviour displayed by candidate solutions can diverge from that specified in their leads without contradiction, which could not happen in the previous domain. This explains how the less favoured leads could beat the most favoured ones to finding a solution, and solutions can contain additional behaviour to that specified in leads.

5.5 Summary

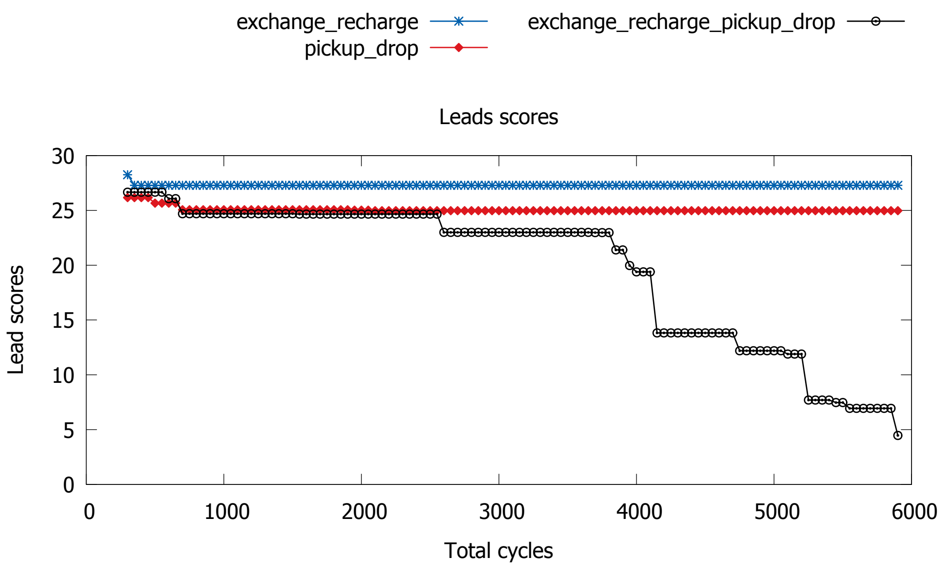
We have described some evaluations of the model and the rationale behind them. These evaluations show that the model is capable of effectively and robustly solving routine problems by generating solutions from the bottom up around assumptions from case knowledge, even with cases selected on the basis of partial matches and where no case exactly matches the problem. It can outperform a regular genetic algorithm that does not have the benefit of using knowledge to bias its search.

We have also described a qualitative evaluation which shows that the model is capable of solving problems that would be effectively impossible without the benefit of injected knowledge to bias the search.

We have also shown that the 'quality' of the knowledge available in cases affects performance. Good quality knowledge accounts (either directly or indirectly) for every object in the problem scenario, and has a clear bias towards a particular use of those objects. Shortcomings in either of these means performance

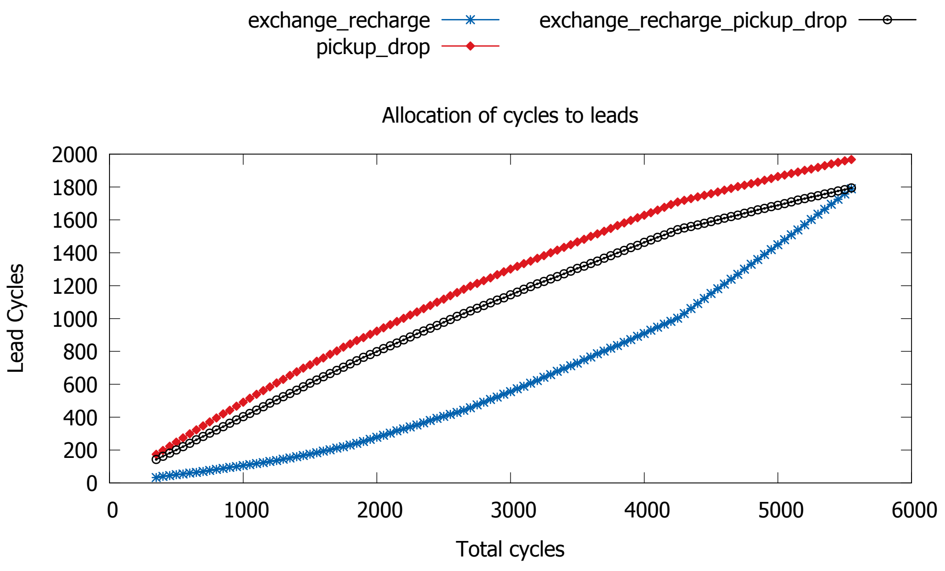


(a) Cycles

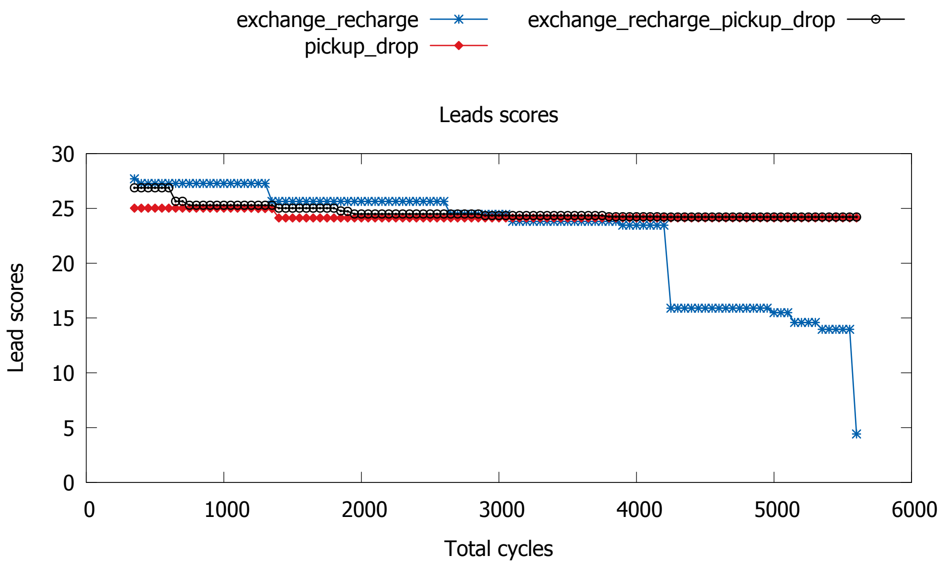


(b) Results

Figure 5.8: Scores and allocation of resources for solution from the *exchange-recharge-pickup-drop* lead



(a) Cycles



(b) Results

Figure 5.9: Scores and allocation of resources for solution from the *exchange-recharge* lead

deteriorates. As discussed in Chapter 4, the model assumes a context in which both of these can be relied upon to some extent.

This approach also provides opportunities for interesting things to arise, through two mechanisms: the operation of the GA (as shown in experiment 3), and the potential for the model to provide occasional opportunities to apply knowledge to a scenario other than that which would normally be associated with it.

Chapter 6

CONCLUSIONS AND FURTHER WORK

6.1 Evaluation of Research Objectives

The objective of the research, stated in Chapter 1, was to explore how the study of creativity and attempts to reproduce it in computers could help give computers problem-solving behaviour with some of the same characteristics displayed by people, namely robustness, flexibility, and the occasional appearance of surprising outputs.

This objective was achieved as follows. In Chapter 4 we developed a model of knowledge-driven problem solving which integrates recurring themes and concepts identified in previous work. The model was designed to display behaviour with some of the characteristics of human problem-solving behaviour, when working in domains and scenarios consistent with certain assumptions on which it is based.

In Chapter 5 we described some experiments undertaken to evaluate the performance of the model in scenarios consistent with those assumptions, and compared it to the performance of a regular genetic algorithm. We have implemented the model in two domains and shown that it is indeed capable of solving routine problems robustly and efficiently in such scenarios. We also explored the impact on the model of variations in the reliability of these core assumptions. The model offers useful improvements in performance in comparison to evolving solutions without knowledge. In the images domain used in experiments 1 and 2, we demonstrated that solutions can be found faster than they would be otherwise, to a degree that varies with the 'quality' of the knowledge available – that is, the amount and consistency of that knowledge. In experiment 3, we demonstrated that the model makes it possible to solve problems in a much more complex domain that would otherwise effectively be impossible.

6.2 Contribution

The contribution of the work is a novel model framework for robust problem solving. The model shows how the key themes and concepts identified in Chapters 2 and 3 appear as properties or artefacts of an integrated process for mapping case knowledge to problem scenarios. These are conceptual spaces, mental set, and exploratory, transformational and combinatorial creativity. Discussion of these concepts in the literature does not usually include the wider contexts or process in which they could occur, or their relationships to each other.

A novel aspect of the work is the consideration of these concepts, and the relationships between them, in the context of the wider problem-solving behaviour that we might want a computer or agent solving problems on our behalf to have. This behaviour is biased towards routine, familiar solutions reflecting the need to solve frequent routine problems, but any congruent ability to explore the non-routine application of knowledge despite this is valuable.

A second novel contribution is applying the concept of conceptual spaces to generate a meta-population of knowledge-based, or knowledge-constrained, genetic algorithms, and showing how these algorithms can robustly and flexibly recombine case knowledge.

6.2.1 Context of Previous Work

Our model is consistent with the recurring themes in the work discussed in Chapters 2 and 3, notably conceptual spaces, exploratory, transformational and combinatorial creativity, and the idea that the difference between routine and non-routine problem-solving is a matter of degree rather than type. A key concept of the model is leads, which are analogous to conceptual spaces. They represent the mapping of knowledge to scenarios and this mapping determines the surprising, or otherwise, properties of the solutions they can contain. This mapping to scenarios is consistent with the idea of affordances discussed in Chapter 2.

The model can be thought of as performing exploratory creativity within these spaces via the operation of the genetic algorithm in the second process and, as discussed in Chapter 2, this is not very powerful in itself. The potential for any surprising solutions to be found depends entirely on the space which, in turn, depends on the familiarity of the mapping from activated knowledge to problem scenario.

Transformational creativity allows things to exist that could not have existed before, via some some process that creates new spaces to be explored by transforming existing ones. In our model, new conceptual spaces are not created by transforming existing ones; they are built alongside them from the bottom up. We suggest that the ability to generate new conceptual spaces is more important than the exact mechanism by which this is achieved.

A conceptual space is defined by a set of mappings from knowledge to scenario. If surprising or unfamiliar mappings lead to a space that yields an appropriate, useful solution we suggest that this mapping would constitute combinatorial creativity.

In the model, surprising conceptual spaces would be generated by the same process as routine spaces; their surprising or otherwise properties are artefacts of the mapping between a given subset of knowledge and a given problem scenario. This provides a plausible mechanism for how non-routine and routine problem-solving behaviour can differ in degree or quality rather than type.

In operation, the model is consistent with Baars' global workspace theory and echoes Wiggins' 'loudness' from Section 3.5 in that there is competition for the opportunity to generate solutions between leads. This competition is also a characteristic of Copycat.

At a high level, the model can be thought of as a form of Case-Based Reasoning, using associative similarity to select cases. In contrast to CBR, cases are then broken down, combined and rebuilt from the bottom up, giving more robust problem solving.

Analogies are not made explicit and do not drive the generation of solutions; rather, implicit *a posteriori* analogies can be considered to be artefacts of the non-routine application of knowledge. This is consistent with Stojanov's suggestion that analogies based on surface or perceptual similarity can be more powerful because they are not based on existing conceptualisations. Finally, although the model is not intended to represent creative cognition, the rationale behind it is consistent with the criteria of production capability and evolutionary context suggested by Wiggins.

6.2.2 A Condition for Creativity

Considering the relationship between routine and non-routine problem-solving behaviour of an agent required to solve problems in much the same contexts as we do allows us to suggest another contribution, a *condition* for creativity:

> The ability of an agent to produce appropriate outputs through the non-routine mapping of a selected subset of existing knowledge to a problem scenario, despite a strong intrinsic tendency for the agent to favour routine mappings wherever possible.

In addition, we suggest that the model provides a plausible mechanism by which an agent in a population sharing similar knowledge and working in similar scenarios might generate a previously unseen conceptual space. This would be a space that others in the population could have generated themselves in different circumstances, meaning that they have a context in which to understand it and be surprised by it. This is another parallel with human creative behaviour.

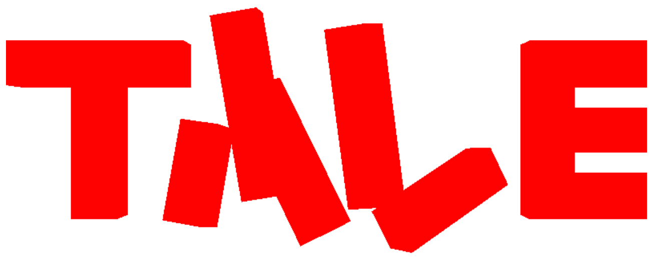


Figure 6.1: Half evolved solution

The scope for an unusual solution depends heavily on the domain. For example Figure 6.1 shows a part solution where an A and an L are being evolved from scratch. In this domain there are very limited arrangements of objects that can make an appropriate solution.

6.3 Further work

6.3.1 Improvements to Model

The model could be improved to include feedback as candidate solutions are evaluated. It could also be expanded to include more of the characteristics that inspired it, including more dynamic problem scenarios or environments and interactions with them, and an ability to keep unsolved problems in memory where they can be revisited when new potentially useful knowledge becomes available, mimicking our own ability to mull over problems. This ability would increase the opportunity for richer interactions between problem scenarios and the case base and so provide more support for non-routine solutions.

6.3.2 Assumptions and Problem Scenarios

The ratio or proportion of ‘attractor’ objects in problems, their dependencies with other objects in problem scenarios, and the way in which assumptions are made can all affect the ability of the model to constrain the search space. In Section 5.2 we saw how the search space and subsequent performance of the model could be influenced by the way in which assumptions were made in scenarios with incomplete knowledge. In the 75% scenario, for example, 3 objects were not accounted for in cases, but the impact on other assumptions was limited. If a different 3 objects were unaccounted for, the effect would have been greater; fewer assumptions could have been applied and the search space would be larger. Further experiments could be run investigating different scenarios.

In experiments 1 and 2, assumptions consist of the arrangement of a set of objects consisting of an object chosen at random from the scenario and any objects that intersect it. Assumptions start with an object chosen from a case, itself chosen from the set of activated cases, which is then mapped to the scenario. An alternative approach would be to choose an object from the scenario, search the set of activated cases for instances, and then weight the different ways in which it is used, choosing the most popular to map to the scenario. In experiments 1 and 2, these approaches would have yielded the same result, but the alternative approach could be more robust and generalisable.

6.3.2.1 Optimisation of Model Algorithms

There are several areas in which our implementation algorithm could be optimised. There is potential for smarter management of leads, for example by ensuring more comprehensive sampling of cases when generating them. Leads also currently remain largely independent, although there is a limited form of crossover in domain 1. A key element of the implementation is that if leads are not too similar there is an opportunity to learn from others that are doing well.

The method for comparing new problem scenarios could also be improved, perhaps with some sort of clustering or prototyping within the case base. Lead weights could also degrade according to their performance relative to their allocation of resources, meaning that if an initially well-performing lead fails to improve other leads are given more of a chance.

6.3.2.2 Dynamic Scenarios

The inspiration for the model is people operating in a dynamic environment, and the model and algorithms could be adapted to respond to changes in the world to drive the selection of new cases and new leads. If the dynamic environment does not exist as such (as in domain 1) candidate solutions could be fed back into the model to see if they lead to any different cases being activated. This would be more useful for non-routine scenarios.

6.3.2.3 Improvements to Evaluation

In domain 1 the current method of evaluating candidates is quite limited. Promising leads could score more highly if the similarity measure were smarter and could recognise that, say, a solution in which an upside down ‘E’ feature appears where an ‘E’ is needed is closer to the target than an otherwise identical candidate with an ‘A’ in the same position. This smarter similarity measure could also be used in the selection and assumption-making stages of the algorithm.

6.3.3 Comparison with other Methods

Our benchmark for evaluation was generating solutions from scratch using an evolutionary approach. There are of course other approaches to knowledge-based problem solving, for example attempting to adapt the most similar solutions in a more top down approach than our bottom up one. In our model, cases are effectively ‘mined’ for building blocks that are then used to construct a solution from the bottom up, with no reference or model for what the final structure should be. A top down approach would take a case solution as the starting point and try and adapt its structure. This approach would be adhering more closely the original CBR methodology.

6.3.4 Non-Routine Problem Solving

While we have demonstrated an improvement in robust, routine problem solving, the case for the appearance of non-routine artefacts is currently lacking. The model specifies that knowledge is always mapped to scenarios as conventionally as possible. If this does not solve the problem but an unconventional mapping (which would otherwise be discarded) does then we suggest the model would have displayed some behaviour analogous to combinatorial creativity. Further experiments could be performed to investigate these scenarios.

Appendix A

Terminals and Functions for Agent Behaviour Trees

A.1 Terminals and Functions

The set of terminals contains

> pickup-from, recharge, drop-at-point1, drop-at-point2, dropdepot, consume-depot, goto-position1, goto-position2

while the set of functions contains

> XOR, AND, OR, NOT, GT, LT, IFTHEN, PROG2, PROG3, PROG4, IFTHENELSE, IS, IS-CARRYING, HAS-RECHARGED

A.1.1 Terminals

recharge The agent recharges if its bounding sphere intersects that of the charger.

dropdepot The agent drops a depot at its current location.

consume-depot If an agents bounding sphere currently intersects with that of a depot the depot is consumed.

pickup-from The agent picks up the closest object in the original group of blocks.

drop-at-point1, drop-at-point2, goto-position1, goto-position2 Each scenario contains two sets of pre-defined points; these terminals let the agent drop what it is carrying or visit the appropriate points. In the problem defined in Experiment 3, one of these points is the charger (see Figure 4.7).

A.1.2 Functions

XOR If one of the first 2 child function nodes evaluate to true the third child node is evaluated.

AND If both the first 2 child function nodes evaluate to true the third child node is evaluated.

OR If either of the first 2 child function nodes evaluate to true the third child node is evaluated.

NOT If the first child function node evaluates to false the second child node is evaluated.

GT If the value of the first child is greater than that of the second child this node evaluates to true.

LT If the value of the first child is less than that of the second child this node evaluates to true.

IFTHEN If first child node evaluates to true the second child node is evaluated.

IFTHENELSE If first child node evaluates to true the second child node is evaluated, if not the third child node is evaluated.

IS If the boolean value of the first child equals the boolean value of the second child this node evaluates to true.

IS-CARRYING If the agent is carrying a block this node evaluates to true.

HAS-RECHARGED If the agent has visited the charger (and not dropped a depot since) this node evaluates to true.

PROG2, PROG3, PROG4 These functions simply evaluate all of their 2, 3, or 4 child terminals sequentially.

Bibliography

A. Aamodt and E. Plaza. Case-based reasoning – foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

M. M. al Rifaie and J. M. Bishop. Swarm intelligence and weak artificial creativity. In *AAAI Spring Symposium: Creativity and (Early) Cognitive Development*, pages 14–19, 2013.

A. Anwar and S. Franklin. Sparse distributed memory for 'conscious' software agents. *Cognitive Systems Research*, 4(4):339–354, 2003.

J. Austin. Distributed associative memories for high-speed symbolic reasoning. *Fuzzy Sets and Systems*, 82(2):223–233, 1996.

B. J. Baars. Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. *Progress in Brain Research*, 150:45–53, 2005.

P. Ball. Computer science: algorithmic rapture. *Nature*, 488(7412):458–458, 2012.

A. G. Baydin, R. L. de Mántaras, and S. Ontanón. Automated generation of cross-domain analogies via evolutionary computation. In *Proceedings of the Third International Conference on Computational Creativity*, pages 25–32, 2012.

R. A. Beghetto and J. C. Kaufman. Toward a broader conception of creativity: a case for "mini-c" creativity. *Psychology of Aesthetics, Creativity, and the Arts*, 1(2):73, 2007.

P. J. Bentley. Is evolution creative? In *Proceedings of the AISB'99 Symposium on Creative Evolutionary Systems (CES)*, pages 28–34, 1999.

R. Bergmann and W. Wilke. Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995.

R. Bergmann, J. Kolodner, and E. Plaza. Representation in case-based reasoning. *The Knowledge Engineering Review*, 20(03):209–213, 2005.

M. Bilalić, P. McLeod, and F. Gobet. Why good thoughts block better ones: the mechanism of the pernicious Einstellung (set) effect. *Cognition*, 108(3):652–661, 2008.

J. A. Biles. Composing with sequences: but is it art? In F. T. Howard, editor, *Applications of Fibonacci Numbers*, pages 61–73. Kluwer Academic Publishers, Dordrecht, 1999.

M. Boden. What is creativity. In M. Boden, editor, *Dimensions of Creativity*, chapter 4. MIT Press, Cambridge MA, 1994.

M. Boden. Creativity and artificial intelligence. *Artificial Intelligence*, 103(1-2):347–356, 1998.

M. A. Boden. *The Creative Mind.* George Weidenfeld and Nicolson Ltd, London, 1990.

P. Bonissone and W. Cheetham. Fuzzy case-based reasoning for decision making. In *The 10th IEEE International Conference on Fuzzy Systems*, volume 2, pages 995–998, 2001.

P. Bonissone and R. L. De Mantaras. F4. 3 Fuzzy case-based reasoning systems.

S. Bringsjord, P. Bello, and D. Ferrucci. Creativity, the Turing test, and the (better) Lovelace test. In J. H. Moor, editor, *The Turing Test*, pages 215–239. Springer, Netherlands, 2003.

W. Byrne, T. Schnier, and R. Hendley. Computational intelligence and case-based creativity in design. In *Proceedings of the International Joint Workshop on Computational Creativity*, pages 31–40, 2008.

J. G. Carbonell. Derivational analogy: a theory of reconstructive problem solving and expertise acquisition. In B. G. Buchanan and D. C. Wilkins, editors, *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, pages 727–738. Morgan Kauffmann, San Mateo CA, 1993.

P. Cariani. Emergence and artificial life. In *Proceedings of the Artificial Life Workshop*, page 775, 1990.

G. A. Carpenter and S. Grossberg. Adaptive resonance theory. *CAS/CNS Technical Report Series*, (008), 2010.

G. A. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART - fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4(6):759–771, 1991.

A. Carr. Organizational discourse as a creative space for play: the potential of postmodernist and surrealist forms of play. *Human Resource Development International*, 6(2):197–217, 2003.

D. J. Chalmers. Subsymbolic computation and the Chinese room. In J. Dinsmore, editor, *The Symbolic and Connectionist Paradigms: Closing the Gap*, pages 25–48. Pyschology Press, New York, 1992.

B. Chandrasekaran. Design problem solving: a task analysis. *AI Magazine*, 11(4):59, 1990.

J. Clement. Observed methods for generating analogies in scientific problem solving. *Cognitive Science*, 12(4):563–586, 1988.

N. Collins and J. d'Escriván. *The Cambridge Companion to Electronic Music.* Cambridge University Press, Cambridge, 2007.

S. Colton. *Automated theory formation in pure mathematics.* PhD thesis, University of Edinburgh, 2001.

S. Colton. Creativity versus the perception of creativity in computational systems. In *AAAI Spring Symposium: Creative Intelligent Systems*, pages 14–20, 2008.

S. Colton. The painting fool: stories from building an automated painter. In J. McCormack and M. d'Inverno, editors, *Computers and Creativity*, pages 3–38. Springer, Berlin, 2012.

S. Colton and G. Steel. Artificial intelligence and scientific creativity. *Artificial Intelligence and the Study of Behaviour Quarterly*, 102, 1999.

S. Colton, A. Pease, and J. Charnley. Computational creativity theory: the face and idea descriptive models. In *Proceedings of the Second International Conference on Computational Creativity*, pages 90–95, 2011.

D. Cope. *The Algorithmic Composer.* AR Editions, Inc., Madison WI, 2000.

F. J. Costello and M. T. Keane. Efficient creativity: constraint-guided conceptual combination. *Cognitive Science*, 24(2):299–349, 2000.

R. D. Coyne. Tools for exploring associative reasoning in design. In W. J. Mitchell, editor, *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era*, chapter 6. MIT Press, Cambridge MA, 1990.

D. H. Cropley. Creativity in engineering. In G. E. Corazza and S. Agnoli, editors, *Multidisciplinary Contributions to the Science of Creative Thinking*, pages 155–173. Springer, Singapore, 2016.

J. De Kleer and J. Brown. Theories of causal ordering. *Artificial Intelligence*, 29(1):33–61, 1986.

W. Duch. Intuition, insight, imagination and creativity. *Computational Intelligence Magazine, IEEE*, 2 (3):40–52, 2007.

B. Eno. Generative music. `http://www.inmotionmagazine.com/eno1.html`, 1996. Accessed: 2016-01-16.

B. Eno and P. Schmidt. *Oblique strategies.* Opal, London, 1978.

B. Falkenhainer, K. D. Forbus, and D. Gentner. The structure-mapping engine: algorithm and examples. *Artificial Intelligence*, 41(1):1–63, 1989.

G. Fauconnier and M. Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1972.

R. A. Finke. Imagery, creativity, and emergent structure. *Consciousness and Cognition*, 5(3):381–393, 1996.

S. T. Fiske and S. E. Taylor. *Social Cognition: From Brains To Culture.* Sage, London, 2013.

L. Gabora. The origin and evolution of culture and creativity. *Journal of Memetics: Evolutionary Models of Information Transmission*, 1(1):1–28, 1997.

L. Gabora. The beer can theory of creativity. In P. J. Bentley and D. W. Corne, editors, *Creative Evolutionary Systems*, pages 147–161. Morgan Kaufmann, San Francisco CA, 2002a.

L. Gabora. Cognitive mechanisms underlying the creative process. In *Proceedings of the 4th Conference on Creativity & Cognition*, pages 126–133, 2002b.

P. Gärdenfors. Representing actions and functional properties in conceptual spaces. In T. Ziemke, J. Zlatev, and R. M. Frank, editors, *Body, Language and Mind, Volume 1: Embodiment*, pages 167–195. Mouton de Gruyter, Berlin, 2007.

D. Gentner. Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

D. Gentner and A. B. Markman. Structure mapping in analogy and similarity. *American Psychologist*, 52(1):45, 1997.

J. Gero, V. Kazakov, J. Bardram, J. P. Hansen, P. Dourish, and V. Bellotti. Adapting evolutionary computing for exploration in creative designing. *Creativity Research Journal*, 17:241–255, 2005.

J. S. Gero. Design prototypes - a knowledge representation schema for design. *AI Magazine*, 11(4):26–36, 1990.

J. S. Gero. Computational models of creative design processes. In T. Dartnall, editor, *Artificial Intelligence and Creativity*, pages 269–281. Springer Netherlands, 1994.

J. S. Gero. Creativity, emergence and evolution in design. *Knowledge-Based Systems*, 9(7):435–448, 1996.

J. S. Gero. Conceptual designing as a sequence of situated acts. In I. Smith, editor, *Artificial Intelligence in Structural Engineering*, pages 165–177. Springer, Berlin, 1998.

J. S. Gero and U. Kannengiesser. The situated function-behaviour-structure framework. *Design Studies*, 25(4):373–391, 2004.

J. S. Gero and V. A. Kazakov. An exploration-based evolutionary model of a generative design process. *Computer-Aided Civil and Infrastructure Engineering*, 11(3):211–218, 1996.

J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston MA, 1979.

A. Goel. Design, analogy, and creativity. *IEEE Expert: Intelligent Systems and Their Applications*, 12 (3):62–70, 1997. 0885-9000.

D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3(2): 95–99, 1988.

K. Grace, J. S. Gero, and R. Saunders. Representational affordances and creativity in association-based systems. In *Proceedings of the Third International Conference on Computational Creativity*, pages 195–202, 2012.

S. Grossberg. Adaptive pattern-classification and universal recoding: II. feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 23(4):187–202, 1976.

J. Hawkins and S. Blakeslee. *On Intelligence*. Owl Books, New York, 2004.

S. Hélie and R. Sun. Incubation, insight, and creative problem solving: a unified theory and a connectionist model. *Psychological Review*, 117(3):994, 2010.

J. Hey, J. Linsey, A. M. Agogino, and K. L. Wood. Analogies and metaphors in creative design. *International Journal of Engineering Education*, 24(2):283, 2008.

D. Hofstadter, M. Mitchell, et al. The Copycat project: a model of mental fluidity and analogy-making. *Advances in Connectionist and Neural Computation Theory*, 2:31–112, 1994.

D. R. Hofstadter. Analogy as the core of cognition. In D. Gentner, K. J. Holyoak, and B. N. Kokinov, editors, *The Analogical Mind: Perspectives from Cognitive Science*, pages 499–538. MIT Press, Cambridge MA, 2001.

B. C. Jeng and T.-P. Liang. Fuzzy indexing and retrieval in case-based systems. *Expert Systems with Applications*, 8(1):135–142, 1995.

R. Johnsey. The design process – does it exist? *International Journal of Technology and Design Education*, 5(3):199–217, 1995.

P. Kanerva. *Sparse Distributed Memory.* Sparse Distributed Memory. MIT Press, Cambridge MA, 1988.

A. Koestler. *The Act of Creation.* Picador, London, 1964.

J. Kolodner. Improving human decision making through case-based decision aiding. *AI magazine*, 12(2): 52, 1991.

J. Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, 1992.

J. Kolodner. Understanding creativity: a case-based approach. In S. Wess, K.-D. Althoff, and M. M. Richter, editors, *Topics in Case-Based Reasoning*, pages 1–20. Springer, Berlin, 1994.

B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man and Cybernetics*, 18 (1):49–60, 1988.

T. Kötter and M. R. Berthold. (Missing) concept discovery in heterogeneous information networks. In M. R. Berthold, editor, *Bisociative Knowledge Discovery*, pages 230–245. Springer, Berlin, 2012.

J. R. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*, volume 1. MIT press, Cambridge MA, 1992.

A. E. Lawson. *The Neurological Basis of Learning, Development and Discovery: Implications for Science and Mathematics Instruction.* Springer, Netherlands, 2003.

D. B. Lenat. On automated scientific theory formation: a case study using the AM program. *Machine Intelligence*, 9:251–286, 1979.

D. B. Lenat, M. Prakash, and M. Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, 6(4):65, 1985.

B. Li, A. Zook, N. Davis, and M. O. Riedl. Goal-driven conceptual blending: a computational approach for creativity. In *Proceedings of the Third International Conference on Computational Creativity*, pages 9–16, 2012.

H. Liu and P. Singh. Conceptnet – a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

M. Maher and A. Gomez de Silva Garza. Case-based reasoning in design. *IEEE Intelligent Systems*, (2): 34–41, 1997.

M. L. Maher. Evaluating creativity in humans, computers, and collectively intelligent systems. In *Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design*, pages 22–28, 2010.

N. Maier. An aspect of human reasoning. *British Journal of Psychology. General Section*, 24(2):144–155, 1933.

C. Mair, M. Martincova, and M. Shepperd. A literature review of expert problem solving using analogy. In *Proceedings of the 13th International Conference on Evaluation and Assessment in Software Engineering*, pages 108–117, 2009.

J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

S. A. Mednick. The associative basis of the creative process. *Psychological Review*, 69(3):220–232, 1962.

J. M. Mendel. Uncertainty, fuzzy logic, and signal processing. *Signal Processing*, 80(6):913–933, 2000.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.

M. Minsky. A framework for representing knowledge. In P. H. Winston, editor, *The Pyschology of Computer Vision*. McGraw-Hill, 1975.

M. Minsky. Why people think computers can't. *AI Magazine*, 3(4):3, 1982.

K. Moorman and A. Ram. A model of creative understanding. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 74–79, 1994.

R. G. Morris, S. H. Burton, P. M. Bodily, and D. Ventura. Soup over bean of pure joy: culinary ruminations of an artificial chef. In *Proceedings of the Third International Conference on Computational Creativity*, pages 119–125, 2012.

D. P. O'Donoghue and M. T. Keane. A creative analogy machine: results and challenges. In *Proceedings of the Third International Conference on Computational Creativity*, pages 17–24, 2012.

B. Olshausen, C. Anderson, and D. Van Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 13:4700–4700, 1993.

A. Pease and S. Colton. On impact and evaluation in computational creativity: a discussion of the Turing test and an alternative proposal. In *Proceedings of the AISB symposium on AI and Philosophy*, 2011.

H. Poincaré. *The Foundations of Science.* Science Press, Lancaster PA, 1913.

C. S. Quintana, F. M. Arcas, D. A. Molina, J. D. F. Rodríguez, and F. J. Vico. Melomics: a case-study of AI in spain. *AI Magazine*, 34(3):99–103, 2013.

G. Ritchie. Assessing creativity. In *Proceedings of the AISB01 Symposium on AI and Creativity in Arts and Science*, 2001.

G. Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99, 2007.

M. Rosenman. The generation of form using an evolutionary approach. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 69–85. Springer, Berlin, 1997.

M. A. Runco. *Creativity: Theories and Themes: Research, Development, and Practice.* Elsevier, 2014.

R. Saunders and J. S. Gero. The importance of being emergent. In *Proceedings of Artificial Intelligence in Design*, 2000.

R. C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, Cambridge, 1983.

R. C. Schank and C. Cleary. Making machines creative. In S. Smith, T. B. Ward, and R. A. Finke, editors, *The Creative Cognition Approach*, pages 229–247. MIT Press, Cambridge MA, 1995.

R. C. Schank and A. Kass. Explanations, machine learning, and creativity. *Machine Learning: An Artificial Intelligence Approach*, 3:31–48, 1990.

R. C. Schank and D. B. Leake. Creativity and learning in a case-based explainer. *Artificial Intelligence*, 40(1):353–385, 1989.

R. C. Schank, T. Berman, and K. Macpherson. Learning by doing. *Instructional-Design Theories and Models*, 2:161–181, 1999.

T. Schnier. *Evolved representations and their use in computational creativity.* PhD thesis, University of Sydney, NSW, Australia, 1999.

J. Searle. Is the brain's mind a computer program? *Scientific American*, 262(1):26–31, 1990.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, 1948.

T. Shen and J.-C. Lai. Formation of creative thinking by analogical performance in creative works. *The European Journal of Social and Behavioural Sciences*, pages 1159–1167, 2014.

W.-M. Shen. Functional transformations in AI discovery systems. *Artificial Intelligence*, 41(3):257–272, 1990.

H. Simon. Problem forming, problem finding, and problem solving in design. *Design and Systems: General Applications of Methodology*, 3:245–257, 1995.

S. A. Sloman. The empirical case for two systems of reasoning. *Psychological Bulletin*, 119(1):3, 1996.

R. Smith, A. Dennis, and D. Ventura. Automatic composition from non-musical inspiration sources. In *Proceedings of the Third International Conference on Computational Creativity*, pages 160–164, 2012.

B. Smyth and M. Keane. Adaptation-guided retrieval: questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102(2):249–293, 1998.

B. Smyth and M. T. Keane. Experiments on adaptation-guided retrieval in case-based design. In *Proceedings of the First International Conference on Case-Based Reasoning Research and Development*, pages 313–324, 1995.

R. J. Sternberg. A three-facet model of creativity. In R. J. Sternberg, editor, *The Nature of Creativity*, pages 125–147. Cambridge University Press, Cambridge, 1988.

G. K. Stojanov and B. Indurkhya. Creativity and cognitive development: the role of perceptual similarity and analogy. In *2013 AAAI Spring Symposium Series*, pages 67–72, 2013.

I. Y. Subbotin and M. G. Voskoglou. Applications of fuzzy logic to case-based reasoning. *International Journal of Applications of Fuzzy Sets*, 1:7–18, 2011.

S. Thaler. The creativity machine: withstanding the argument from consciousness. *APA Newsletter on Philosophy and Computers*, 11(2), 2012.

T. Veale and Y. Hao. Comprehending and generating apt metaphors: a web-driven, case-based approach to figurative language. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, volume 2, pages 1471–1476, 2007.

G. Wallas. *The Art of Thought*. Jonathan Cape, London, 1926.

A. Webb. TRIZ: an inventive approach to invention. *Manufacturing Engineer*, 81(4):171–177, 2002.

R. Weber, K. Ashley, and S. Brüninghaus. Textual case-based reasoning. *The Knowledge Engineering Review*, 20(03):255–260, 2005.

A. A. Weir and A. Kacelnik. A New Caledonian crow (Corvus moneduloides) creatively re-designs tools by bending or unbending aluminium strips. *Animal Cognition*, 9(4):317–334, 2006.

R. W. Weisberg. Problem solving and creativity. In R. J. Sternberg, editor, *The Nature of Creativity*, pages 148–176. Cambridge University Press, Cambridge, 1988.

M. Welch. Analyzing the tacit strategies of novice designers. *Research in Science and Technological Education*, 17(1):19–34, 1999.

G. A. Wiggins. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems*, 19(7):449–458, 2006.

G. A. Wiggins. Crossing the theshold paradox: modelling creative cognition in the global workspace. In *Proceedings of the Third International Conference on Computational Creativity*, pages 180–187, 2012.

J. Wiley. Expertise as mental set: the effects of domain knowledge in creative problem solving. *Memory and Cognition*, 26:716–730, 1998.

W. Wilke and R. Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. In A. P. d. Pobil, J. Mira, and A. Moonis, editors, *Tasks and Methods in Applied Artificial Intelligence*, pages 497–506. Springer, Berlin, 1998.

M. Wilson. Six views of embodied cognition. *Psychonomic Bulletin & Review*, 9(4):625–636, 2002.

R. Withagen, H. J. de Poel, D. Araújo, and G.-J. Pepping. Affordances can invite behavior: reconsidering the relationship between affordances and agency. *New Ideas in Psychology*, 30(2):250–258, 2012.

R. P. y Pérez and M. Sharples. Three computer-based models of storytelling: BRUTUS, MINSTREL and MEXICA. *Knowledge-Based Systems*, 17(1):15–29, 2004.

L. A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3): 77–84, 1994.