The Impact of Positive and Negative Feedback in Insight Problem Solving

Andrew Roxburgh
Supervised by: Dr. Antonija Mitrovic
and Prof. Stellan Ohlsson (University of Illinois at Chicago)

15 November 2004

Abstract

Insight problems are problems which are simple to state but relatively difficult to solve, and require some sort of 'insight' (creative thinking) to solve them. This insight requires looking at the problem or some objects within the problem in an unconventional way, or some other form of restructuring the problem. This project looks at the impacts of providing feedback in two contrasting ways to people attempting to solve one insight problem, the nine-dots problem. Feedback is given in either a positive format – telling them what they *should* do, or a negative format – telling them what they *should not* do. The relative benefits of these two forms of feedback are evaluated, and it is concluded that a positive style is better because people who receive feedback phrased in a positive style are more likely to follow it than those who receive it in a negative style.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Insight problems are difficult problems that appear simple, and require some form of 'insight' to solve them. The insight requires looking at the problem in a different way, or seeing some other solution path that is not obvious. They are interesting because by understanding how we solve insight problems we can better understand how we solve problems in general.

The nine-dots problem is one such problem. It involves drawing four straight lines, without lifting the pen, to cover nine dots arranged in a square. It requires drawing lines outside of the square formed by the dots, and making non-dot turns (ie. the solution contains turns outside the square). A number of studies have been carried out on the nine-dots problem, making it is the best understood of all insight problems.

Feedback is essential in all forms of education. It is used to provide information to the learner, and can be given in different styles. Positive and negative feedback differ in the manner in which information is given. When feedback is given to a problem-solver, it can be given in either positive style, telling the problem-solver what they should do, or negative style, telling the problem-solver what they should not do.

This experiment evaluates the impact of giving feedback to people attempting the nine-dots problem. Software is developed that lets people attempt the problem. One version gives all feedback in a positive manner, the other version gives all feedback in a negative manner. Volunteers are recruited to use the system and their actions and success rates recorded.

It is concluded that phrasing feedback in a positive way is superior to phrasing it negatively. Problem-solvers who received positively phrased feedback were neither significantly more likely to solve the problem than those who received negatively phrased feedback, nor more likely to do so faster. However, feedback messages given in a positive manner were more likely to be adhered to by the problem-solver than those given in a negative manner.

This report is structured as follows: First, Chapter 2 gives background information on insight problems in general, the nine-dots problem, and the difference between positive and negative

feedback. The evolution of the project and the software developed for it are discussed in Chapter 3, the experiment is described and its results analysed in Chapter 4 and conclusions and future work are discussed in Chapter 5.

Chapter 2

Background

2.1 Insight

This section will define the concept of 'insight', which is fundamental to this research. Motivation for why insight is studied is given, and two insight problems are discussed as examples. Several competing theories about insight as a concept are discussed.

Insight is a psychological phenomenon concerning how people solve a certain kind of problem, known as 'insight' problems. According to Kershaw & Ohlsson (2004), insight problems are generally considered to be problems that:

1. Can be stated simply.
2. Contain only a small number of objects and relations.
3. Have a solution which is relatively simple once it is known.
4. Are nonetheless very difficult to solve.

Because the solutions are fairly simple once the 'insightful' part is known, someone who has seen the solution to an insight problem before is likely to be able to solve it very quickly in the future. The idiom "thinking outside the square" refers to the process needed to solve insight problems. Examples of insight problems are discussed in the following few sections.

When solving insight problems, there is generally a period of action in which the problem-solver tries whatever paths toward the solution seem initially obvious. This period of action is generally unsuccessful, and is followed by an impasse; at this point it is not uncommon for the problem-solver to believe that the problem is unsolvable. The impasse, however, is often followed by the correct solution, which always requires some sort of 'insightful' change in thought process or problem representation by the solver (Knoblich, Ohlsson & Raney 2001). The time when this change in thought process or problem representation occurs is known as the 'moment of insight', and the actual change itself is referred to as 'insight'. Knoblich et al. (2001) conducted a study in which they tracked problem-solvers' eye movements as they attempted insight problems. This

study found that initially the problem-solvers' eyes moved rapidly around the problem, but as time went on they became increasingly stationary. This was when they reached impasse. The data on where the eyes were focused were also used to analyse what parts of the problem people spent their time on.

It is not properly understood what happens at the moment of insight, or why there is an impasse at all. That people can solve insight problems given enough time shows that they are capable of finding the solution, so the question remains: Why do they not solve them straight away? Psychologists and cognitive scientists have come up with various theories as to how people solve problems in general, and some consider solving insight problems to be simply a special case of this. Others consider insight problem solving to rely upon entirely different mental processes to non-insight problem solving. Section 2.3 discusses this further.

A person's ability to solve insight problems is not a measure of their intelligence, but it has been shown that a greater ability to solve normal (non-insight) problems gives an increased probability at having a greater ability to solve insight problems (Maier & Janzen 1969). It has also been shown that higher scores in math problems corresponds to a greater ability to solve insight problems, but that there is no correspondence with higher scores in verbal problems (Maier & Casselman 1970*b*).

Although insight problems are so-called because they generally require a single, insightful thought in order to solve them, both Maier & Casselman (1970*a*) and Kershaw & Ohlsson (2004) show that not all ostensibly insight problems have a single source of difficulty. In these two studies, even when given hints directly pertaining to the part of the problem which requires insight, far less than 100% of students solved the problems. Weisberg (1995) argue that many problems which are commonly referred to as insight problems are, at best, 'hybrid insight problems' because restructuring the problem is not the only way in which the solution can be achieved. He classifies the nine-dots problem, which is the insight problem used in this study (see Chapter 4), in this way.

Weisberg (1995) continues to argue that studies based only on what he calls 'pure insight problems' may be accepted as relevant to insight research. However, the many studies into the nine-dots problem as an insight problem show that this view is far from mainstream. That there is not a single source of difficulty in solving a problem does not mean that it is not an insight problem; it can still fill the necessary conditions of having a low number of objects and relations and a simple solution, but remain difficult when more than one source of difficulty is present.

Insight has been considered not just on a single problem-solving level, but on the scale of revolutionary scientific developments. Ippolito & Tweney (1995) and Gruber (1995) are two examples of works that speak of major scientific discoveries, such as those by Archimedes, Einstein and Darwin, in terms of insight. If the insightfulness needed to create or discover new scientific theories is the same as that needed to solve insight problems, then there is much added motivation to better understand how we solve these problems, and especially how we can improve our abilities to solve these problems.

2.2 Example Insight Problems

The nine-dots problem is the insight problem studied in this experiment; here two other examples of insight problems are given to explain the concept.

2.2.1 The Six Match Problem

The goal in the six match problem is to arrange six matches to form four equilateral triangles. The length of each edge of each triangle must be the length of one match – the matches are not allowed to be broken or bent at all.

The solution to this problem involves building a three-sided pyramid. The insightful part of this problem is that the matches must be arranged in three dimensions. People attempting this problem invariably work only in two dimensions, even though it is fairly obvious after only a few attempts that no solution is possible in this way.

2.2.2 The Two String Problem

The goal of the two string problem is to tie together two strings that are both hanging from the ceiling at opposite ends of a room. They are long enough to reach each other, but not long enough that it is possible to reach one whilst holding the other one. Also in the room are a chair and a table, and on the table is a screwdriver and a piece of string, but this piece of string is not long enough that even when it is tied to one of the hanging strings, the other hanging string could be reached. Figure 2.1 shows this set-up. The two string problem was introduced by Maier (1931).

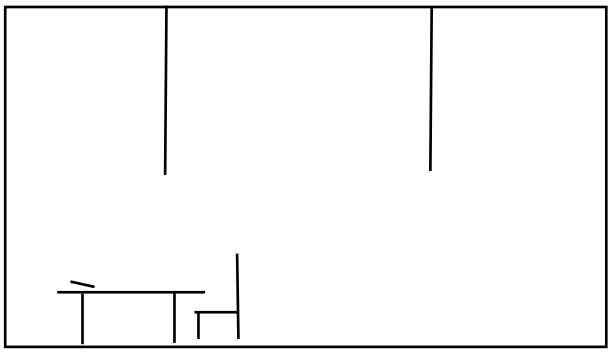

Figure 2.1: The set-up of the two string problem. The goal is to tie the two hanging strings together.

A number of different incorrect solutions are commonly seen in experiments. These include trying to lengthen one of the strings by tying the string on the table to it, trying to somehow 'hook' the other string while holding one of them, and trying to tie one string to the chair and leaving it in the middle while bringing the second string to it. The correct solution (the one that actually works) is to form a pendulum by tying the screwdriver to one string and starting it swinging, and

bringing the other string to the middle while the first still swings. The pendulum can then be caught and the two strings tied together (Weisberg 1995).

The insightful part of this problem stems from using an everyday object in a novel way. The problem solver must realise that the screwdriver can be used to form a pendulum, which can be used to bring the two strings together. It fits the definition of insight problems given above because it is a simple problem: the complete description here took only one reasonably short paragraph; a person shown a room with two strings could have the problem explained in five words — "tie those two strings together". The number of objects and relationships in the problem is very small: only two strings and a few assorted objects also in the room. The solution, of making a pendulum out of one of the strings using the screwdriver, is fairly simple, once it is known, but is not obvious given the problem description. In order to solve the problem it must be re-arranged in the problem-solvers mind; the screwdriver must be viewed not as a common household tool, but as an object of mass, which can be tied to a string.

2.3 Theories of Insight

There are two major differing views of what insight is: the 'special process' view, that insight is a separate process from other mental processes, and the 'nothing special' view, that solving insight problems follows the same process as solving other types of problems (Davidson 1995).

Davidson (1995) discusses research that found people solving non-insight problems generally had a good idea when they were getting closer to the solution, but that when solving insight problems were bad judges of how close to completeness they were. When attempting insight problems, people tended to predict the opposite of the truth: if they were near solving they thought that they were not; if they were not near solving they thought that they were. This finding leads to the belief that the mental processing of insight problems is fundamentally different from normal problems, but says nothing about what this processing is. The 'Aha' experience of solving an insight problem (Auble, Franks & Salvatore A. Soraci 1979), (Trottier 2003), (Chronicle, MacGregor & Ormerod 2004) supports the finding that people do not know when they are about to solve such a problem, but again does nothing to show what actually happens at this moment.

The alternative view, that insight problem solving is really no different from normal problem solving, is based upon this point. It essentially says that because no special sort of mental processing has yet been found, it is incorrect to presume that one exists. They argue that it is only conflicting past experience and prior expectations that impede solution to insight problems (Weisberg & Alba 1981). To refute this point, however, Davidson (1995) says that insight has not been identified because it is complicated; she theorises that insightful thinking is made up of at least three distinct processes which make it especially difficult to define exactly.

2.4 The Nine-Dots Problem

The nine-dots problem is perhaps the most studied of all insight problems. It has been studied since 1930, when Maier (1930) introduced it. The goal in the nine-dots problem is to draw four straight lines such that nine dots arranged as in Figure 2.2 are each intercepted by at least one line. The lines must be drawn as if without lifting a pencil from the page, so each line must start from where the previous one stopped. The lines are allowed to cross each other, but the problem must be solved without tracing back over any of the previous lines. The solution involves drawing the lines in an arrow-head shape, as shown in Figure 2.3. The solution can be drawn by starting from either end of the arrow. The only other solutions are equivalent to this one; they are all either rotations, or reflections, or both, of the same arrow-head shape.

In the nine-dots problem, although the need to draw lines outside of the square formed by the dots is considered the insightful part of the problem, solution rates when given this information do not raise above 40% in a short period of time (Kershaw & Ohlsson 2004). This is not because the nine-dots problem is not an insight problem, but rather because in order to solve the problem drawing lines outside the square is not the only thing that is required — lines must stop in places that are not over any dots ('non-dot turns') and the lines drawn must be in the arrow-head shape.

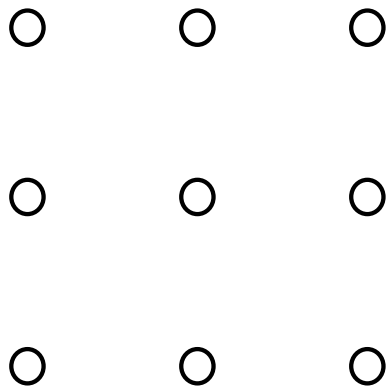

Figure 2.2: The arrangement of the nine dots in the nine-dots problem.

As mentioned above, the nine-dots problem has been well studied over the past 74 years. Theories of insight have been built around these studies, and hypotheses regarding the exact cause of the difficulties in the problem developed. As examples, several of the more interesting recent nine-dots problem studies are discussed here.

In MacGregor, Ormerod & Chronicle (2001) the authors explore a new theory of how people approach the nine-dots problem. They postulate that there are two principals that problem-solvers use when solving the insight problems:

1. A 'locally rational' operator which says how many further dots must be intersected by the next line.

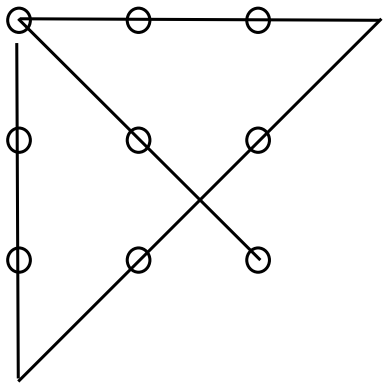

Figure 2.3: The solution to the nine-dots problem.

2. A 'global criterion' to measure overall progress through the problem.

Together, these two principals say that at each stage the problem-solver will draw a line that intersects as many new points as possible, and that if not enough points can be intersected from the current state, more possible solutions should be considered. That is, at first we simply try and intersect as many points with each line as possible, but when we discover that this does not work, we broaden the locally rational operator to allow us to consider moves ('emergent moves' is the term used in the article) that don't immediately intersect the most points.

This model is used to predict the percentage of people who will solve the nine-dots problem, and several variations of it, with some accuracy. This is interesting, and important, because it successfully provides a *quantitative* measure of the difficulty of an insight problem for the first time. Earlier studies had given only *qualitative* measures — the nine-dots problem is hard because the problem-solver must make turns outside of the square formed by the dots — but had not been able to say *how difficult* they would be. The authors later applied the same theory to other insight problems (Ormerod, MacGregor & Chronicle 2002).

The nine-dots problem is also studied in Akin & Akin (1996), but, surprisingly, in the context of architectural design. The authors conduct experiments to show that the problems typically encountered when solving the nine-dots problem have equivalents when solving certain design problems, and they argue that the same sort of insight is needed to solve both. This provides further motivation for understanding the nine-dots problem, and understanding how to help people solve the sorts of problems that it contains, because such skills could be transferred to other domains.

2.5 Positive and Negative Feedback

Feedback is a crucial part of all educational paradigms. If we are teaching ourselves to hit a golf ball, or play the piano, or any other task, we get feedback from how well we accomplish the task. If we have no idea what a good golf shot looks like, or what the piano piece should sound

like, we have no idea whether we are learning. In order to actually *learn*, feedback is crucial. Similarly, when we are being taught by another person (eg. a teacher, tutor, or lecturer) one of their primary roles is to give us feedback as to how we are doing. A teacher who merely provides information but does not give the student the opportunity to practise applying what they have learnt does not truly *teach*, because they do not give opportunity for mistakes to be discovered and corrected, by not having the opportunity to give feedback. Even in computer-based teaching systems the notion of feedback is critical; a system that gave problems but did not even evaluate their correctness would not be called a *teaching* system. A computer-based system, or a human being, that did not at least give the correctness of questions answered by the student would not be considered a teacher any more than a textbook with no exercises would; they would be merely a mechanism for knowledge transfer. For real *teaching*, providing some form of feedback is essential.

2.5.1 Educational Implications

Given the essential nature of providing feedback, it is important educationally to know how this feedback should be provided. Many aspects of giving feedback can be varied. The medium by which it is given can be changed: we can give feedback in words, in pictures or diagrams, or in some combination. The time to give feedback can also be changed. Feedback could be given immediately after a mistake has been made or part of the problem completed successfully, or it could only be given when the student has had time to continue working unassisted for a while. Allowing them to carry on alone could possibly help them understand by themselves what they have done right or wrong. The emotional style in which e the feedback is given can also be changed. An important question that can be asked is: Should the feedback tell them exactly what they *should* do, or should it tell them what they have done that they *should not* do? In terms of educational philosophy, the two feedback styles are at polar extremes: to aid understanding by showing what is right, or to point out what is wrong but let the student deduce what is right for themselves?

Elster (2000) describes how constraining ourselves can be beneficial in the creative arts. He argues that "sometimes there are benefits from having fewer opportunities rather than more"[1]. One example he gives is that the usual rules of writing prose (which words, tones, etc. can be used in given situations) constrain the author into something a reader can understand, but still leave room for creativity. He mentions James Joyce's almost unreadable Finnegan's Wake (Joyce 1939), which deliberately ignores these rules (by containing many obscure references and words borrowed from many other languages), as a work which has at best 'debatable ... artistic gain', and would almost certainly have no artistic merit were it written by someone of less genius[2]. Although creative writing is not the same as creative problem solving, this argument supports the point that by limiting ourselves we can still be creative, and that perhaps we can be creative in a more worthwhile way by imposing constraints upon ourselves. Providing negative feedback is a way of constraining the problem-solver.

[1] page 1
[2] page 210

2.5.2 Cognitive Science Implications

The difference between providing feedback in a positive way compared to providing it in a negative way is subtle when compared with the psychological implications if one works significantly better than the other. If positive feedback works significantly better it would imply that the problem-solver could not originally conceive of the correct solution, but by providing them with our positive feedback, giving them new information about what the solution should look like, we open new possibilities to them which enable them to solve the problem. The implication of negative feedback working, however, is that the problem-solver had the correct solution available to them but was fixated on at least one competing solution. By giving negative feedback we would eliminate some of the incorrect solutions from the options available to the problem-solver, thereby making only the correct path to solution available in their solution tree.

Figures 2.4 and 2.5 show diagrams of solution trees, and the actions that the two forms of feedback effect. In the diagrams, the solution tree is represented as a series of lines and ovals. The ovals represent possible solutions, with the solid oval being the correct solution, and the lines represent possible actions at each stage. Note that here solution is used to mean any possible approach to solving the problem, correct or otherwise. All the white ovals can therefore be referred to as 'incorrect solutions'. Figure 2.4 shows the situation that positive feedback is expected to aid. Originally no path to the correct solution exists, but positive feedback will provide the jagged line, making it possible to find the correct solution. Figure 2.5, the problem tree that negative feedback working would imply, shows that originally the whole solution tree is accessible to the problem-solver, but they are drawn to the two middle branches of the tree (the branches drawn thicker in the diagram). Negative feedback can eliminate these incorrect branches, bringing the correct path in the tree to the fore. Note that the key difference between the two is that with the positive feedback tree, the correct solution is not available at the start, but we make is possible by providing feedback. With negative feedback, the problem-solver is able to find the correct solution from the beginning, but they do not because other possible paths in the solution tree are more prominent. By providing negative feedback we remove these branches, or decrease the probability of using them, so that the correct solution can be found.

The idea that people attempting insight problems have the solution available to them, so to speak, but do not find it because they are fixated on other possible paths to solution is not a new one. The idea, however, of curtailing these incorrect solution paths by effectively crossing them out using negative feedback is. The same information can be contained in a negatively phrased feedback message as in a positively phrased message, which could equally be used to remove branches from the solution tree. The point is, however, that phrasing the information negatively explicitly says to the problem-solver that they should not proceed down this branch; the positive version is more indirect in that it does not explicitly remove the branch but says instead that another needs to be used. An outcome of this project will be to determine whether this more direct reference to the action that needs to be taken with regards to modifying the solution tree is any better or worse than the perhaps more intuitive alternative.

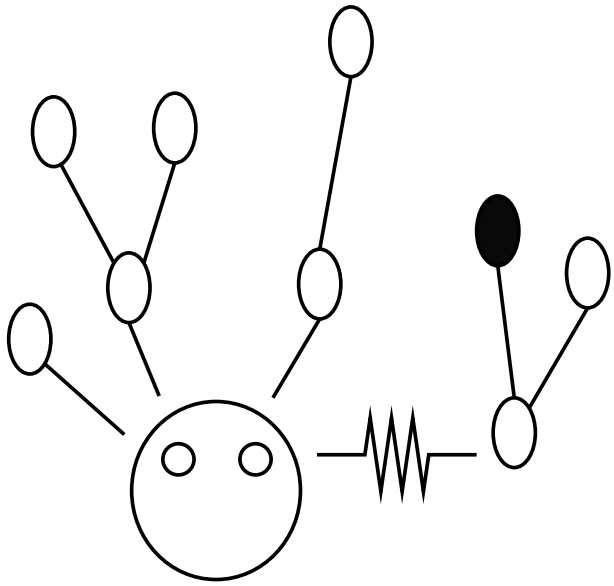

Figure 2.4: Positive Feedback. Each oval represents a way of solving the problem. The black oval is the correct solution. By giving *positive* feedback we add the jagged line, making it possible to solve.

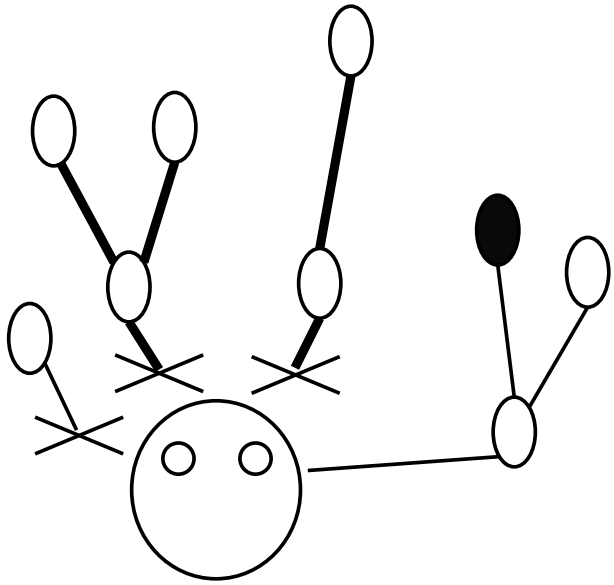

Figure 2.5: Negative Feedback. Each oval represents a way of solving the problem. The black oval is the correct solution. The thicker lines are the ones that the problem-solver originally considers. By giving *negative* feedback, we eliminate incorrect solutions, bringing the correct solution to the fore.

It could be argued that, given that negative feedback only eliminates what is wrong, but does not explicitly say what is right, there are problems for which negative feedback has no practical use.

If the tree of possible solutions is very flat, so there are many possible first moves but no second moves, it seems that if we simply eliminate the branches one by one the solution will continue to evade us, due to the large number of branches that must be removed. Certainly if there were a problem that had an infinite number of possible solution paths, each of which was different in *every* possible way, negative feedback could not be used to solve it. It is hard to imagine such a problem, in which there is not any similarity between some of the possible solutions. All problems could therefore be solved by someone receiving either positive or negative feedback, but some problems will be better suited to positive feedback than negative, and vice versa. A problem with a very wide solution tree will be better suited to positive feedback, so that every branch must not be eliminated separately, although this could still work. A problem with a very deep solution tree would be better suited to negative feedback. Negative feedback would be more appropriate for a tree with several very deep branches, where many solutions could be eliminated by a single negative feedback message. Figure 2.6 shows an example of a wide solution tree which would not suit negative feedback. Figure 2.7 shows a deep solution tree which negative feedback would be well-suited to.

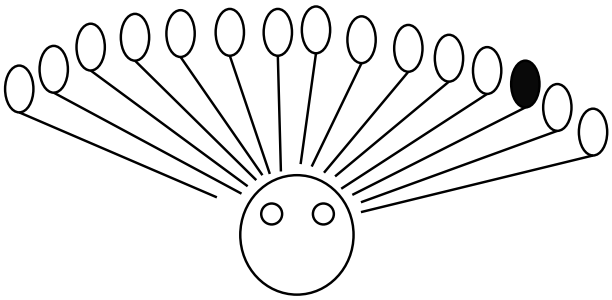

Figure 2.6: A very wide solution tree. Negative feedback could still work, by eliminating the all incorrect options, but positive feedback will be better.

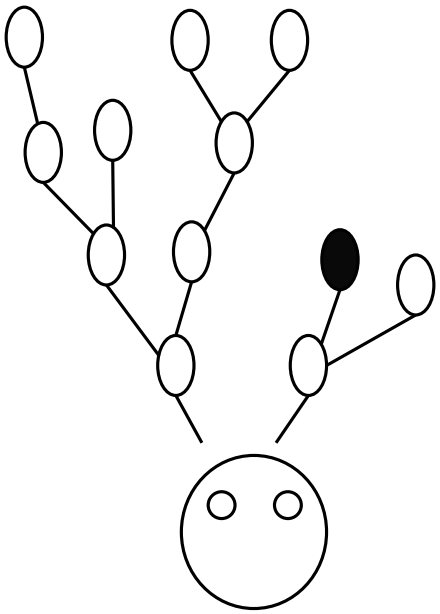

Figure 2.7: A very deep solution tree. Negative feedback could work well, because a single message can eliminate many incorrect solutions.

Chapter 3

Project

3.1 Evolution of the project

The original goal of this research was not to evaluate feedback paradigms in insight problem solving, but to create an Intelligent Tutoring System (ITS) that could teach users how to solve *any* insight problem. Intelligent Tutoring Systems (ITSs) are automated tools that seek to facilitate learning by replicating the process of human tutoring. It has been shown that one-to-one teaching is more effective than normal classroom teaching (Bloom 1984) so ITSs aim to replicate this. Intelligent Tutoring Systems have been shown to work in many domains where there are well-defined algorithms that state how problems should be solved (mathematics, physics) and when there are theories as to what the solution should look like (database design, computer programming). For examples see Anderson & Skwarecki (1986) (computer programming in Lisp), Goshi, Wray & Sun (2001) (computer science theory), Mitrovic, Martin & Mayo (2002) (database programming in SQL), Anderson, Corbett, Koedinger & Pelletier (1995) (geometry). All these ITSs, and almost all others, have been built to teach task-specific skills — skills that can be applied only in their domain and no others. An ITS for insight problems would teach the more general skill of creative thinking. It was intended that software systems for several insight systems would be developed, and that these would, as well as giving the *problem-specific* feedback provided by the nine-dots system, also give *problem-independent* feedback that would help solve any insight problems. These *problem-independent* hints would include advice such as "do not repeat actions which you now know do not work — try and do something *different*."

Through two small pilot studies and various changes in the feedback conditions[1] the project developed into the experiment discussed in this document. Details of the intermediate stages and goals of the project are given for completeness, and to explain how the feedback conditions were decided upon. The first goal was to create an ITS as discussed above, and comparatively little feedback was given in this version. The second version was also an ITS as above, and attempted to provide more information directly relating to the nine-dots problem, rather than insight problems in general, in the feedback, but to provide feedback only when patterns of lines were repeated in

[1] Feedback Condition is a term used throughout this document to mean the prerequisites for a certain feedback message to be given.

separate attempts at the problem. Because this too was found to be unsuccessful in helping people learn how to solve insight problems, the objectives of the project again changed tack, to focus on the differences of two forms of feedback as already described. The transition points between these three goals were two small pilot studies of the systems. The pilot studies took place on the 28th of June and the 30th of August, and are described in this section.

3.1.1 Pilot Study 1

On Monday 28 June a brief pilot study of the nine-dots problem ITS was carried out. The purposes of this study were two-fold: it was partly to investigate the usability of the graphical user interface of the ITS, and partly to investigate the effectiveness of the ITS. Although the study was limited in scope, it uncovered some areas of concern and some areas for further work.

Participants

Four volunteers participated in the study. All volunteers were post-graduate students of the Computer Science and Software Engineering Department at the University of Canterbury. None were familiar with the nine-dots ITS, but two recalled having seen the nine-dots problem before.

Procedure

Each of the participants were shown the nine-dots problem and given brief demonstrations of using the ITS. Once they were familiar with it they were left to use the system and attempt to solve the problem. Log files detailing all of the attempts at the problem by each participant were created. A brief informal discussion on the nine-dots problem and the ITS took place with each participant separately afterwards.

Hints

There were four conditions in which hints were presented to the user of the ITS. These were:

1. No lines outside square. If the recent attempts had not included any lines outside the square formed by the nine dots, a hint was given.

2. Repeated patterns. If a participant repeated a starting pattern of lines several times in succession, a hint was given. At this stage four patterns that were considered the most likely to be repeated were built into the system; later evolutions dynamically match any pattern, so any pattern that is repeated can generate a hint message (see Section 3.1.3).

3. Time idle. If a participant was idle for more than 45 seconds, they were given a hint encouraging them to 'have a go'.

4. Correct solution shape but wrong size. If a participant repeatedly drew the correct arrow-head shape of the solution, but drew it too small, a hint was given.

Experiment Results

Two of the participants revealed having seen the nine-dots problem before, although neither could remember the solution immediately. One of these participants solved the problem shortly after receiving the 'you can draw lines outside the square' hint; the other solved it after 3.5 minutes, on their 5th attempt at the problem. Neither of the two participants who had not seen the problem before made any real progress toward solving it. No time limit was given, but one gave up after 2.1 minutes and the other after 5.4.

Hint-conditions encountered

Tables 3.1 and 3.2 contain information about the number of times the hint-conditions were violated by each of the pilot study participants. Table 3.1 contains information about the two participants who were unfamiliar with the nine-dots problem whilst Table 3.2 contains information about the two participants who had previously seen the problem. The 'correct shape, wrong size' condition never came up and although most participants received the 'time idle' hints the log files unfortunately did not record this information.

Participant no.	No. of Attempts	Outside Constraint	Pattern Constraint
1	11	6	-
2	8	5	2

Table 3.1: Pilot Study 1: Hint-condition information for those unfamiliar with the nine-dot problem

Participant no.	No. of Attempts	Outside Constraint	Pattern Constraint
1	5	1	-
2	5	-	-

Table 3.2: Pilot Study 1: Hint-condition information for those familiar with the nine-dot problem

Conclusions

From observing participants using the system and discussing it with them afterwards, several avenues of further work appeared.

1. Stronger Hints Toward Solution. The most worrying part of the pilot study results was that it did not appear as if participants not already familiar with the problem's solution were making any significant progress toward achieving it. Despite receiving hints to the contrary, the participants continued to repeat similar patterns of lines and did not draw anything approaching the right shape. The participants tended to lose interest and drift off, so to keep them interested in the problem and to give them a realistic chance of solving it, the ITS was extended so that as well as providing the existing hints, it would drive them more strongly toward the correct solution with hints more specific to the nine-dots problem. One

such hint is 'Make sure you draw lines outside of the square formed by the dots'. Hints such as this are problem-specific so it was not envisaged that they would help participants learn how to solve unrelated insight problems, but only that they would provide motivation to continue using the ITS. More problem-specific hints and increased motivation were intended to increase the rate of solution of the problem. Having more participants solve the problem was considered important because if participants did not solve the nine-dots problem, they would probably be less likely to learn from the problem-independent hints.

2. Hints always available. Related to the above issue was a problem raised by one participant — most of the time the participants were left to attempt the problem by themselves and only sometimes did a hint appear. The participant was of the opinion that either always providing hints or having a 'request hint' button (as some other computer-based tutors do) would be useful.

3. Resetting before three lines. When attempting the nine-dots problem it is often obvious after two lines have been drawn that the solution is not correct. The nine-dots ITS, however, only analysed attempts when at least three lines had been drawn. It was observed that this resulted in potentially useful information about one participant being lost because when it became apparent that the solution was not correct they reset the problem and started again. To counter this, the algorithms for matching patterns were updated to match only two lines.

3.1.2 Continued Work and a Change in Focus

According to decisions made on the basis of Pilot Study 1, hints containing more problem-specific information were added to the ITS. There was only one condition on which a hint would be given in this version: only when the problem-solver made several attempts that started with equivalent patterns to one another. Equivalent patterns here means any patterns that are the same when reflected or rotated around. Matching equivalent patterns is discussed in Section 3.1.3. The rationale behind this decision was that when attempting insight problems, it is when impasse is struck that help is required. It was reasoned that when the problem-solver started repeating themselves it could be concluded that they had reached an impasse. Hint messages varied in this version, depending upon what sort of pattern was repeated, but the messages contained problem-specific information such as 'draw lines outside the square'.

The focus of the project was shifted, however, when it was realised that it would still be very uncommon to actually receive messages from this system. The number of possible distinct starting patterns is huge, and given that volunteers could only be expected to use the system for a short period of time, it was unreasonable to expect that many patterns would be repeated. They would therefore not receive much in the way of feedback, and not have the opportunity to learn how to better approach insight problem solving.

The new focus of the study was decided to be evaluating any differing effects of positively and negatively worded feedback styles, within the context of the nine-dots problem. The aim would no longer be to provide problem-independent hints that could aid in the solution of any insight

problem, but would instead be to try and specifically help to solve the nine-dots problem. For this purpose, the software system was split into two versions — one with entirely negatively phrased feedback hints and the other with entirely positively phrased feedback hints. Conditions for receiving feedback remained the same as before the positive-negative split was made, and a second small pilot study was conducted to evaluate the effectiveness of these new feedback conditions.

3.1.3 Recognising Repeated Patterns

The problem-solver repeating a pattern of lines in sequential attempts was the only condition for feedback in one version of the system, and it forms one of the feedback conditions in the final version. How this is achieved is described here.

The nine-dots problem has two lines of reflectional symmetry, and rotational symmetry of order four. This means that there are many solution shapes which are equivalent. For example, an 'N' shape and a 'Z' shape are the same shape but rotated by 90 degrees. When recognising repeated patterns, it is desirable that patterns that are equivalent (but transformed) are still recognised as being equivalent. For this reason, whenever a new attempt is created it is *standardised*. A standardised series of lines is one which starts from the top-left dot and first moves in a right-and-downwards direction. Once the first line has been rotated and/or reflected to achieve this, all other lines are transformed in the same way.

3.1.4 Pilot Study 2

On Monday 30 August a second brief pilot study was conducted for the nine-dots problem project. Only two participants could be found for this study, largely because many potential volunteers were previously made familiar with the nine-dots problem, the study, and the project as a whole, by a progress speech given by the author. Both participants were post-graduate students of Computer Science and Software Engineering at the University of Canterbury. The goal of the study was to determine whether the system gave enough feedback.

Feedback in this version was still given only when patterns were repeated. If the same pattern was repeated more than twice, feedback messages would be given depending upon the pattern. If the pattern did not go outside of the square formed by the dots the feedback would contain a hint regarding this. If it did not have the correct number of turns the feedback would contain a hint regarding this. Continued repetition of the same shape would cause the same feedback to be given, but in an increasingly forceful fashion. When patterns were not repeated the feedback simply contained the message 'that is not correct'.

Results

Both students used the negative-feedback version. One student used it for 6 minutes 50 seconds. The other used it for 6 minutes 40 seconds. The first student made 14 separate attempts; the second 10. The first student received feedback beyond 'that is not correct' 3 times, and the second did twice. Neither ever got any more advanced feedback. Neither student ever had the 'insight'

to try drawing lines outside of the square, and both had basically given up by the end. It did not appear as if either would have solved the problem in the short-term future.

Conclusions

It was concluded that more feedback was required, but it was not obvious when it could be given. It was decided that feedback should be given in four situations. These are:

- Angle. The correct solution to the nine-dots problem requires three 45 degree angles. If the attempt does not contain three such turns, feedback can be given.
- Outside. The correct solution requires that lines be drawn outside of the square formed by the nine dots. If no lines in the attempt go outside the square, feedback can be given.
- Pattern. If an incorrect Pattern of lines (see Section 3.1.3) is repeated, feedback can be given.
- Time. If the problem-solver is inactive for 45 seconds, a message is given encouraging them to keep trying. This is to keep them interacting with the system so that their attempts can be recorded, and because the more active they are, the more feedback messages they will receive.

It was decided to give all possible feedback, as often as possible. The pilot studies showed that the problem is difficult and giving minimal feedback is not particularly useful, so to give as much chance as possible to the problem-solvers to actually complete the problem, an exuberant approach to providing feedback was deemed to be better than a miserly one. For this reason, repeated pattern feedback was given the second time a pattern was repeated. Each subsequent time it was repeated the feedback got stronger, for a maximum of three times.

3.2 Software System

A software tool was developed that allowed students to attempt the nine-dots problem. The tool was developed in Java. It gave feedback after each attempt at the problem. Two versions were created: one which gave only positive feedback and one which gave only negative. The architecture of the system can be split into two main parts: there is the `Graphical User Interface` (GUI) where the problem-solver can attempt the problem and receive feedback, and there is the `Analyser` which stores and evaluates attempts at the problem, and decides what feedback to give.

Figure 3.1 shows a diagram of the basic structure of the system. In reality it comprises 33 `Java` classes, but the diagram shows only the essential details. The four stages that occur whenever a user makes an attempt at the problem are enumerated in the diagram. They are:

1. Attempt is transferred to Analyser. Originally the Attempt is a series of points where the problem-solver clicked to draw the lines. This is transformed to the internal abstraction of lines and directions, and information such as the angles between the lines is generated. If the solution is correct the GUI is notified straight away and displays congratulations to the problem-solver.

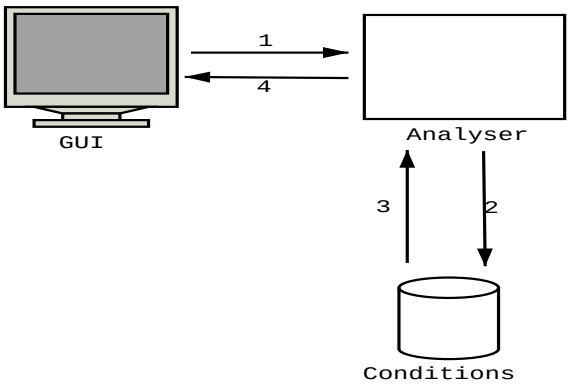

Figure 3.1: The architecture of the software system.

2. If it was not correct, the Attempt and all previous Attempts are then passed on to each of the Conditions.

3. Each Condition evaluates the Attempts, and either sends feedback back to the Analyser or sends nothing back.

4. The Analyser passes all feedback that is generated back to the GUI. The GUI then displays this to the user, highlighting appropriate words.

3.2.1 Graphical User Interface

A screen shot of the GUI for the nine-dots problem interface is shown in Figure 3.2. The GUI is split into three segments. The top contains instructions about the problem, the bottom is where feedback is given, and the middle shows the nine dots themselves. To attempt it, the problem-solver simply has to click once to begin the first line, then click once again. This causes a line to be drawn from the first point to the second. Subsequent mouse-clicks cause a line to be drawn from the previously drawn point to the current one. As soon as four lines have been drawn, feedback is given. Alternatively, the problem-solver can stop the attempt at any stage by clicking on the 'Done' button. When the attempt is finished, in either of these fashions, all relevant feedback is given in the feedback area in the bottom of the application. When the feedback is displayed, the problem-solver is free to click on 'Start Again', which clears whatever lines were drawn, and the feedback area, and lets them start again. If they don't draw any lines within 45 seconds a message encouraging them to 'have a go' is presented. Because it is sometimes not easy to line up the point at which to click exactly, the problem-solver is able to move the end of a line within 10 pixels of its original position. This is achieved by clicking on the point and dragging it. A small circle is drawn showing the limits of where the point can be dragged to. When the feedback is given, certain words are highlighted to make it more clear. For example, the word 'not' is always hight-lighted, so when negative feedback such as "It is not correct to draw all your lines inside of the square..." the word 'not' is emphasised by making it bold and red.

The method of drawing lines in this system does not directly mirror that used when drawing with a pen or pencil. It is possible that the two-click action required here, rather than the more obvious click-and-drag technique, will mean that the rate of solution will be different from students solving the problem using this interface rather than pen and paper, as previous studies have used. In the experiment, there was no control group of problem-solvers attempting the problem without any feedback at all, but the two pilot studies (see Sections 3.1.1 and 3.1.4) showed at least that this problem is not easy for the population used in this study. Based on this it is believed that the computer interface to the problem did not somehow make it easier for those attempting it.

3.2.2 Analyser

The GUI provides a way for the problem solver to interact with the system; the Analyser looks at each attempt and returns whatever feedback is appropriate to the user interface. It is built around two data structures, the `Condition` and the `Attempt`, as described in the next section.

3.2.3 Data Structures

Two structures are required for the nine-dots system. There is a representation of an Attempt at the problem, and there is a representation of Conditions that require feedback to be given, which includes the associated feedback. An Attempt contains information about the attempt at solving the problem which various Conditions might depend upon.

Condition

A Condition[2] represents a situation in which the problem-solver will be given feedback by the system. There are four Conditions. When a new Attempt is made each condition is given the collection of all Attempts (including the new one). Each condition is responsible for evaluating these Attempts and updating the feedback as it sees fit. For this reason, each Condition must provide the following methods:

- +getHint():String
- +addFeedback(Controller controller):void
- +test(Object[] attempts):boolean
- +description():String

The *getHint* method must return the hint that is to be given to the user. The *addFeedback* method actually delivers the feedback to the user, via the Controller. The *test* method is the key method of a Condition — it must evaluate the Attempts and decide on whether or not any feedback is required. If it returns false, feedback will be given to the user. *description* returns a simple textual description of the Condition, and is used only for testing purposes.

[2]The classes in the system are actually called Constraints. The term Constraint, however, is a misnomer because it is now often used when referring to Constraint-Based Student Modeling (see Ohlsson (1993)) so the term Condition is used hereafter to avoid confusion.

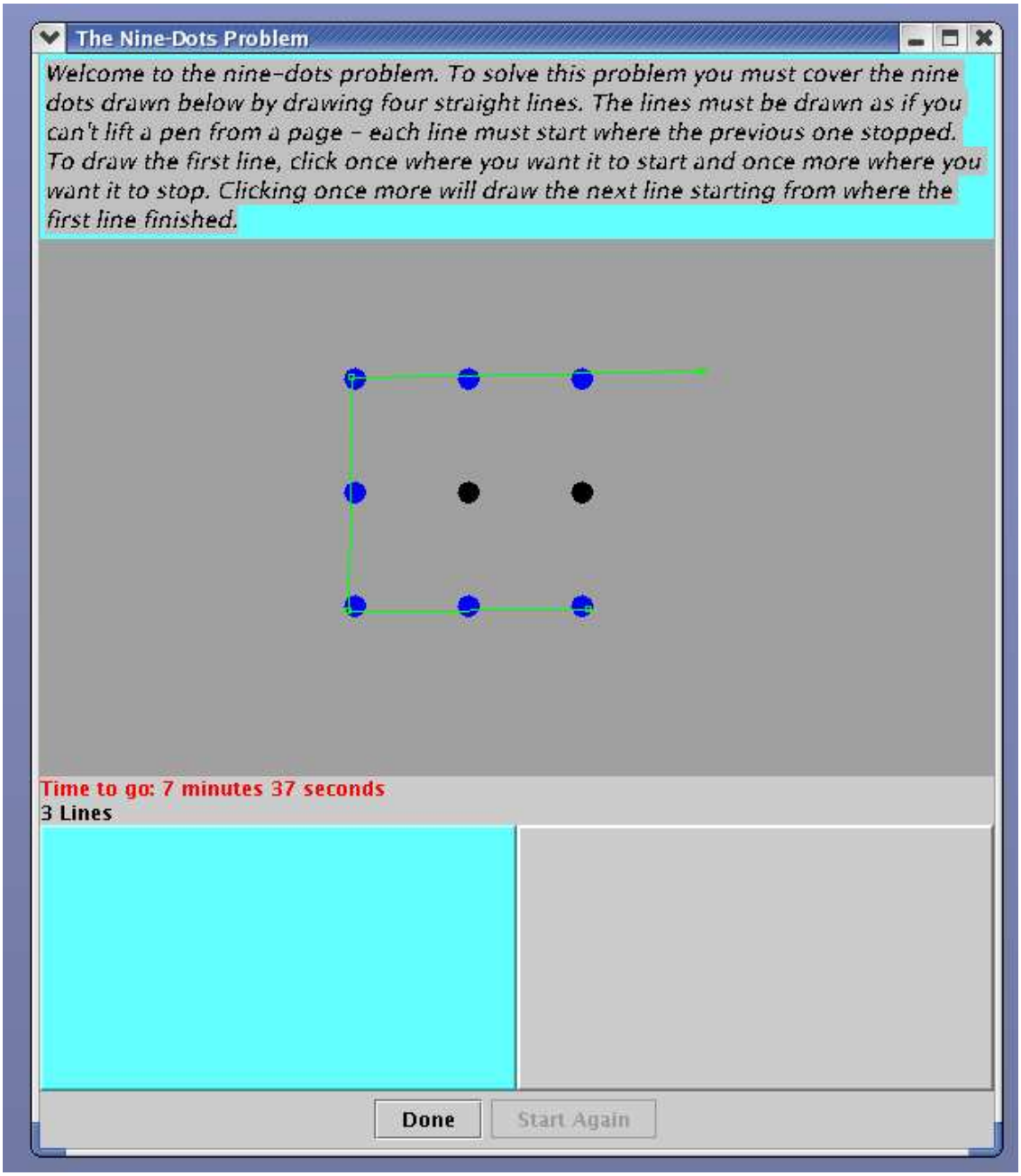

Figure 3.2: The nine-dots user interface

There are two versions of each Condition — a positive version and a negative version. All the methods are the same for these, except for the method which sets the feedback. When the Conditions are created at the start, either the positive version of each Condition or the negative version of each Condition is created. Then, whenever there is a new attempt, each Condition has its test method called, and whichever of these return false have their feedback messages returned to the user.

Attempt

An Attempt contains information such as whether any lines went outside the square formed by the dots, and how many 45 degree turns were made. It also contains a Pattern[3] which is a collection of PatternLines which represent a *standardised* (see section 3.1.3) version of the shape drawn. One of the Conditions is that if a shape is repeated by the problem-solver they will get a hint telling them to try something different, and so that this hint is still given even if they repeat the same shape but at a different rotation around the centre of the dots, each Pattern is standardised before it is stored. Standardising means rotating and/or reflecting the pattern, so that it starts from the top-left dot and the first line ends to the right of this position.

As mentioned above, a Pattern is a series of PatternLines. A Pattern also has a start position. Each PatternLine has a magnitude in the x direction and a magnitude in the y direction.

3.2.4 Log Files

To be able to analyse the way the problem-solvers attempted the nine-dots problem, all their actions had to be recorded. For this purpose, whenever a new attempt was made by the user, or whenever feedback was given by the system, this information was logged. This logging information also included the times at which these events happened, and recorded the success or failure of each participant to solve the problem. An example log file is given here:

[3] not to be confused with software design patterns.

```
version="false" start-time="1096339007651"
1096339039056:  Attempt:
starting at: java.awt.Point[x=201,y=360]
horizontal: 2 vertical: 0
horizontal: 0 vertical: -2

1096339039056: That is wrong. You have too many right-
angled turns in your solution. It is not correct to draw
all your lines inside of the square formed by the dots.
Do not do this next time. That is not the solution.

1096339103086:  Attempt:
starting at: java.awt.Point[x=81,y=363]
horizontal: -4 vertical: 0
horizontal: 3 vertical: -2

1096339103086: That is wrong. You do not have enough
45-degree turns in your solution.  That is not the
solution.

1096339164678:  Attempt:
starting at: java.awt.Point[x=199,y=218]
horizontal: 2 vertical: 0
horizontal: -2 vertical: -3
```

The 'false' at the start means that in this case the negative version of the system was used — all feedback was given in a negative fashion. All times are numbers of milliseconds since January 1, 1970, 00:00:00 GMT (as `Java Date` objects measure time). Each attempt has a start location, which is the pixel co-ordinates within the application window of the first line. The lines for each Attempt are as they are standardised (see section 3.1.3). The units used here are number of inter-dot-spacings:

```
horizontal: 2 vertical: 0
```

means that a horizontal line was drawn from left to right, and that this line was as long as twice the gap between two of the dots. The next part of the log file is the feedback that was generated in response to that attempt. The real log files were generally much longer than this, and also contained a line like:

```
1096339203864:  Finished Problem: Correct = true
```

which is self-explanatory.

Chapter 4

Experiment

There have been a number of previous studies of the nine-dots problem in which the aim of trying to improve the ability of people to solve it has been carried out by providing some form of training or feedback to help solve it. In these studies, the training or feedback is always information about what the problem-solver *should* do, and is never information regarding what they *should not* do. For example, Kershaw & Ohlsson (2001) provided training on how to make non-dot turns and to cross lines. This training is given by teaching how to solve variations of the nine-dots problem. The variations include extra dots, and still having nine dots but with one row or column displaced from the square.

The experiment described here compares providing feedback to people attempting the nine-dots problem in a positive way with providing it in a negative way. In this study, volunteers were assigned to versions of the software described in Chapter 3. Half were assigned the positive feedback version, the other half the negative. Volunteers were assigned the version of the system depending upon the order in which they arrived. Each volunteer attempted the problem at a different time, or in a separate room from the others. Each had the problem explained to them, were given a quick demonstration of the User Interface, and were given eight minutes to attempt to nine-dots problem. They could attempt the problem as many times as they liked within the eight minutes; the only limit was the time.

If a positive feedback format proves to be significantly better than a negative format for solving insight problems, it may also be more appropriate for other problems which have a creative element, so all teachers with students encountering this type of problem should make sure that they give any feedback in this way. If negative feedback is better, the opposite is true: all teachers of creative-type problems should make sure that they phrase their feedback in a negative manner.

40 volunteers were sought from three undergraduate Computer Science courses: COSC122, COSC221, COSC226. Students were persuaded to volunteer by the offer of a $5 coffee voucher to those who did. The attempts of four volunteers had to be discarded, due to them having been exposed to the nine-dots problem before, which left 18 volunteers in each group. No demographic data on the volunteers were collected.

4.1 Solution Rates

Table 4.1 shows the solution rates with the two forms of feedback. Note that previous studies have all shown a solution rate of near 0% when given a comparably short period of time to attempt the problem (MacGregor et al. 2001). Of the twelve using the positive version who solved it, three solved it after only one attempt and one on their first. It is unlikely that they solved it because of the feedback they received, especially given that the three who solved it after only one attempt all received different feedback on their first attempt. Because it is considered likely that these four did not solve because of the feedback, but may have been extraordinarily good at the nine-dots problem, or may have had previous exposure to the problem (which they all claimed they had not), results for positive are also given excluding these four. Even when they are included, an unpaired t-test with unequal variances shows there to be no significant difference between the rates of solution of the two groups.

Feedback type	Number of participants	Number completed successfully
Positive	18	12 (66%)
Positive > 1 attempt	14	8 (57%)
Negative	18	8 (44%)

Table 4.1: Solution Rates

4.2 Solution Speed

Table 4.2 shows the average speed of solution, for those who solved it within the alloted eight minutes. As above, results for positive feedback are given both including and excluding four problem-solvers who solved it outstandingly quickly. An unpaired t-test with unequal variances shows there to be no significant difference between the average times, even when the four outliers are included. When they are not included the average times for both positive and negative feedback are almost identical: 4 minutes 3 seconds and 4 minutes 5 seconds, respectively.

Feedback type	Number who solved	Average time to solve
Positive	12	2 min 34 sec
Positive > 1 attempt	8	4 min 03 sec
Negative	8	4 min 05 sec

Table 4.2: Solution Speed

4.3 Number of Attempts

Table 4.3 shows the average number of attempts at the problem made by those who successfully solved it, for the two groups. Again, the positive feedback group is split into all those who solved it, and only those who had more than one failed attempt before success. An unpaired t-test with unequal variances shows there to be no significant difference between the average number of attempts required.

Feedback type	Number who solved	Average number of attempts to solve
Positive	12	3.8
Positive $>$ 1 attempt	8	5.4
Negative	8	5.6

Table 4.3: Average Number of Attempts required by those who solved successfully

4.4 Feedback Messages

Usage of the feedback messages is analysed in this section. Whether the problem-solvers actually attended to the information given in the messages is first discussed (Section 4.4.1), then the solution rates are investigated to judge whether any particular messages were more useful than others (Section 4.4.2).

4.4.1 Using Feedback Information

The interest in this project is whether people comprehend and use the information in the feedback messages when they are given in different styles. A measure of how well the messages are understood is how many times a problem-solver gets the same feedback message in short secession. Getting the same message twice in a row means that they ignored the information in it the first time, and repeated their error. Table 4.4 shows the percentage of times a feedback message is repeated in the positive feedback version across all participants. Table 4.5 shows the same information for those receiving positive feedback who successfully solved the problem. Table 4.6 shows the percentages for all problem-solvers receiving negative feedback, and Table 4.7 shows it for those who received negative feedback and correctly solved the problem. The leftmost column of these tables gives the condition for the feedback being given; the second column the number of times it was given immediately after the same message; the third column the number of times it was given two attempts (but not one) after the same message; the fourth column gives the number of times that message appeared altogether and the fifth column shows the percentage of times that the message is repeated, the first value being repeated immediately and the second being repeated with exactly one attempt's gap.

The high number of times that some feedback messages were repeated shows that sequential attempts were seldom disjointed; there was often at least some similarity between two consecutive attempts. The number of times that mistakes were repeated is larger for negative feedback than it

Feedback type	Repeat next time	Repeat time after next	Total times encountered	Percentages
Right angles	7	8	37	20% & 21%
Square	7	1	22	32% & 5%
45 Degree	39	20	82	48% & 24%
Repeat Pattern	3	0	31	10% & 0%

Table 4.4: Positive Feedback Repeat Rates For All Participants

Feedback type	Repeat next time	Repeat time after next	Total times encountered	Percentages
Right angles	2	1	12	17% & 8%
Square	4	0	12	33% & 0%
45 Degree	7	3	21	33% & 14%
Repeat Pattern	3	0	9	33% & 0%

Table 4.5: Positive Feedback Repeat Rates For Solved Successfully

Feedback type	Repeat next time	Repeat time after next	Total times encountered	Percentages
Right angles	44	22	101	44% & 22%
Square	115	6	137	84% & 4%
45 Degree	35	15	82	43% & 18%
Repeat Pattern	9	9	43	21% & 21%

Table 4.6: Negative Feedback Repeat Rates For All Participants

Feedback type	Repeat next time	Repeat time after next	Total times encountered	Percentages
Right angles	10	3	19	53% & 16%
Square	10	0	16	63% & 0%
45 Degree	5	0	14	36% & 0%
Repeat Pattern	0	0	3	0% & 0%

Table 4.7: Negative Feedback Repeat Rates For Solved Successfully

is for positive feedback. For the 'no right angles' feedback message, 44% of times it was encountered in the negative feedback version, it was given again after the next attempt as well. This contrasts with only 20% in the positive feedback version. In both cases, the message was repeated around 20% (22% for negative and 21% for positive) further times two attempts later (but not on the immediately subsequent one). For the 'outside the square' hint and the 'repeated pattern' hints, a similarly disproportionate number of problem-solvers repeated their error within the next two

attempts of those who received negative feedback compared with those who received positive feedback. The only instance in which the two versions of feedback have similar rates of repeat is with the 'three 45 degree turns' message. In this case the error is repeated slightly more often when the message is given in a positive fashion than in a negative one, but the difference is not significant. There is no case in which negative feedback is markedly more successful than positive. The outstanding figure in this data is that 84% of the time a problem-solver received the negative hint "It is not correct to draw all your lines inside of the square formed by the dots. Do not do this next time" they again drew their lines only inside the square. This inexorable repetition of lines only inside the square is considerably more pronounced than when the message is given in the positive fashion ("You need to draw lines outside of the square formed by the dots in order to solve this problem. Try this next time"). Why those receiving negative feedback were so strongly disinclined to heed the message is unclear.

4.4.2 Most Useful Messages

Table 4.8 shows the number of times that each feedback message was received as the last message before the problem was solved.

Feedback type	No right angles	Square	45 Degree angles	Repeated pattern
Negative	0 (0%)	0 (0%)	6 (86%)	1 (14%)
Positive	1 (9%)	4 (36%)	7 (64%)	0 (0%)
Total	1 (5%)	4 (21%)	13 (68%)	1(5%)

Table 4.8: Last feedback message before correct solution.

The feedback message regarding 45 degree angles was received immediately before solution a disproportionate number of times. Overall, it was given 164 times out of a total of 352 attempts at the problem (47%), but it was given in the penultimate attempt before solution 13 of 19 times (68%). It was the last message more often in the negative feedback version than in the positive.

Tables 4.9 and 4.10 respectively show the percentage of problem-solvers who received each feedback message at least once, for those who solved it successfully, and those who did not. The average number of attempts to solve was lower for those receiving positive feedback (see table 4.3), but of these, the average percentage receiving each message at least once is similar. For those who did not solve the problem successfully, those receiving negative feedback were more likely to see all types of feedback message, but again there is not much difference between the two.

Feedback type	No right angles	Square	45 Degree angles	Repeated pattern
Negative	0.63	0.75	0.75	0.38
Positive	0.55	0.64	0.73	0.36

Table 4.9: Proportion of those who solved who received each feedback message.

Feedback type	No right angles	Square	45 Degree angles	Repeated pattern
Negative	1.00	1.00	1.00	0.90
Positive	0.94	0.72	0.94	0.89

Table 4.10: Proportion of those who did not solve who received each feedback message.

These two analyses show that there is no single feedback message that makes it significantly more likely (or unlikely) for the problem-solver to successfully complete the nine-dots problem. Of those who solved correctly, approximately the same proportion saw each of the feedback messages at least once with both forms of feedback, so it does not appear as if a certain message proved to be more or less necessary in either format. The feedback message regarding 45 degree turns was received by almost every participant, but it often appeared as the last message before solution. It is concluded that this message, especially in the negative form, proved to be the necessary 'tipping point' for those who were set to solve it. That is not to say that if you receive it you will solve the problem; it is to say that if you do not solve the problem after receiving it several times you are unlikely to do so.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

Insight problems, specifically the nine-dots problem, have been discussed. Positive and negative feedback have been discussed. A software system that allows problem-solvers to attempt the nine-dots problem, and receive feedback either in positive or negative fashion was presented, and an experiment using this system to measure the impact of these different feedback formats was described.

Giving feedback in either a positive or a negative fashion has no significant effect on the rate of solution of people attempting the nine-dots problem. Feedback given in either a positive or a negative manner can increase the rate of solution from around 0% to around 50%. There are no single feedback messages, of those that were given, which are either necessary or sufficient to solve the nine-dots problem. Problem-solvers who receive feedback in a positive style are more likely to apply the information contained in that message than if it were phrased in a negative manner, however. It is therefore concluded that delivering feedback in a positive style to people attempting to solve the nine-dots problem is superior than delivering it in a negative manner. This conclusion may extend to all insight problem solving, or education systems in general.

If positive or negative feedback had significantly altered the likelihood of the solution to the nine-dots problem being found, conclusions relating to the original mental structure of people's solution tree could have been drawn. If negative feedback had proved to be significantly better it could have been concluded that people can already cognate the solution when they begin the problem; if positive feedback had proved to be significantly better it could have been concluded that people did not originally have the solution available but generated it on the basis of the feedback. Because neither proved to be significantly better than the other, no conclusions regarding this can be drawn.

5.2 Future Work

This project has investigated different forms of feedback for solving the nine-dots problem, but has also uncovered many areas for future investigation. Some of these are detailed here:

5.2.1 Intelligent Tutoring System for Insight Problem Solving

As discussed in section 3.1, the original goal of this project was to create an Intelligent Tutoring System (ITS) for insight problem solving in general. This goal was abandoned due to time constraints, but is believed to still be achievable. Future work could include another attempt at this. Weisberg (1986) argues, based upon experiments reported in Hayes (1981), that creativity is something which can be improved, and that creative acts normally require practise and study before they can occur. His argument regards learning about a field before being able to think creatively in that field, as opposed to learning how to be creative, but it does support the idea that a creative ability is not inherent from birth, but can actually be developed in some way, which supports the case that an ITS for insight problem solving could be created.

5.2.2 Mouse-movement Studies

It was observed that when thinking while attempting the nine-dots problem, some participants traced possible solutions using the mouse pointer. The path of these traces could provide information about the way that the user is thinking about the problem. Eye-movement studies of people attempting insight problems have already been used to test theories of insight and evaluate impasse (Knoblich et al. 2001); similar studies for other problems could be carried out by analysing mouse movements rather than eye movements.

5.2.3 Positive and Negative Feedback in other domains

This study showed that problem-solvers are more likely to follow the advice of feedback when it is given in a positive style than when it is given in a negative style, but focused only on the nine-dots problem. Previous studies have shown the danger of giving negative feedback sometimes (Geddes & Baron 1997) (the authors talk about workplace aggression as a result of negative feedback!), but there are many more problem domains where the differing forms of feedback could be compared.

Acknowledgements

Thanks to everyone who participated in either of the pilot studies, or the experiment. Thanks also to Dave, Emma, Frances and Tony for proof-reading this report, and providing valuable feedback (both positive and negative).

Bibliography

Akin, O. & Akin, C. (1996), 'Frames of reference in architectural design: analysing the hyperacclamation (a-h-a!)', *Design Studies* **17**, 341–361.

Anderson, J. R., Corbett, A. T., Koedinger, K. R. & Pelletier, R. (1995), 'Cognitive tutors: Lessons learned', *The Journal Of The Learning Sciences* **4**(2).

Anderson, J. & Skwarecki, E. (1986), 'The automated tutoring of introductory computer programming', *Communications of the ACM* **29**(9), 842–849.

Auble, P. M., Franks, J. J. & Salvatore A. Soraci, J. (1979), 'Effort toward comprehension: Elaboration or "aha!"?', *Memory and Cognition* **7**, 426–434.

Bloom, B. S. (1984), 'The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring', *Educational Researcher* **13**, 4–16.

Chronicle, E. P., MacGregor, J. N. & Ormerod, T. C. (2004), 'What makes an insight problem? the roles of heuristics, goal conception, and solution recoding in knowledge-lean problems', *Journal of Experimental Psychology: Learning, Memory and Cognition* **30**(1), 14–27.

Davidson, J. E. (1995), The suddenness of insight, *in* R. J. Sternberg & J. E. Davidson, eds, 'The nature of insight', The Massachusetts Institute of Technology Press, pp. 125–155.

Elster, J. (2000), *Ulysses Unbound*, Cambridge University Press.

Geddes, D. & Baron, R. A. (1997), 'Workplace aggression as a consequence of negative performance feedback', *Management Communication Quarterly* **10**(4), 433–455.

Goshi, K., Wray, P. & Sun, Y. (2001), An intelligent tutoring system for teaching and learning Hoare logic, *in* 'Proceedings of the fourth international conference on Computational Intelligence and Multimedia Applications', pp. 293–297.

Gruber, H. E. (1995), Insight and affect in the history of science, *in* R. J. Sternberg & J. E. Davidson, eds, 'The nature of insight', The Massachusetts Institute of Technology Press, pp. 397–432.

Hayes, J. R. (1981), *The Complete Problem Solver*, The Franklin Institute Press.

Ippolito, M. F. & Tweney, R. D. (1995), The inception of insight, *in* R. J. Sternberg & J. E. Davidson, eds, 'The nature of insight', The Massachusetts Institute of Technology Press, pp. 433–462.

Joyce, J. (1939), *Finnegan's Wake*, Faber.

Kershaw, T. C. & Ohlsson, S. (2004), 'Multiple causes of difficulty in insight: The case of the nine-dot problem', *Journal of Experimental Psychology: Learning, Memory and Cognition* **30**(1), 3–13.

Kershaw, T. & Ohlsson, S. (2001), Training for insight: The case of the nine-dot problem, *in* J. Moore & K. Stennings, eds, 'Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society', pp. 489–493.

Knoblich, G., Ohlsson, S. & Raney, G. E. (2001), 'An eye movement study of insight problem solving', *Memory and Cognition* **29**(7), 1000–1009.

MacGregor, J. N., Ormerod, T. C. & Chronicle, E. P. (2001), 'Information processing and insight: A process model of performance on the nine-dot and related problems', *Journal of Experimental Psychology: Learning, Memory and Cognition* **27**(1), 176–201.

Maier, N. R. F. (1930), 'Reasoning in humans: I. on diretion.', *Journal of comparitive psychology.*

Maier, N. R. F. (1931), 'Reasoning in humans: II. the solution of a problem and its appearance in consciousness.', *Journal of comparitive psychology.*

Maier, N. R. F. & Casselman, G. G. (1970*a*), 'Locating the difficulty in insight problems: individual and sex differences', *Psychological Reports* **26**, 103–117.

Maier, N. R. F. & Casselman, G. G. (1970*b*), 'The SAT as a measure of problem-solving ability in males and females', *Psychological Reports* **26**, 927–939.

Maier, N. R. F. & Janzen, J. C. (1969), 'Are problem-solvers also creative?', *Psychological Reports* **24**, 139–146.

Mitrovic, A., Martin, B. & Mayo, M. (2002), 'Using evaluation to shape its design: Results and experiences with sql-tutor', *User Modeling and User-Adapted Interaction* **12**(2-3), 243–279.

Ohlsson, S. (1993), 'Constraint-based student modeling', *Journal of Artificial Intelligence in Education* **3**(4), 429.

Ormerod, T. C., MacGregor, J. N. & Chronicle, E. P. (2002), 'Dynmaics and constraints in insight problem solving', *Journal of Experimental Psychology: Learning, Memory, and Cognition* **28**(4), 791–799.

Trottier, L. (2003), 'The current state of insight research', *Canadian Undergraduate Journal of Cognitive Science* .

Weisberg, R. (1986), *Creativity: Genius and other myths*, W. H. Freeman and Company.

Weisberg, R. W. (1995), Prolegomena to theories of insight in problem solving: A taxonomy of problems, *in* R. J. Sternberg & J. E. Davidson, eds, 'The nature of insight', The Massachusetts Institute of Technology Press, pp. 157–196.

Weisberg, R. W. & Alba, J. W. (1981), An examination of the alleged role of "fixation" in the solution of several "insight" problems, *in* 'Journal of Experimental Psychology', pp. 169–192.