Understanding complex systems through examples:

A framework for qualitative example finding.

Colin G. Johnson.

Computing Laboratory.

University of Kent at Canterbury.

Canterbury, Kent, CT2 7NX, England.

Telephone: 01227 827562

Fax: 01227 762811

Email: `C.G.Johnson@ukc.ac.uk`

April 2001.

Understanding complex systems through examples:

A framework for qualitative example finding.

Abstract

Many complex information systems in science, business and design have the characteristic that we can classify objects in the system in some way, but that these classifications are distributed through a parameter space in some complex fashion. In order for a human to get an understanding of the system, we would like to present this user with one example of an object for each class. Examples of such problems can be found in information retrieval, bioinformatics, computational geometry, computer-aided design, software testing and cellular automata. In this paper we will show how problems in all these areas can be put into a general framework of finding qualitative examples, and argue that general heuristic approaches to this type of problem are an important and neglected area of machine learning. We contrast this with some other well-studied problems, showing how this problem is distinct and investigating what we can learn from these problems. We then discuss some of the requirements for a heuristic to solve these problems, and mention some recent work on this using genetic algorithms.

Keywords: Heuristics, classification, novelty, diversity, information systems, machine learning.

INTRODUCTION.

The concept of *meta-heuristics* (Osman, 1996; Reeves, 1993; Corne et al., 1999) is a powerful idea in solving problems involving complex information systems. A meta-heuristic is a technique for finding approximate solutions to a problem, which can be applied to a large number of different domains of application.

The two canonical examples of the sort of problem a meta-heuristic can tackle are *optimization* and *search*. Examples of meta-heuristic methods for these types of problem are genetic algorithms, hill-climbing and simulated annealing. To solve a particular problem using such a technique we need to do two things. Firstly we need to show how the particular problem we have in hand can be phrased in terms of the method. So in genetic algorithms we need to provide a measure of solution quality ("fitness") and operators for crossover and mutation of solutions. In a hill-climbing method we also need to provide a quality measure, but we need a "move" operator which says how we choose the next solution in each round of the iteration. We can then apply the method, typically by running it on the computer. It is the first stage which distinguishes meta-heuristics from heuristic methods which have been created with a specific problem in mind. A meta-heuristic can be applied to many different specific problems, and it has the advantage (modulo the arguments in (Wolpert and Macready, 1995; Culberson, 1998; Tuson, 1999)) that an improvement to the meta-heuristic will redound to improvements across a range of problems.

This is a powerful methodology; nonetheless we appear to be stuck in a rut in which search and optimization are the only two areas for which meta-heuristics have been developed. In this paper I would like to outline a new area—*qualitative example-finding*—which appears ripe for the creation of meta-heuristic methods. In order to do this I explain what a qualitative example-finding problem is, show how a large number of real-world problems fit naturally into this framework, and then discuss strategies for creating meta-heuristics for this problem area.

Qualitative example-finding is an interesting machine learning problem which has not been investigated as a general problem. Traditional machine learning is the problem of abstracting a set of classification rules from a set of examples placed into classes. The qualitative example finding problem is, in some sense, the "opposite" problem. Given a procedure for classifying objects, can we find an example of an object for each class? This is an interesting problem with wide potential application, yet no general heuristics for this problem have been investigated. Perhaps this is because machine learning fits into a clear artificial intelligence tradition of attempting to reproduce human abilities, whereas qualitative example finding is a problem involving the use of computers to help human understanding, which is a completely different area of study.

More formally the qualitative example-finding problem is this. We are given a large set of objects, $\mathcal{O}$, and another (smaller) set $\mathcal{C}$, together with a classifying function $f : \mathcal{O} \to \mathcal{C}$. We assume that there is some underlying structure to the way in which these objects are classified, but that this structure is non-trivial and that we don't have any meta-knowledge such as the kinds of rules used to assign the classifications to the objects. Let us also say that calculating the classification assigned to a particular object is not computationally trivial, and that there is no way of doing a direct reverse calculation of a sample object given a particular class. This, combined with the large size of the set of objects, renders an enumerative search of the space $\mathcal{O}$ computationally out of the question. The problem we want to solve is to find an example of an object which fits into each classification, i.e. a set $\{o_1, o_2, \ldots, o_n\} \subset \mathcal{O}$ such that $\{f(o_1), f(o_2), \ldots, f(o_n)\} = \mathcal{C}$. Once we have found one example in a class we are no longer interested in finding other examples in that class, and there is no concept of one object being "better" or "fitter" than another in an absolute sense. Clearly the sort of situation we are considering is where the classification needs to be calculated, not just where all of the data is stored in a database.

Related problems require us to create the classification as we go along, based on some kind of metric that we impose on $\mathcal{O}$. Another variant is where we are able to classify things exactly, but we don't know how many members are in the classification set before we start.

QUALITATIVE EXAMPLE FINDING PROBLEMS.

In the previous section we have outlined an abstract problem area. In this section we will show how problems from a number of different areas can be placed into this framework. This demonstrates that effort spent on developing a general heuristic for problems of this type would be a valuable activity.

Creating test suites for agents and other software.

An *autonomous agent* (Huhns and Singh, 1998; Maes, 1994) is a piece of software which is supplied with various goals and a wide variety of possible simple behavioural patterns. The software is programmed at a high level to learn ways to combine those behaviours is such a way as to achieve the goals. Typically such an agent would work in a dynamic environment. However in advance of releasing it into that environment we would like to test it by presenting it with a range of qualitatively different scenarios, and checking that it can cope with each of those test scenarios. Simply creating these test scenarios by choosing environments at random is not guaranteed to produce a range of qualitatively different behaviours, and generating them by hand is likely to be time-consuming. If we can generate a set of qualitatively different test environments in an automatic way then we have a good basis for testing the agent.

A problem within a similar domain is explored in (Menczer et al., 1999).

This is one of very few papers to tackle a qualitative example finding problem. This paper is concerned with evolving a variety of distinct architectures for agents based around neural networks.

This system is based on the idea of *local selection* (Menczer and Belew, 1998). In this scheme each solution maintains an "energy" value as one of its attributes. This energy value begins at a certain value, and energy is gained by being in a good area of the search space, and energy is reduced if there are multiple agents attempting to explore the same region of the search space. Selection is *local* in the sense that there is a fixed energy threshold below which individuals are removed from the population, rather than solutions being globally compared to other solutions. Variations in selection pressure are maintained by a rule enforcing total energy conservation and a population size which is variable, where increase in population size occurs by good solutions reaching a "reproduction threshold" where they split and divide their energy between the two children.

Nonetheless there are limitations to the work described in (Menczer et al., 1999). Firstly they make no use of recombination, and it would be interesting to see if recombination could work to bring together structural features which lead to increased diversity. Secondly they are still largely working within an optimization framework, there is the concept of improving a solution that has been found, whereas in a lot of the problems that we are looking at there is no such concept. Nonetheless it remains the only attempt at explicitly searching for diversity using genetic-like methods rather than doing optimization.

Other related problems occur in generating a reasonable suite of test examples for testing a complex program. We can imagine two different kinds of processes here. The first just looks into the application domain, and the problem here is to create a tractable number of test examples which provide a wide range of qualitative behaviours found in the problem domain. The second would be to interact with a particular program that was to be tested. We generate a range of potential inputs, and monitor which parts of the program are being well tested by these inputs, then we use this data to find examples which test other parts of the program.

Software security.

Work by Forrest and others (Forrest et al., 1997; D'haeseleer et al., 1996; Dasgupta and Attoh-Okine, 1997) has made use of diversity for computer security. The core concept here is that one of the major security holes in computer systems is their similarity—the same software is run by many people, and so if someone can exploit a loophole in the way that software is written, they can breach the security of many systems. Also the potential cracker of the security of the system can use software identical to that of their intended victim in order to search for such loopholes. An example of their diversity-based method to defeat this kind of problem is that of *randomized compilation*—this creates many different forms of a program by treating the arbitrary decisions that compilers must make in a special way. Traditionally compilers have responded to the need to make arbitrary decisions by taking a standardised default value. In randomized compilation these arbitrary decisions are made at random, which means that many different compiled versions of the same functionally-equivalent program can be created.

Information retrieval.

Most problems in *information retrieval* have the flavour of optimization problems : we have a certain number of requirements and constraints and a large pool of data, and we want to find the examples of that data which satisfy these constraints and requirements in the best way. A canonical example of this kind of problem is *free-text* information retrieval (van Rijsbergen, 1979; Belew, 2000). Here we want to find the documents that most accurately match a number of query words, some of which may be marked as essential, or ranked in some way, or linked by boolean relations, et cetera.

One difficulty with information retrieval is that one query often maps onto a number of distinct qualitative areas of conceptual space, e.g. "java" maps onto a country, a programming language and a kind of coffee. Solutions to these difficulties sometimes require a more sophisticated knowledge of the query language than is possessed by the typical individual using the system, and sometimes they fox even the advanced user.

Thus we have the following problem. Given an information retrieval problem such as this, can we present the user not with the set of solutions which "best" match the criteria, but instead present the user with a set of solutions which "minimally" match the criteria, but which illustrate wildly different contexts for which the criteria hold. We can then have a second stage at which the user "contextualizes" the search by choosing one of those contexts, and the computer automatically creates a second search based on the data contained in documents which match well with that context.

Perhaps there are some useful ideas to be drawn from work on the "opposite" problem, i.e. finding information that is very similar (Dean and Henzinger, 1999; Broder et al., 1997).

There are some related questions concerning *data mining* (Piatetsky-Shapiro and Frawley, 1991), which is the attempt to discover interesting patterns in large databases. We can imagine using a qualitative example-finding technique to discover a range of qualitatively different patterns in a set of data.

Another related problem consists of exploring some kind of territory, whether real or virtual, in which a number of items of many different types are found. The prototypical example here is archaeological exploration, where we would like to explore a large area of ground and discover a diversity of objects that are used in that area; it is better to find one example of each kind of historical object than it is to find hundreds of examples of the same thing. Again we have a qualitative example finding problem; given each discovery we can measure some of its characteristics (ranging from physical characteristics to speculations about the role in the society that the object will have had, and we would like our heuristic to estimate where in the domain we should look for other objects which are different from the ones that we have found so far.

Knot classification and other mathematical problems.

Mathematical knot theory (Adams, 1994; Murasugi, 1996) is the study of the placement of loops in space. These three dimensional structures are commonly studied by means of *diagrams*, i.e. 4-valent plane graphs with under- and over- crossings marked. One important problem in knot theory is *knot classification*, i.e. finding a sample diagram for each distinct class of spatial structures. For a given number of crossings there are many two dimensional knot diagrams with that number of crossings (note that for every n-vertex 4-valent planar graph we can create 2^n knot diagrams by choosing crossings as over- or under- crossings). A similar calculation holds for braids—for a n-crossing m-string braid there are $n^m 2^n$ different braids. However these knots can be put into a much smaller number of categories based around the notion of two diagrams being "ambient isotopic", that is representing different views of a topologically identical three dimensional object. For example for 16 crossings there are around 10^{24} braids of 16 strings (which includes, if we remove trivial loops, all of the braids of fewer strings), but only 10468805 topologically distinct knots (Hoste et al., 1998).

This is an interesting piece of algebra which has applications in theoretical physics, for example the solutions of certain calculations in topological quantum field theories correspond to the different types of knots that can be found (Witten, 1989; Aneziris, 1994).

It is not trivial to calculate whether two knots belong to the same classification (Birman and Hirsch, 1998; Aneziris, 1994; Hoste et al., 1998). Nonetheless we can calculate *invariants*, which are functions have the property that if we calculate the same invariant for objects which belong in two different classes, then they must belong to two different classes (Jones, 1985; Kauffman, 1987; Kauffman, 1988). We can calculate a reasonable upper bound on the number of classes that there are, but there are no sharp bounds known for this.

There are some other mathematical problems with similar characteristics. For example in Gilbert Baumslag's computational group theory program *magnus* (Baumslag, 1993) there are routines which generate elements of a (possibly infinite) group. One danger with this kind of routine is that it can will "get stuck into a rut" producing examples which just reflect the underlying structure of the algorithm which is generating them, rather than presenting a diverse set of examples which could be studied further. A system which generated such a diverse set would be valuable, and the meaning of "diverse" could be steered by the user in an interactive way. Work on other parts of *Magnus* has shown that genetic algorithms can be used in this domain (Miasnikov, 1999; Baumslag et al., 1999).

Protein folding and drug discovery.

Increasingly research is finding out more and more about ways in which the sequence of a DNA molecule tells us about the geometric and reactive structure of the resultant protein molecule. Similar things exist in many other areas of chemistry. However this kind of problem is a computationally intensive task.

One thing that we would like to do, e.g. in drug discovery, is to take a particular kind of chemical structure, and discover the qualitatively different physical and reactive structures that different examples of this kind of chemical can take on. We can imagine a situation in which carrying out hundreds of experiments with real chemicals is costly and time-consuming, and so we would like to narrow down our search to just those experiments which are likely to produce qualitatively different behaviours. Clearly this is a very hard problem—at present we don't have a lot of detail about how the proteins fold, even—but it is an interesting potential application of these ideas.

We can imagine doing this in one of two ways. One idea would be to use an explicit folding model to discover the tertiary structure of the resultant chemical, and to search for as much diversity in this structure as possible. This would require an understanding of protein folding and related areas far in advance of current knowledge, though this is a rapidly advancing area of science. An alternative would be to take data from experiments and use that data to suggest which variants are likely to be different from the examples already tested. A similar approach (using inductive logic programming rather than GAs, and using explicit optimization criteria) as been investigated by King, Muggleton and colleagues (King et al., 1992; King et al., 1995).

Cellular automata.

Cellular automata (Wolfram, 1994) produce various behaviours depending on the parameters used to describe the interactions between cells and, to a lesser extent, the intimal configuration chosen. These different qualitative behaviours result in a complex way from the interactions between the cells. It would be an interesting problem to attempt to evolve a diversity of behaviour. This could be achieved by searching the space of different starting configurations for those that demonstrate these different behaviours, or by searching the space of rule sets in an attempt to find those rules which produce interestingly diverse qualitative behaviours.

Finding diverse behaviours in a simulated dynamical system.

Another area of interest is suggested by the work of Nowak and Sigmund on the iterated prisoner's dilemma (Nowak and Sigmund, 1992). In their experiments they find that the presence in the initial population of just one or two solutions from a particular part of the search space changes the long term dynamics of the game in a substantial way. In this case there are only a small number of qualitative dynamics producible within the system. However we can imagine other systems which have a wide variety of dynamics, e.g. in a simulation of the spread of a disease (Bailey, 1975; Anderson and May, 1991; DeAngelis and Gross, 1992) where we want to know what the range of results might be from various intervention strategies. These interventions do not necessarily produce a simple linear range of results—some might cause the disease to be eradicated in part of the population, so might cause a general decline in the level of the disease in the population, whilst others might cause an evolutionary change in the disease itself (Nesse and Williams, 1995; Ewald, 1994).

Similar problems occur in finding a good range of starting values for a genetic algorithm. It may be interesting to begin a genetic algorithm function optimizer with a step in which we calculate an initial population that samples the search space by finding a qualitatively diverse set of initial solutions, rather than selecting at random. A similar idea could be used for finding neighbourhoods to explore in tabu search. In tabu search we need to look around the neighbourhood for a next best solution, however sometimes the neighbourhood is too large to search exhaustively (Glover et al., 1993; Glover, 1990). Rather than selecting at random it might be good to run some quick procedure for finding a qualitatively diverse set of possible next steps.

Exploring the range of a sound synthesis algorithm.

In my earlier work on evolutionary interfaces for sound synthesis algorithms (Johnson, 1999), I developed a system which allowed users to rate sounds according to their level of interest in that sound or according to how closely the sound approximated some sound that they were searching for. The program would then present the user with the result of breeding the parameter strings which were used to generate those sounds, which would typically be closer to them than the non-chosen ones.

Now when playing with this program I found that there was another way of working, which was to set the mutation rate fairly high, then narrow in one one area of the sound-space, then move onto another area, and by doing so get a feel for the entirety of the sound capability of the synthesis algorithm being used. This again is an example of qualitative example-finding, though we don't know in advance what the categories will be.

So our problem is to write some kind of algorithm which allows us to input some kind of synthesis algorithm and which outputs an example of each of the qualitatively different sounds which that algorithm is capable of producing. There are two possible ways of doing this, the first would be to do some acoustical analysis on the phenotypes, and derive some measures from this. The alternative would be to do this through interaction with the user. Both of these are interesting, and it would be worth doing the two and attempting a comparison.

Evolutionary design.

Evolutionary methods have been used for a wide variety of design problems, as surveyed in (Bentley, 1999). Domains that have been explored by these methods include design of industrial processes (Goldberg, 1989; Parmee, 1996; Parmee, 1997; Parmee, 1998), design of mechanical linkages (Ékart, 2000), design of electronic circuitry (Miller et al., 1999), and architecture (Jackson, 1999; Rosenman, 2000).

These programs facilitate the exploration of design-space in various ways. Some of them are very traditional genetic algorithms, e.g. where there are large numbers of mechanical or physical constraints and the aim is to find a satisfactory design which satisfies all of these constraints within a certain tolerance. More interesting in the context of this project are those programs which facilitate *exploration* of a wide range of possible designs.

This idea is found particularly in the architectural work. The work of Jackson (1999) and Rosenmann (1999) is designed to allow architects to explore the different possibilities of creating some kind of structure by allowing them to explore the space of possible designs, combining interesting features from different designs. In their work this is guided by a human user, who rates the designs. It would be interesting to explore an alternative approach where the system generates a wide diversity of possible designs, rather than trying to "optimize" designs one at a time. Instead we take certain basic physical, functional and other constraints and create a wide sample of designs which satisfy that constraint, thus giving the designer an overview of the structure of the design-space.

RELATED APPROACHES.

A number of other problems have a some commonality with the qualitative example finding problem. In this section we shall outline a number of these problems, suggest ways in which they are distinct from qualitative example finding, and examine ways in which solutions to these problems might inspire heuristics for our problems.

Machine learning of classifications from examples.

A problem which has some of the same characteristics as the above problems is that of machine learning of classifications from examples. These kind of problems are particularly well studied in the neural networks community, using techniques such as *competitive learning* (Hertz et al., 1991; Bishop and Hinton, 1995). The idea here is to take a set of training examples each of which has been classified as belonging to one of a set of classes. The program learns some of the features of those examples, either by explicit symbolic learning or more typically by finding some subsymbolic "representation", and this learning is then applied to testing new examples which haven't been presented during the training stage.

A typical application of this is in recognizing people from images or biometric data such as gait or fingerprint (Jia and Nixon, 1995; Jain et al., 1999). In these circumstances there are a large (potentially infinite) set of images of each of a number of people, and we take a set of those to train our system. Then, when a new image is presented, e.g. by a camera taking a photo, the computer then tries to recognize the person. Similar ideas have been used in e.g. handwriting recognition.

As discussed above, this is distinct from our problem, in that we already know the classification procedure and we want to know the examples.

An important idea that we can take from this kind of work is the idea of processing information in a subsymbolic fashion (Brooks, 1991b; Brooks, 1991a). In traditional AI a major problem was that of finding representations for knowledge. Neural networks and similar systems have demonstrated that there is no need to directly represent each piece knowledge in a system is a discrete way. We can use this idea when it comes to producing heuristics for qualitative example finding. For example a useful concept in such heuristics might be the ability of a particular substructure of an object to have lots of distinctly classified objects built on it. A symbolic approach would be to identify these regions explicitly, but the subsymbolic paradigm shows that this explicit representation is not necessary.

Classifier systems.

A *classifier* is a rule of the form `if` *pattern* `then` *replacement pattern*. We can use sets of these rules as the beginning of a system that learns to find patterns in data, a so-called *classifier system*. Such a system consists of populations of these rules, together with a system for the apportionment of credit to the various rules that have found a successful example and a way of manipulating this population based on that apportionment, often by the use of a genetic algorithm. Details can be found in (Goldberg, 1989).

As with machine learning problems, classifier systems are concerned with the extraction of rules from examples, whereas we are concerned with the opposite problem. Again, the main lesson to take away from this problem area is the importance of subsymbolic approaches.

Computational models of creativity.

One of the perennial minor themes of artificial intelligence research over the years is how to make computers act in a creative manner. These have been pursued both with the motivation of understanding the nature of human creativity and with the motivation of producing creative work in a computer by means which don't correlate with ways in which humans are creative. These ideas are reviewed in (Boden, 1990; Partridge and Rowe, 1994).

Such studies date back to the early years of AI research. They can be split loosely into two different kinds of models—those which attempt to model the human aspect of creativity, and those that attempts to find an alternative computational model for creativity.

Some of these programs are designed to be creative within artistic domains, such as Cohen's drawing program *aaron* (Cohen, 1999), and programs designed to write stories (Meehan, 1977; Racter, 1984).

Other such programs work in scientific and mathematical domains, such as Lenat's controversial AM (Lenat and Brown, 1984; Ritchie and Hanna, 1984), work on conjecture-making in graph theory (Epstein, 1988), and various programs which make conjectures about chemical reactions (Langley et al., 1987). The important feature of these programs is that they are not focussed on solving specific problems, but they take a large database of information and attempt to induce conjectures from that information by a mixture of domain knowledge, ways of acting on this domain knowledge and "meta-knowledge" about what kinds of patterns are "interesting". So for example AM contains basic domain knowledge about set theory, ways of acting on this such as making conjectures, investigating the converse of known theorems, and so on, and meta-heuristics such as saying that some operator is interesting if it can be repeated an indefinite amount of times.

A third kind of "creative" system is that of creative analogy finding. This is typified by the work of Hofstadter, Mitchell and others on analogy making (Hofstadter and The Fluid Analogies Research Group, 1998; Mitchell, 1996). A typical experiment of this kind will involve a pattern problem like "if *aabc* becomes *aabd*, then what does *abcc* become?". The methodology here is similar to the more open ended systems such as AM, in that it works from a certain set of heuristics about how to manipulate these symbols, and a parallel set of heuristics about which kinds of patterns should be treated as most interesting.

In a similar vein is the work of Partridge and Rowe (1994) on creating systems which attempt to inductively learn the rules which are being used to create a pattern in a sequence of symbols presented to the computer. This attempts to go beyond simple unstructured sets of rules by allowing rules to be associated with subgoal structures known as "k-lines", which abstract groups of rules which work together well to achieve subgoals. These are then reused in solving other problem areas.

Finally a kind of creativity has been suggested for systems which are able to solve well-specified problems in interesting way, such as genetic programming (Koza, 1992). Thus we find papers which claim to use genetic methods as a "discovery engine" (Miller et al., 1999). This would seem to be a weaker form of creativity than the claims made above.

A major division here (see (Boden, 1990; Nelson, 1999) for more discussion) is between two types of creative behaviour. In the first kind of behaviour, there is a domain which can be described exactly, however the complex properties of the domain are not obvious from its description, and so creativity consists of finding "novel" things within this domain. The second type of creativity consists of "jumping out" of the present domain entirely, and creating a new domain in which to think. Whether these two kinds of creativity are really distinct is part of the ongoing philosophical debate on the nature of knowledge. It may be that for a sufficiently broad definition we can say that all "knowledge" *exists*, and the challenge is in searching in a sufficiently efficient way through the search space. Nonetheless there is a practical element to this when constructing artificial intelligence systems, as representing a limited knowledge domain is an standard task on a computer, so the distinction here is probably clearer than when we are considering human creativity.

There is a lot of common ground between the kind of work that we are doing and these studies in computational creativity. However the work in AI on creativity is focussed largely on understanding the creative faculty in human cognition, whereas we are interested in the discovery of novelty using any method, regardless of its connection to human creativity. Another contrast is that we are interested in finding a range of qualitative examples rather than finding particularly "interesting" ones. In many ways the contrast between this work and the problems described here is that in the studies of creativity we are interested in what makes a particular solution "interesting", whereas in our problems we are concentrating more on what makes a solution "different". For example in a program which creates interesting music, we are interested not in exploring the entirety of musical space within a single song, but on narrowing in on a small area of that space.

Furthermore we are exploring a predefined search-space in carrying out this kind of analysis. Our interest is in how we can explore this search-space in an efficient way, rather than looking into how we can escape from the search-space.

Nonetheless these studies of creativity have somewhat of a similar feel to what we are trying to achieve in the problems described here, and there are a number of techniques drawn from these studies which could prove useful.

One idea that is particularly interesting is the idea of "search heuristics". In programs such as AM, one of the big ideas is that of the program applying a range of heuristic techniques to suggest how one established idea can be turned into another. In such programs the heuristics transform one solution into another, for example using ideas such as "create the converse of this statement" and "swapping an *and* for an *or*" (Boden, 1990). This has similarities to a symbolic version of the mutation operator. It would be interesting to explore other kinds of operator designed to work in a similar way to the recombination operator, i.e. providing standard heuristics for combining two or more concepts. There may be some ideas for such heuristics in work such as that of Polya (1945) and De Bono (1990) which attempts to unpack the heuristics used by humans in creative problem solving. This provides an interestingly contrasted view on what genetic algorithms are doing.

Another interesting idea which can be drawn from a lot of studies of creativity (see e.g. (Boden, 1990)) is that it is not just sufficient for a system to do creative things, but that a system must recognize when it is being creative. There may be some way of casting this into a genetic algorithms framework by a dynamically created fitness function which measures the novelty of a system, remembering that novelty in our problems is a well-defined property, in contrast to the problems discussed by the computational creativity researchers.

Cluster analysis and related approaches.

Another idea which has some superficial similarities with what we are trying to achieve is that of *cluster analysis*, which is a statistical technique which takes multivariate data and groups it into a number of clusters based around metrics in the space of solutions (Krzanowski, 1990).

There are other problems which are in the domain of "grouping". A well known NP-hard problem is the bin packing problem, which consists of filling a number of containers of a given capacity with a number of objects of given size. A similar problem is the *set partitioning problem* where we want to split down a set into a number of categories, assigning each member of the set to one of those categories in such a way that some scoring function is maximized. This is use e.g. in transport scheduling, where a number of aircraft have to be assigned to a number of limited-capacity routes in a way that minimizes distance travelled. Both of these problems have been tackled using genetic algorithms (Falkenauer, 1998; Levine, 1994).

In the cluster analysis problems we are working from a large body of data about the problem at hand. We are interested instead in the kind of problem which is too large for traditional statistical analysis, and where we instead need to provide a strategy for deciding which data to compute in the first place.

The grouping problems are different in a distinct way. These problems are essentially optimization problems, rather than example-finding problems. Furthermore the individual solution string in one of these problems represents a whole solution to the problem (i.e. it is an example of a *Pitt approach*, to use a well-known terminology from machine learning/classifier systems), whereas we are interested in problems requiring a *Michigan approach*, where the individual chromosomes each represent a single component of the solution.

It is difficult to see how these ideas can be adapted easily for this kind of research. In these kind of problems we need a comprehensive database view of the population in order to calculate statistical measures on the population, and that is exactly what we don't have in our kind of problem—the phenotype is calculated in a nontrivial way from its representation, and there is a very large population. Also we know the categories into which things fall in many of our problems, whereas in the problems described in this section one of the main aims of the technique is to discover the natural categories for things from metrics on pairs of individual elements in the space.

Multimodal optimization.

A class of problems which have much in common with the kinds of problems that we are discussing above are *multimodal optimization* problems. A multimodal optimization problem consists of some fitness function which has a number of local optima, and the aim of the problem is to find all of these optima, or to find all optima which are above some threshold value, or to find a certain number of local optima.

Multimodal optimization problems have been successfully tackled using genetic algorithms. Many of these approaches are based on the idea of *niching* (Mafoud, 1997), that is modifying the way fitness is distributed so that solutions are rewarded both for being fit relative to the problem being solved, and also for being distinct from other solutions—occupying a particular niche in the fitness landscape.

The first approach to this is via *fitness sharing* (Goldberg and Richardson, 1987). Firstly the fitness of individuals is calculated in a standard way. Then this fitness of each individual is divided by (some function of) the number of individuals which are "similar" to that individual. This similarity can be defined by some metric on the space of genotypes (Mafoud, 1997), such as hamming distance, or by the use of an application-domain specific metric on the phenotype. This can be calculated either directly, by comparing each individual with the others in a structured way (Goldberg and Richardson, 1987; Deb and Goldberg, 1989), or a statistical technique such as cluster analysis can be applied to break the population down into similar groupings (Yin and Germay, 1993), or by a prior estimation of the number of niches in the population (Miller and Shaw, 1995).

Crowding is an alternative approach which is also based on the idea of niche formation (Grüninger and Wallace, 1996; Mafoud, 1992). In a crowding system an alternative form of selection is used which is in two stages. Firstly a large number of recombinative pairs are generated at random from the population, without regard to fitness, and these pairs are then subjected to a form of selection where each of the children competes against one of its parents in order to enter into the new population. Decoupling of recombination and selection ensures that selection happens locally, whilst recombination allows global information exchange. An alternative is to choose certain individuals as dominant individuals and, each generation, clear out all solutions that don't fall within a certain (phenotypic) radius of that individual according to some domain-specific metric (Pétrowski, 1996).

An alternative approach to these kind of problems is via an explicit speciation approach which breaks the population down into a number of well-defined subpopulations (Deb and Spears, 1997). This can be carried out by a number of methods. The first is to attach "tag bits" to members of the population, which say which other individuals they are allowed to breed with. An alternative is "island models" (Tanese, 1987; Tanese, 1989; Calégari et al., 1997), where the population is split into a number of subpopulations, and most breeding goes on within those subpopulations, with only an occasional good solution able to travel between islands. This simulates the conditions in which speciation occurs in natural populations.

Despite the superficial commonalities between multimodal optimization and qualitative example finding, there is actually a wide gulf between the two problems. In the multimodal function optimization problems the emphasis is still on *optimization*, i.e. the search space still has an exogenous fitness function. This contrasts with our problems where we are working in a landscape where there are only qualitative categories and not any absolute measure of fitness. We want to create a system where finding diverse solutions is the direct goal, not just a byproduct of an exogenous-fitness based optimization procedure.

Also a lot of the work here goes on *within* the subpopulations, e.g. breeding within the population to find a better example of that local peak in the fitness landscape. In our problems we are concerned only with finding an example in each class, once we have found one example we want to move away from any further consideration of items in that class.

Nonetheless it may be possible to use the idea of “niching” in a different way. A qualitative example finding heuristic could create (temporary) subpopulations which explore a particular area of the space which is proving to be a particularly fecund source of examples, that subpopulation dissolving away as that area of the space becomes “mined out”.

An area often confused with multimodal optimization is *multiobjective optimization*. Whereas multimodal optimization is concerned with finding multiple solutions to a population with a single solution, multiobjective optimization is concerned with finding a single good solution to a problem on a space where there are several fitness functions which need to be combined in some way.

Simulating biological diversity.

A number of computational studies have investigated the process whereby diversity evolves in natural populations. This has drawn both on ideas from theoretical evolutionary biology and ecology, and on the techniques of “artificial life” which provide ways of creating models of populations based around simulating the behaviour and evolution of individuals.

A good example of this is the work of Maley (1998,1999). In (Maley, 1999) three models are presented which aim to investigate what characteristics a model must have in order to satisfy a set of conditions for "open ended" evolution. One of the conditions is that

> *'An open-ended evolutionary system must exhibit continuing ("positive") new adaptive activity.'*

The model used to investigate this is based around a grid, each of the squares of which represents a distinct niche in the environment which can be occupied by members of the population. It is shown that pure neutral evolution rapidly fills up a large number of niches. However when selection is included in the model niches are filled in a more selective manner. The simplicity of the niche structure in these experiments is in marked contrast to the structure found in the problems that we are interested in.

Clearly in these studies are distinctive from qualitative example finding in that the aim is that of simulating what happens in the world of natural biology, whereas our work is focussed on applying the ideas to other problem domains. Nonetheless the ideas described in (Maley, 1999) begin the investigation of which evolutionary properties are required for active exploration of a niched environment. However the niching structure in these experiments is trivial, whereas in our problems the niches have a complex geometry in the search space. Seeing whether the conditions outlined by Maley for the exploration of this kind of space extend to these more complex niche-spaces could provide an interesting way forward for approaching these ideas.

Another area of biological simulation which might provide useful is the work that has used the immune system as its source of inspiration (Dasgupta and Attoh-Okine, 1997; Dasgupta, 1998). This is discussed further below.

CONSTRUCTING HEURISTICS FOR QUALITATIVE EXAMPLE FINDING.

In this section of the paper we shall discuss how we can develop heuristics for finding qualitative examples. Essentially this is a search problem—we want to search the space $\mathcal{O}$ of objects that fall into categories which have not been explored previously, making use of each attempt to learn some information about the structure of the search space and which areas might be more worthy of investigation in the future.

Desirable characteristics of an example finding heuristic.

We can draw up a number of properties that such an algorithm might require. Firstly the algorithm should be capable of exploiting any substructures that are discovered which are good building blocks for a wide variety of examples. These substructures could be represented explicitly, e.g. by doing some kind of commonality analysis on a set of objects which have already been found and which span a large number of classes. These common structures could then be built upon in many different ways. Alternatively we could represent the building blocks in a subsymbolic fashion. This is illustrated by genetic algorithms, where building blocks representing substructures which make a high contribution to fitness are indirectly represented by being present in a large proportion of the population. This lack of direct symbolic representation actually *adds* to the computational power of GAs—because the same "subsymbol" can play a role in multiple useful structures, thus leading to *implicit parallelism* (Holland, 1975; Goldberg, 1989).

A second desirable characteristic for these heuristics is that they should recognize when they have "mined out" a particular substructure, and move on to looking in other areas of the search space. This suggests that current explorations shouldn't base themselves on the entire history of the search procedure, but should forget structures after a few rounds of exploration. This is how population-based search techniques such as GAs get rid of information which is useless to them, and it is explicitly used in tabu search (Glover, 1989; Glover, 1990; Glover et al., 1993), where items on the tabu list are dropped after a few rounds. Similarly a heuristic must recognize (implicitly or explicitly) when a particular substructure is not a fecund base on which to build other original structures. For example in a bioinformatics setting we might be interested in the shape that is presented by a molecule to the outside world. In this case, any substructure that ends up folded inside the molecule and not exposed on the outside at all can be changed as much as possible and still have no impact on the class into which that molecule is put. An algorithm should recognize this (from the lack of changes of classification when this area is modified) and not waste further time exploring this area further.

Thirdly the algorithm should not return to areas that are mined out or unproductive. This could be achieved either directly or indirectly. An example of a direct method would be to use something akin to tabu search, which maintains a list of items in search space which have recently been visited and prevents the algorithm returning to them. Another strategy which has similarities to this, and which has been applied successfully to population-based incremental learning (Baluja, 1994), is the work of Sebag and Schoenhauer (1997), and Robillard and Fonlupt (1999), which update each round a vector of probabilities for each element in a bitstring summarizing the worst solutions so far in an optimization problem. The alternative is to represent these regions indirectly, for example in a genetic algorithm areas of search space which are not contributing to fitness will converge to values in which these areas are avoided.

A fourth characteristic that the algorithm will need is the ability to bring together compatible fruitful substructures, which suggests the use of a process such as recombination in GAs.

Some more subtle characteristics were discovered during early experiments with the algorithms. Whilst the characteristics above are not dissimilar to characteristics which have proven useful for optimization, these other characteristics are more particular to the qualitative example finding problem.

One problem with early implementations of heuristics for these problems was that solutions which had fruitful structures within them were not recognized. Consider a GA-like population based algorithm. We generate a number of solutions, several of which are in classed which haven't been found before. However one of these contains a very fruitful substructure, but because only one example of that structure has been found, it is swamped by all of the other less fecund solutions and sometimes not chosen for the next generation at all. We have been experimenting with a couple of remedies for this problem. The first of these (see below) is to use the data about which solutions come from which parents, and bias the selection scheme towards those individuals which come from parents that were more successful at producing novel solutions by recombination. A second approach would be to take individuals, apply a hypermutation to them (i.e. a mutation at many times the normal mutation rate) several times, and see which produced most novel solutions, then use those as the parents for the next generation.

Another problem that has occurred with a GA based approach is that sometimes the population will only find a small number of solutions. This obviously eliminates the diversity from the population. Experiments with hypermutation have proven successful at getting out of these problem areas without losing useful structures within the small number of solutions.

A GA-based heuristic.

Our initial experiments have concentrated on producing a genetic algorithm based heuristic for these problems. GAs are capable of exploiting niches within a population, and are a search method which is based on the exploitation of substructures within a search space. Therefore they would seem to be a good basis for building a qualitative example finding heuristic.

Clearly the main contrast between traditional GAs and the qualitative example finding problems is that there is no measure of fitness in the latter problem. It makes no sense to say that one example is "better" than another, or to give it any kind of rating. Our approach to managing this is to create a fitness value on the fly for each individual in each generation. This is entirely dependent on the context that the individual is in, and the history of which examples have been found so far in the run.

The simplest way in which to allocate this fitness is to take each item in the population, and allocate it a fitness of 1 if it is in a class that hasn't yet had and example, and 0 if it is in a class which already has an example. One disadvantage to this is that it scores all items the same, and makes to attempt to identify those which contain substructures which are likely to form the basis for other novel solutions. An alternative method which identifies these substructures consists of calculating which members of the current population are novel, giving their parents a score of 1 for each novel child, and then assigning the child the sum of its parents' scores (figure 1).

FIGURE 1 ABOUT HERE

This has been shown to produce more novel solutions in a shorter time on some test problems compared to simply assigning 1 for novel and 0 for not. Details can be found in the paper (Johnson, 2000).

Another feature that has proven successful in our GA approach to this problem is to apply a hypermutation when a population contains very few novel individuals. When a population has very few novel individuals then the diversity of the population in the next round will be very low, because none of the non-novel solutions will contribute to that round. In the most extreme case, if only one novel solution is found then there will be completely converged population. To avoid these problems we apply a much higher mutation rate for one generation once the size of the novel population is below a threshold. This reintroduces enough diversity into the population for it to be able to begin exploring again, whilst also retaining good substructures that are in the small population.

We have also carried out experiments on other aspects. Mutation rate experiments have tended to show that a higher mutation rate than typical is ideal on our test problems, a rate of 0.1/bit producing good search behaviour. This contrast with the low level of mutation, typically 0.01–0.001/bit, or $\frac{1}{\mathit{bitstring\text{-}length}}$ (Bäck, 1996) used in optimization GAs. This gives support to the idea that mutation is the main way in which these algorithms are exploiting good substructures. Another experiment which was less successful was introducing some random strings into the population each generation to provide a constant source of diversity—this was too blunt an instrument. Details of these experiments can be found in (Johnson, 2000).

Other approaches.

The GA approach has had some success, and we are looking at other approaches which could be used for this problem, either as a complement to or to work alongside the GA.

One approach which has potential promise is not to populate some arbitrary "population space" with the objects in the search space, but instead to populate the search space $\mathcal{O}$ with individuals, and provide ways for them to move through the space, ways for successful individuals to breed. Thus each individual agent would represent a good search strategy for the neighbourhood in which it is located, and ecological notions such as crowding could be used to control the exploration of the search space. Similar ideas have been used in robotics (Watson et al., 1999) and database searching (Degeratu and Menczer, 2000; Menczer, 1999; Morton-Firth and Bray, 1998). This approach may be more scalable than the GA approach, as the population can split into smaller subgroups whereas in the GA approach the population will often follow a single niche until it is fully exploited, then move onto another niche, rather than exploiting commonalities between niches to explore each niche in a faster way.

Another biological system which might be useful here are artificial immune systems (Dasgupta, 1998). The idea of an immune system is that it learns to recognize self from non-self objects, so it might be possible to use an immunity inspired system alongside some search method to learn what parts of the space have already been mined for examples (self) and which remains to be explored (non-self).

A final alternative might be to carry out a more symbolic, mathematical treatment of these problems. This could work by the use of some inductive process for learning constraints, e.g. the system could learn that certain parts of the search space are mined out and put a constraint on exploring that part of the space. Constraint programming (Marriott and Stuckey, 1998) could then be used to calculate areas of the search space that are most worthy of future exploration. Other related possibilities would be to analyse sets of current novel solutions directly for common substructures, and explore combinations of these substructures.

Conclusions and questions.

We has discussed the requirements for a search-based qualitative example finding heuristic, explained how we have implemented this heuristic using a variant on GAs and outlined alternative approaches. Our major effort now is concerned with scaling these approaches up to more realistic problems. A number of outstanding questions remain.

- How well do these methods scale to realistic problems?

- How general can these heuristics be? Will they be capable of being applied to a wide variety of problems as e.g. GAs, simulated annealing and neural networks have been, or will they need a lot of individual tuning for individual problems?

- How can we analyse these methods theoretically? Are there analogies of Holland's schema theorem (Holland, 1975) which will allow us to formalize some of the informal statements about how substructures come together.

- What kinds of tunable test examples can we create which will allow us to test conjectures about how these algorithms work?

CONCLUSIONS.

In this paper we have shown how a large number of problems from several different areas can be shown to be examples of the qualitative example finding problem. This motivates work on general meta-heuristics for this problem. We have shown that the problem is distinct from a number of other well known problems, and discussed how solutions to the well-known problems can be used as inspiration for our problem area. Finally we have discussed the design of qualitative example finding algorithms, and explained how we have modified the traditional genetic algorithm to provide a meta-heuristic for these problems.

ACKNOWLEDGEMENTS.

This paper was first given as a research seminar to the information systems research group at Kingston University, and I would like to thank the members of that group for their comments.

REFERENCES

Adams, C. A. (1994). *The Knot Book*. W.H. Freeman.

Anderson, R. and May, R. (1991). *Infectious diseases of humans : dynamics and control*. Oxford University Press.

Aneziris, C. (1994). Is a knot classification possible? Preprint DESY-94-230, Deutsches Elektronen-Synchrotron, Hamburg.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

Bäck, T., Fogel, D. B., and Michalewicz, Z., editors (1997). *Handbook of Evolutionary Computation*. Oxford University Press / Institute of Physics.

Bailey, N. (1975). *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin.

Baluja, S. (1994). Population-based incremental learning. Technical Report CMU-CS-94-163, Carnegie Mellon University.

Baumslag, G. (1993–present). Magnus: a system for exploring infinite groups. Mathematics Department, City College of New York. Available from `http://zebra.sci.ccny.cuny.edu/web/`.

Baumslag, G., Miasnikov, A., Miasnikov, A., and Shpilrain, V. (1999). On the Andrews-Curtis equivalence. Preprint, City College of New York Mathematics Department.

Belew, R. K. (2000). *Finding out about : information retrieval and other technologies for seeking knowledge.* Cambridge University Press.

Bentley, P. J., editor (1999). *Evolutionary design by computers.* Academic Press.

Birman, J. S. and Hirsch, M. D. (1998). A new algorithm for recognizing the unknot. *Geometry and Topology*, 2:175–220.

Bishop, C. and Hinton, G. (1995). *Neural Networks for Pattern Recognition.* Clarendon Press.

Boden, M. (1990). *The Creative Mind: Myths and Mechanisms.* Abacus.

Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. (1997). Syntactic clustering of the web. In *Sixth World Wide Web Conference.*

Brooks, R. A. (1991a). Intelligence without reason. Technical Report (A.I. Memo No. 1293), Massachusetts Institute of Technology Artifical Intelligence Laboratory.

Brooks, R. A. (1991b). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

Calégari, P., Guidic, F., Kuonen, P., and Kobler, D. (1997). Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing*, 47:86–90.

Cohen, H. (1999). Colouring without seeing: a problem in machine creativity. *AISB Quarterly*, 102.

Corne, D., Dorigo, M., and Glover, F., editors (1999). *New Ideas in Optimization*. McGraw-Hill.

Culberson, J. (1998). On the futility of blind search: An algorithmic view of 'no free lunch'. *Evolutionary Computation*, 6(2):109–128.

Dasgupta, D., editor (1998). *Artificial Immune Systems and their Applications*. Springer.

Dasgupta, D. and Attoh-Okine, N. (1997). Immunity-based systems: a survey. In *Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics*. IEEE Press.

De Bono, E. (1990). *Lateral Thinking*. Penguin.

Dean, J. and Henzinger, M. R. (1999). Finding related pages in the world wide web. In *Eighth World Wide Web Conference*.

DeAngelis, D. and Gross, L., editors (1992). *Individual-based Models and Approaches in Ecology*. Chapman and Hall.

Deb, K. and Goldberg, D. (1989). An investigation of niche and species formation in genetic function optimization. In (Schaffer, 1989), pages 42–50.

Deb, K. and Spears, W. M. (1997). Speciation methods. In (Bäck et al., 1997), pages C6.2.1–C.6.2.5.

Degeratu, M. and Menczer, F. (2000). Infospiders: Complementing search engines with online browsing agent. Submitted to IAAI-2000.

D'haeseleer, P., Forrest, S., and Helman, P. (1996). An immunological approach to change detection: Algorithms, analysis and implications. In *IEEE Symposium on Security and Privacy*. IEEE Press.

Ékart, A. (2000). Shorter fitness distance preserving genetic programs. In (Fonlupt et al., 2000). Lecture Notes in Computer Science 1829.

Epstein, S. (1988). Learning and discovery: One system's search for mathematical knoweldge. *Computational Intelligence*, 4(1):42–53.

Ewald, P. (1994). *The Evolution of Infectious Disease.* Oxford University Press.

Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems.* Wiley.

Fonlupt, C., Hao, J.-K., Lutton, E., Ronald, E., and Schoenhauer, M., editors (2000). *Artificial Evolution 1999.* Springer. Lecture Notes in Computer Science 1829.

Forrest, S., Somayaji, A., and Ackley, D. H. (1997). Building diverse computer systems. In *Sixth IEEE Workshop on Hot Topics in Operating Systems*, pages 67–72. IEEE Press.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing*, 1(3):190–206.

Glover, F. (1990). Tabu search—part II. *ORSA Journal on Computing*, 2(1):4–32.

Glover, F., Taillard, E., and de Werra, D. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41:3–28.

Goldberg, D. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In (Greffenstette, 1987), pages 41–49.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Greffenstette, J., editor (1987). *Genetic Algorithms and their Applications : Proceedings of the Second International Conference on Genetic Algorithms*. Erlbaum.

Grüninger, T. and Wallace, D. (1996). Multimodal optimization using genetic algorithms. Technical report, Massachussets Institute of Technology Computer Aided Design Laboratory.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Addison Wesley.

Hofstadter, D. and The Fluid Analogies Research Group (1998). *Fluid Concepts and Creative Analogies*. Penguin.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press. Second edition 1992.

Hoste, J., Thistlethwaite, M., and Weeks, J. (1998). The first 1,701,936 knots. *The Mathematical Intelligencer*, 20(4):33–48.

Huhns, M. N. and Singh, M. P., editors (1998). *Readings in Agents*. Morgan Kaufmann.

Jackson, H. (1999). A symbiotic coevolutionary approach to architecture. In (Patrizio et al., 1999), pages 49–54.

Jain, A., Bolle, R., and Pankanti, S., editors (1999). *Biometrics: Personal Identification in Networked Society*. Kluwer.

Jia, X. and Nixon, M. (1995). Extending the feature vector for automatic face recognition. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 17(12):1167–1176.

Johnson, C. G. (1999). Exploring the sound-space of synthesis algorithms using interactive genetic algorithms. In Wiggins, G. A., editor, *Proceedings of the AISB Workshop on Artificial Intelligence and Musical Creativity, Edinburgh*.

Johnson, C. G. (2000). Qualitative example-finding using genetic algorithms. In John, R. and Birkenhead, R., editors, *Proceedings of Recent Advances in Soft Computing 2000*. Physica-Verlag/Springer.

Jones, V. F. (1985). A polynomial invariant for knots via Von Neumann algebras. *Bulletin of the American Mathematical Society*, 12(1).

Kauffman, L. H. (1987). State models and the Jones polynomial. *Topology*, 23(3).

Kauffman, L. H. (1988). New invariants in the theory of knots. *American Mathematical Monthly*, pages 195–242.

King, R., Muggleton, S., and Sternberg, M. (1992). Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the National Academy of Sciences*, 89(23):11322–11326.

King, R., Srinivasan, A., and Sternberg, M. (1995). Relating chemical activity to structure: an examination of ILP successes. *New Generation Computing*, 13(3–4):411–433. See also correction in 14(1), page 109.

Koza, J. R. (1992). *Genetic Programming : On the Programming of Computers by means of Natural Selection*. Series in Complex Adaptive Systems. MIT Press.

Krzanowski, W. J. (1990). *Principles of multivariate analysis : a user's perspective*. Oxford University Press.

Langley, P., Simon, H., Bradshaw, G., and Zytow, J. (1987). *Scientific Discovery*. MIT Press.

Lenat, D. B. and Brown, J. S. (1984). Why AM and EURISKO appear to work. *Artificial Intelligence*, 23:269–294.

Levine, D. (1994). *A Parallel Genetic Algorithm for the Set Partitioning Problem*. PhD thesis, Illinois Institute of Technology/Argonne National Laboratory.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the Association of Computing Machinery*, 37(7).

Mafoud, S. W. (1992). Crowding and preselction revisited. In Männer, R. and Manderick, B., editors, *Parallel Problem Solving from Nature II*, pages 27–36. Elsevier.

Mafoud, S. W. (1997). Niching methods. In (Bäck et al., 1997), pages C6.1.1–C6.1.4.

Maley, C. C. (1998). *The Evolution of Biodiversity : A Simulation Approach*. PhD thesis, Massachusetts Institute of Technology.

Maley, C. C. (1999). Four steps toward open-ended evolution. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1336–1343. Morgan Kaufmann.

Marriott, K. and Stuckey, P. J. (1998). *Programming with Constraints.* MIT Press.

Meehan, J. (1977). Tale-spin, an interacive program that writes stories. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 91–98.

Menczer, F. (1999). Is agent-based online search feasible? In *Working Notes of the AAAI Spring Symposium on Intelligent Agents in Cyberspace.*

Menczer, F. and Belew, R. K. (1998). Local selection. In Porto, V. W., Saravanan, N., Waagen, D., , and Eiben, A., editors, *Evolutionary Programming VII : Proceedings of the Seventh Annual Conference on Evolutionary Programming.* Springer. Lecture Notes in Computer Science.

Menczer, F., Street, W., and Degeratu, M. (1999). Evolving heterogeneous neural agents by local selection. In Honavar, V., Patel, M., and Balakrishnan, K., editors, *Advances in the Evolutionary Synthesis of Neural Systems.* MIT Press.

Miasnikov, A. D. (1999). Genetic algorithms and the Andrews-Curtis conjecture. *International Journal of Algebra and Computation*, 9(6).

Miller, B. and Shaw, M. (1995). Genetic algorithms with dynamic niche sharing for multimodal function optimization. Technical Report 95-010, University of Illinois Genetic Algorithms Laboratory.

Miller, J., Kalganova, T., Lipnitskaya, N., and Job, D. (1999). The genetic algorithm as a discovery engine: strange circuits and new principles. In (Patrizio et al., 1999), pages 65–74.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Series in Complex Adaptive Systems. Bradford Books/MIT Press.

Morton-Firth, C. and Bray, D. (1998). Predicting temporal fluctuations in an intracellular signalling pathway. *Journal of Theoretical Biology*, 192:117–128.

Murasugi, K. (1996). *Knot Theory and its Applications*. Birkhäuser.

Nelson, P. (1999). Creativity and embodied rationalization. In Patrizio, A., Wiggins, G. A., and Pain, H., editors, *Proceedings of the AISB'99 Symposium on Musical Creativity*, pages 1–6. Society for Artifical Intelligence and the Simulation of Behaviour.

Nesse, R. and Williams, G. (1995). *Evolution and Healing : The New Science of Darwinian Medicine*. Weidenfeld and Nicholson.

Nowak, M. A. and Sigmund, K. (1992). Tit for tat in heterogeneous populations. *Nature*, 355:250–253.

Osman, I. (1996). *Meta-heuristics*. Kluwer Academic Publishers.

Parmee, I. (1996). Towards an optimal engineering design process using appropriate adaptive search strategies. *Journal of Engineerign Design*, 7(4):341–362.

Parmee, I. (1997). Cluster-oriented genetic algorithms (coga's) for the identification of high-performance regions of design spaces. *Journal of Applied and Industrial Mathematics.*

Parmee, I. (1998). Evolutionary and adaptive strategies for efficient search across whole system engineering design hierarchies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12:431–445.

Partridge, D. and Rowe, J. (1994). *Computers and Creativity.* Intellect Books.

Patrizio, A., Wiggins, G. A., and Pain, H., editors (1999). *Proceedings of the AISB'99 Symposium on Creative Evolutionary Systems.* Society for Artifical Intelligence and the Simulation of Behaviour.

Pétrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *IEEE International Conference on Evolutionary Computation.* IEEE Press.

Piatetsky-Shapiro, G. and Frawley, W. J., editors (1991). *Knowledge Discovery in Databases.* AAAI Press / The MIT Press.

Polya, G. (1945). *How to Solve It: A New Aspect of Mathematical Method.* Princeton University Press.

Racter (1984). *The Policeman's Beard is Half-Constructed.* Warner Books.

Reeves, C. R., editor (1993). *Modern Heuristic Techniques for Combinatorial Problems.* Blackwells.

Ritchie, G. and Hanna, F. (1984). AM: A case study in AI methodology. *Artificial Intelligence*, 23:249–268.

Robillard, D. and Fonlupt, C. (2000). A shepherd and a sheepdog to guide evolutionary computation. In (Fonlupt et al., 2000). Lecture Notes in Computer Science 1829.

Rosenman, M. (2000). Evolutionary case-based design. In (Fonlupt et al., 2000). Lecture Notes in Computer Science 1829.

Schaffer, J., editor (1989). *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kauffmann.

Sebag, M. and Schoenauer, M. (1997). A society of hill-climbers. In *IEEE International Conference on Evolutionary Computation.* IEEE Press.

Tanese, R. (1987). Parallel genetic algorithm for a hypercube. In (Greffenstette, 1987), pages 177–183.

Tanese, R. (1989). Distributed genetic algorithms. In (Schaffer, 1989), pages 434–439.

Tuson, A. L. (1999). *No Optimisation Without Representation.* PhD thesis, University of Edinburgh.

van Rijsbergen, C. J. (1979). *Information Retrieval.* Butterworths, London.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In Angeline, P., Michalewicz, Z., Schoenhauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 335–342. IEEE Press.

Witten, E. (1989). Quantum-field theory and the Jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399.

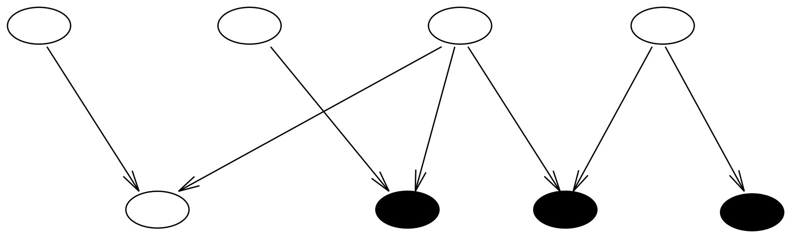
Wolfram, S. (1994). *Cellular Automata and Complexity.* Addison Wesley.

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. Technical Report Technical Report SFI-TR-95-02-010, Santa Fe Institute.

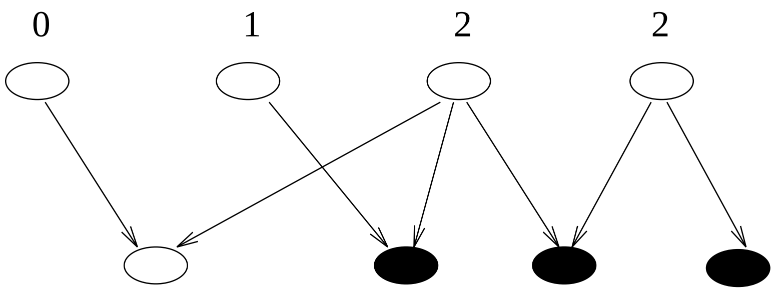
Yin, X. and Germay, N. (1993). A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In Albrecht, R., Reeves, C., and Steele, N., editors, *International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 450–457. Springer.

figure 1

Stage 1. Identify the novel solutions



Stage 2. Pass fitness back to parents



Stage 3. Pass the accumulated fitness back to the children

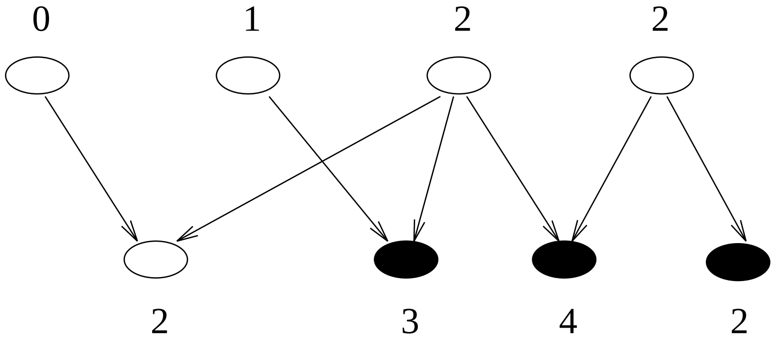


Figure captions.

Figure 1: A three-part algorithm for on the fly fitness assignment.