An Elaboration Account of Insight

Christopher MacLellan
Computing Science and Engineering
Arizona State University, Tempe, AZ 85287

Abstract

In this paper we discuss an elaboration account of insight that provides answers to the two main questions regarding insight problem solving: why insight problems are so difficult for humans and why insight is so rapid in nature. We claim that the difficulty in insight problems is due to misguided heuristic search and that this difficulty is overcome using a reformulation mechanism. Furthermore, we claim that search is carried out quickly when the heuristics are good–explaining the rapid nature of insight. We clarify our account by providing examples and initial empirical results. In conclusion, we review related work and discuss possible future work.

Introduction

Insight phenomena, or discovering solutions to problems in flashes of insight, are an interesting occurrence that has been studied by scholars for over a century (Hadamard 1945; Poincarè 1952). These phenomena have drawn great attention because they have resulted in some of the most important scientific discoveries throughout history–discoveries such as Archimedes' Principle and Einstein's Special Theory of Relativity (Einstein 1982). Despite this attention, the mechanisms underlying insight are still not well understood.

To study these phenomena in a controlled environment psychologists have given insight problems to human subjects and observed their behavior. Insight problems are simple puzzles which require some trick or insight to solve. Without the proper insight the problems are impossibly difficult. But, when armed with the right insight the problems are so trivial that solutions are often found in a flash of realization–resembling discoveries in actual insight phenomena. Psychologists have studied this class of problems in an attempt to determine why they are so difficult for humans and why solutions, when they are found, come so rapidly (Isaak and Just 1996; Knoblich et al. 1999; Knoblich, Ohlsson, and Raney 2001; MacGregor, Omerod, and Chronicle 2001; Omerod, MacGregor, and Chronicle 2002; Jones 2003; Chronicle, MacGregor, and Omerod 2004; Segal 2004; Langley and Jones 1988; Ohlsson 1984b; 1984a; Kershaw and Ohlsson 2004).

In this paper we introduce an elaboration account of insight problem solving which provides answers to these questions via two mechanisms, weak heuristic search and reformulation. We begin the paper by reviewing two predominant theories of insight problems solving which lay the foundation for our work. Next, we introduce our theory and present its implementation in a computational system. After this, we present examples and results of our system running on a simple non-insight blocks world problem and the nine dots insight problem. We qualitatively analyze these results and show how they support our claims. We conclude the paper by presenting related work and directions for future research.

Background

In this section we briefly review the background material relevant to our theory. First, we give a more detailed definition of insight problems, accompanied with an illustrative example. Second, we present two predominant theories of insight which explain how humans may handle such problems. The key elements of these theories, reformulation and greedy heuristic search, lay the foundation for our elaboration account of insight.

Insight Problems

Insight problems are commonly defined as problems which are nearly impossible to solve without some crucial insight or reformulation of the problem. This insight most often takes the form of additional domain operators that were not considered, awareness of constraints that are improperly self-imposed, or incorrect default assumptions that are corrected. Once a solver has discovered the proper insight, the solution to the problem is rapidly discovered–in what most humans describe as a flash.

One famous insight problem that has often been cited in the literature is the nine dots problem. This problem and its solution are shown in Figure 1. The objective is to connect all nine dots by drawing four consecutive straight lines without lifting the pen from the paper and never retracing a line. When humans are first presented with this problem they almost always make the assumption that they must not go outside the box. This assumption makes a solution impossible. At some point the problem solver usually realizes

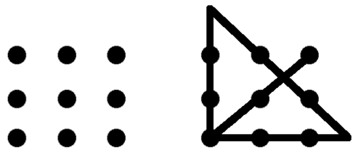

Figure 1: The classical nine dots problem and its solution.

this is an incorrect assumption and in a flash of insight a solution becomes trivial.

The nine dots problem is just one of many insight problems that have been studied in the literature.[1] These problems are a mystery that has both intrigued and baffled scholars. If an insight problem is eventually solved, then we know that the solver has the capacity for a solution. What is the source of difficulty in finding a solution and why, when a solution is found, does it come so rapidly? These are questions that any theory of insight should answer.

Representational Change Theory

Representational change theory (Knoblich et al. 1999) extends from the work of Gestalt psychologists who believed that the mind uses structures to represent every situation and that these structures could be reformulated. In this theory it is believed that one encounters impasses, not knowing what to do, when attempting to solve an insight problem. Representational change theory posits that these impasses are overcome by reformulation of the problem representation. Knoblich, Ohlsson, Haider, and Rhenius present two hypothetical mechanisms to facilitate this reformulation: relaxation of constraints and decomposition of perceptual chunks. Since previous experience is the source of constraints and perceptual chunks, they posit that the difficulty in insight problem solving is due to the challenge of overriding past experience when faced with new situations. Their arguments for reformulation are convincing and have been supported experimentally (Knoblich, Ohlsson, and Raney 2001) but they fail to offer a convincing story of why solutions come so rapidly once insight is discovered, simply claiming that a proper reformulation leads to “purposeful and swift goal attainment.”

Criterion of Satisfactory Progress Theory

The criterion of satisfactory progress theory, presented by Chronicle, Ormerod, and MacGregor (2001), explains the difficulty in insight problem solving as a result of search with maximizing and progress monitoring heuristics. The criterion of satisfactory progress theory posits that when solving an insight problem, only actions which make adequate progress towards the goal are considered. From this set of adequate states, the best states are expanded and the remainder are marked as promising states for later review. Due to the greedy nature of this search, the solver can get stuck, resulting in an impasse. When this happens it is theorized that the promising states are then evaluated. Once all of the promising states have been expanded the process is repeated with a look-ahead. Upon failure, this process is repeated with deeper and deeper look-ahead until the solver eventually finds a solution. In the case where the heuristics correctly guide the search, the solution is found quickly.

Experimental evidence supports the theory that the difficulty in insight problem solving is due to maximizing and progress monitoring heuristics (MacGregor, Omerod, and Chronicle 2001; Omerod, MacGregor, and Chronicle 2002; Chronicle, MacGregor, and Omerod 2004). This theory may also explain why insight comes so quickly, due to the rapid nature of heuristic search. The weakness of this theory lies in its ambiguity regarding how impasses are overcome. Promising states and look-ahead are suggested but these mechanisms are not studied in the context of impasse resolution.

A Novel Account of Insight

Elements of both the representational change theory and the criterion of satisfactory progress theory have been corroborated as mechanisms of insight in human studies (Knoblich, Ohlsson, and Raney 2001; MacGregor, Omerod, and Chronicle 2001; Omerod, MacGregor, and Chronicle 2002; Chronicle, MacGregor, and Omerod 2004) but each theory fails to tell a complete story about insight problem solving. Representational change theory does not account for the rapid nature of finding a solution once insight is discovered. Conversely, the criterion of satisfactory progress theory fails to adequately explain how one overcomes impasses in search. By combining elements of each theory we can present a more complete story of insight problem solving. This new theory, an elaboration account of insight, makes three explicit claims about the insight problem solving process:

1. The primary difficulty in insight problem solving is due to misguided heuristic search–resulting in impasses.
2. Impasses are overcome by changing the search heuristic via a reformulation mechanism.
3. Fast search, due to good heuristics, results in the rapid discovery of a solution.

Together these claims tell a story about insight phenomena and serve as the main contribution of our paper. In the next section, we present a computational system for solving insight problems that is consistent with these claims. This system will use iterative sampling and goal elaboration to facilitate solutions to insight problems.

After presenting our implementation we present examples and experimental results of our system running on a blocks world problem and the nine dots problem. These initial results are positive and offer support for our claims. We show that misguided heuristic search is the primary difficulty in solving the nine dots problem. Furthermore, our results show that reformulation enables our system to overcome this difficulty. Lastly, we show that the problem, once reformulated, is qualitatively as easy as a simple blocks world problem–suggesting a rapid solution. It is important to note that we make no claims regarding exactly which search and

[1] See (Isaak and Just 1996) for a complete list.

reformulation mechanisms are used by humans. Additionally, we believe that other implementations of weak search and reformulation, besides the ones we use, may also suffice.

Implementation

Our implementation of an elaboration account of insight utilizes a modified version of iterative sampling (Langley 1992) as the weak search mechanism and goal elaboration with Answer Set Programming (Gelfond 2007) as the reformulation mechanism. These mechanisms, outlined in the following two subsections, require relational formulations of any problems we wish to solve.

Together these mechanisms form a complete system that is capable of solving both insight and non-insight problems. Given an insight problem, our system will first use modified iterative sampling, a greedy non-deterministic search, to attempt a solution. Since the goals of insight problems are only minimally specified, the domain independent heuristics, which are derived from the goal state, initially misguide search. Since the search is misguided, a solution will be almost impossible as the heuristics will consistently recommend the wrong actions. When the system reaches a certain number of failed attempts, goal elaboration with answer set programming is employed to infer a more complete goal statement. When the goal is reformulated and heuristic estimates are improved, a solution comes quickly–hence the rapid feeling of insight.

Weak Search Mechanism

For our weak search mechanism we implemented a basic forward chaining planner in the Lisp programming language. This planner accepts problems in a STRIPS (Fikes and Nilsson 1971) like representation and uses modified iterative sampling as its search technique. Additionally, the system uses a domain independent heuristic to guide search and accepts a maximum plan length which limits the depth of the search.

Iterative Sampling This is a non-deterministic search technique that uniformly samples actions at each state. When the maximum search depth is reached or a repeated state is detected this technique simply gives up and restarts search from the root node. Unlike depth-first search, backtracking is never performed. This search technique is similar to progressive deepening, a technique de Groot (1978) has observed in human chess players.

We use a modified version of this search method which uses heuristics to aid the search. Instead of uniformly sampling the actions, our version samples from the heuristically weighted set of actions. Since the probability of selecting an action is proportional to its heuristic estimate, those actions which are estimated to be better are more likely to be chosen.

Our implementation of iterative sampling tries 15 attempts at solving a given problem before reporting a failure. If in the course of these 15 attempts a solution is discovered, it is immediately returned.

Search Heuristic The heuristic used by our system is the number of goal literals satisfied in a given state. A boost is applied to the heuristic when the number of literals satisfied is greater than those satisfied in the current state, encouraging progress. This domain independent heuristic is derived from the goal literals so the heuristic landscape will change when the goal state is changed.

Reformulation Mechanism

For our reformulation mechanism we implemented goal elaboration, a technique which elaborates a given goal state. We chose to use goal elaboration because our search heuristic is based entirely on the goal. By changing the goal state the heuristic landscape of the search is modified. To conduct this goal elaboration we use Answer Set Programming, a logic programming language similar to Prolog. Our implementation is capable of finding fully specified goal states for a given problem.[2]

Goal Elaboration The purpose of goal elaboration is to turn a partially specified goal state into a more fully specified goal state. We conduct goal elaboration by finding goal literals that can be inferred from the initial and goal states. An abstract example can be given in the blocks world domain:

Initial State:	((block A) (block B) (block C) (on-table A) (on-table B) (on-table C))
Goal State:	((on A B))

From this problem we might infer the additional goal literals (on-table B) or (on B C), since B must be on something. Both elaborations in this example are correct and each will lead the heuristics to recommend different actions to achieve the goal.

Answer Set Programming To conduct goal elaboration we used Answer Set Programming, a logic programming language which finds the answer set for a given logic program. An answer set is a set of all minimal models that are consistent with the rules of a given logic program. Thus, for each problem we wished to reformulate, we describe the problem in logic rules (including the initial and goal literals) and use answer set programming to find all minimal models consistent with these rules. Each minimal model corresponds to an elaborated goal. Answer Set Programming has an option to return the first model that it finds, halting further search. In our implementation we utilize this option, as we only need one elaboration.

If Answer Set Programming fails to find any models, then a solution to the problem is impossible. In the event that this happens we believe Answer Set Programming can use generative rules, rules which generate literals, to infer additional literals for the initial state. If the correct literals can be inferred this should enabling a solution. The details of

[2]It is not crucial that we find all elaborations for a problem, that these elaborations are fully specified, or even that they are logically valid. In fact, we believe exhaustively finding elaborations is an excessive approach. Goal elaboration with answer set programming simply provides one mechanism by which goals could be reformulated.

this generative ability have not yet been fully worked out. To simulate this functionality we have wrapped Answer Set Programming with a program that adds user defined literals to the problems initial state in the event that no models are found.

Examples and Experimental Results

In this section we present example runs of our system on two different problems. Additionally, we provide experimental evidence which supports our claims. We show that the difficulty in solving the nine dots insight problem lies in misguided heuristic search. Furthermore, we show that this difficulty can be eliminated by elaborating the goal. Finally, we show that performance on the problem, once it has been reformulated, is equivalent to that of a simple blocks world problem–a problem humans are capable of solving almost instantly. This suggests that the rapid nature of insight is due to heuristically guided search.

First, we present a non-insight blocks world problem. We show our representation of the problem and an example of how our system finds a solution. Next, we present the nine dots insight problem. We show our representation of the problem and show examples of how our system finds a solution to a version of this problem which has been modified to enable a solution. After this, we present results from the system running on the original and modified nine dots problem with various configurations. We finish this section by discussing the presented examples and how they support our claims about the insight process.

Non-Insight Blocks World Problem

To provide an illustrative example, we show our system solving a simple problem from the well known blocks world domain. Since the problem is not an insight problem, reformulation is not required. Thus, we only show an example of the system solving the problem with the original goal formulation.

Representation For our blocks world problem we used the common representation. The problem, shown in Table 1, consists of four relations and four actions. The four relations featured in our problem are:

- (on ?x ?y), describing that a block ?x is on a block ?y.
- (on-table ?x), meaning that a block ?x is on the table.
- (holding ?x), representing that the agent is holding the block ?x.
- (block ?x), signifying that ?x is a block.

The goal of the problem is to find a plan to move three blocks sitting on the table into a tower of three blocks. This simple task is one that most humans will immediately know the answer to. Later, we will show that the performance on this task is qualitatively equivalent to the performance on the nine dots insight problem after it has been reformulated, giving some measure of how quickly one finds a solution to an insight problem once a proper reformulation is achieved.

Table 1: A non-insight blocks world problem.

Max Plan Length:	50
Initial State:	((block A) (block B) (block C) (on-table A) (on-table B) (on-table C))
Goal State:	((on A B) (on B C) (on-table C))
Action:	(lift ?x)
Preconditions:	((block ?x) (not (on ?any ?x)) (on ?x ?y) (not (holding ?any2)))
Effects:	((not (on ?x ?y)) (holding ?x))
Action:	(lift-from-table ?x)
Preconditions:	((block ?x) (not (on ?any ?x)) (on-table ?x) (not (holding ?any2)))
Effects:	((not (on-table ?x)) (holding ?x))
Action:	(stack ?x ?y)
Preconditions:	((block ?x) (block ?y) (holding ?x) (not (on ?any ?y)))
Effects:	((not (holding ?x)) (on ?x ?y))
Action:	(place-on-table ?x)
Preconditions:	((block ?x) (holding ?x))
Effects:	((not (holding ?x)) (on-table ?x))

Example Shown in Table 2 is the trace of our system solving the blocks world problem.[3] Our system starts off in the initial state (step 1) with an empty plan. It then samples an action from the heuristically weighted set of available actions. In step 2 action (lift-from-table A) is selected. This is the incorrect action (B should be placed on C before A is placed on B) but our system is unaware of this and continues search anyway. Next, in steps 3 to 5 the sample and select action sequence is repeated. Finally, in step 5, after the action is selected, the system realizes that it is in a state that it has previously visited. This causes the planner to restart, as shown in step 6. In steps 7 through 10 the system again follows the sample and select sequence. Upon selecting the (stack A B) action in step 10 the planner realizes that it has achieved the goal, causing the system to terminate and return the current plan.

As shown in Table 7, the system had a 100% success rate on the blocks world problem and averaged 1.13 failed attempts before finding a solution. These numbers provide a measure of performance on an easy task. Later we will compare these results with those on the modified nine dots problem with a reformulated goal.

Nine Dots Insight Problem

To support the claims of our theory we give example traces and experimental results of our system solving the nine dots insight problem. We show experimental results of running

[3]In this trace, and all those following, chosen actions are pushed onto the front of the plan, so the plans shown in the table are in reverse order.

Table 2: A solution trace of our system solving the blocks world problem.

Step #	Current Plan
1	()
2	((lift-from-table a))
3	((stack a b) (lift-from-table a))
4	((lift-from-table c) (stack a b) (lift-from-table a))
5	((place-on-table c) (lift-from-table c) (stack a b) (lift-from-table a)) **Failure, repeated state detected**
6	()
7	((lift-from-table b))
8	((stack b c) (lift-from-table b))
9	((lift-from-table a) (stack b c) (lift-from-table b))
10	((stack a b) (lift-from-table a) (stack b c) (lift-from-table b)) **Goal achieved**

our system in three configurations on two versions of the problem.

Representation There were multiple levels of abstraction at which we could have represented the nine dots problem. We chose a high-level representation that still allowed us to perform sufficient reformulation. The original problem, shown in Table 3, consists of six relations and five actions. The six relations from our problem are:

- (dot ?x), signifying that ?x is a dot.
- (three-adj ?x ?y ?z), representing that ?x, ?y, and ?z are adjacent to each other and in a row.
- (crossed ?x), meaning that ?x has been crossed by a line.
- (started-at ?x), used to signify the starting location that the pen was initially placed at.
- (line ?x ?y), showing that there is a line from ?x to ?y.
- (pen up), signaling that the pen has not yet been placed.

The problem is also given a max plan length of five actions.[4]

This original formulation of our problem is unsolvable because two additional dots outside of the original nine are needed for a solution. For this reason we also include a modified version of the nine dots problem which has the two necessary dots (d10 and d11) and relations for their orientation. We have shown a graphical representation of the dot orientation and numbering in Figure 2, which also shows the locations of the additional dots.

Our reformulation mechanism enables a solution to the original formulation when given the correct literals to inject into the problem. The literals given to the system are those for the two necessary dots. When Answer Set Programming encounters the original formulation it is unable to find any

[4]A plan length of five makes sense in the nine dots problem because it is part of the problem description. In other insight problems other depths may be preferable.

10			
3	6	9	
2	5	8	
1	4	7	11

Figure 2: Numerical value of the dots in our nine dots formulation.

models. This triggers the user defined wrapper to inject one of the two necessary dots into the formulation. Since a solution is still impossible with only one additional dot, Answer Set programming is again unable to find any models. This triggers the user defined wrapper to inject the second necessary dot, enabling Answer Set Programming to find a goal elaboration. This elaboration is then returned along with the new modified problem to the solver.

Example With Original Goal Shown in Table 4 is a trace from our system solving the modified nine dots problem with the original goal and no reformulation. In this example the system starts off incorrectly by choosing to place the pen at dot 5 (in step 2), an unsolvable situation. At step 6, when only dot 6 remains uncrossed, the system fails because the max plan length has been reached. The second attempt starts off correctly but again fails to find a solution. This pattern repeats for an additional 13 failed attempts before the system finally reports that it failed to find a solution. Since the system is not using reformulation in this example, no further work is performed. In this example the planner experienced difficulty due to the poor search heuristics which failed to consistently guide the planner to the correct actions. This inability to consistently choose the correct action resulted in the non-deterministic equivalent of a problem solving impasse.

As shown in Table 7, the system had only a 31% success rate on the modified version of the nine dots problem with the original goal and no reformulation.[5] For the 31 successful runs the system averaged 7.32 failed attempts before finding a solution.[6]

Example With Reformulated Goal When the system fails to find a plan, as in the previous example, a reformulated goal is constructed. This is done by employing Answer Set Programming. Answer Set Programming returns the first elaboration it finds, the fully elaborated goal state shown in Table 5. Our system will then uses this elaborated goal and try to solve the nine dots problem again. A resulting trace is shown in Table 6. Now that the goal has been reformulated the heuristics are much more informative, resulting in a rapid solution. Steps 1 through 6 show the planner immediately selecting the correct actions to achieve the goal.

As shown in Table 7, the system had a 100% success rate on the modified version of the nine dots problem with the

[5]If the number of attempts allowed for each search trial is increased this success rate will improve but it will have substantially more failed attempts and take much longer.

[6]This number is generated from the trials that were successful. If you account for the 69 failed attempts this number would be even higher.

Table 3: The original nine dots insight problem.

Max Plan Length:	5
Initial State:	((dot d1) · · · (dot d9) (three-adj d4 d5 d6) · · · (three-adj d1 d2 d3))
Goal State:	((crossed d1) · · · (crossed d9))
Action:	(place-pen ?x)
Preconditions:	((dot ?x) (pen up))
Effects:	((started-at ?x) (at ?x) (crossed ?x) (not (pen up)))
Action:	(draw-line ?x ?y)
Preconditions:	((dot ?x) (dot ?y) (at ?x) (three-adj ?x ?y ?z))
Effects:	((not (at ?x)) (at ?y) (crossed ?x) (crossed ?y) (line ?x ?y))
Action:	(draw-line ?x ?y)
Preconditions:	((dot ?x) (dot ?y) (at ?x) (three-adj ?z ?x ?y))
Effects:	((not (at ?x)) (at ?y) (crossed ?x) (crossed ?y) (line ?x ?y))
Action:	(draw-line ?x ?y)
Preconditions:	((dot ?x) (dot ?y) (dot ?z) (at ?x) (three-adj ?x ?z ?y))
Effects:	((not (at ?x)) (at ?y) (crossed ?x) (crossed ?y) (crossed ?z) (line ?x ?y))
Action:	(draw-line ?x ?y)
Preconditions:	((dot ?x) (dot ?y) (dot ?z) (dot ?q) (at ?x) (three-adj ?x ?z ?q) (three-adj ?z ?q ?y)
Effects:	((not (at ?x)) (at ?y) (crossed ?x) (crossed ?y) (crossed ?z) (crossed ?q) (line ?x ?y))

reformulated goal and averaged 1.5 failed attempts before finding a solution. These results are qualitatively equivalent to those of the blocks world problem–a simple problem that humans quickly find the solution for. This suggests that once the nine dots problem is reformulated the solution will come almost as quickly as the solution to the blocks world problem.

When the system was given the original goal with the reformulation mechanism activated, the steps taken in the previous two examples were combined and automated. The system solves 100% of the trials but averaged 14.1 failed attempts before finding a given solution. This is due to the

Table 4: A solution trace of our system attempting to solve the nine dots problem without reformulation.

Step #	Current Plan
1	()
2	((place-pen d5))
3	((draw-line d5 d3) (place-pen d5))
4	((draw-line d3 d1) (draw-line d5 d3) (place-pen d5))
5	((draw-line d1 d7) (draw-line d3 d1) (draw-line d5 d3) (place-pen d5))
6	((draw-line d7 d9) (draw-line d1 d7) (draw-line d3 d1) (draw-line d5 d3) (place-pen d5))
	Max plan length reached
7	()
8	((place-pen d9))
9	((draw-line d9 d1) (place-pen d9))
10	((draw-line d1 d7) (draw-line d9 d1) (place-pen d9))
11	((draw-line d7 d3) (draw-line d1 d7) (draw-line d9 d1) (place-pen d9))
12	((draw-line d3 d2) (draw-line d7 d3) (draw-line d1 d7) (draw-line d9 d1) (place-pen d9))
	Max plan length reached
	· · ·
	13 more equivalent failed attempts

Table 5: The reformulated goal state for the nine dots insight problem.

Goal State:	((crossed d1) · · · (crossed d9) (crossed d10) (crossed d11) (started-at d9) (line d9 d1) (line d1 d11) (line d11 d10) (line d10 d1))

additional failed attempts from trying to solve the with the original goal.

We have also shown the results of our system solving the original nine dots problem with reformulation. In this case 100% of the trials are successfully solved but the system averaged 16.61 failed attempts before each solution. This is due to failing 15 times on the original problem before modifying the problem and reformulating the goal, enabling a solution.

Discussion

The examples and initial experimental results shown in the previous section appear to support the three primary claims of our paper:

1. The primary difficulty in insight problem solving is due to misguided heuristic search–resulting in impasses.

2. Impasses are overcome by changing the search heuristic via a reformulation mechanism.

Table 7: Success rates given 100 trials (average number of failures before solution)

	Original goal, no reformulation	Reformulated goal, no reformulation	Original goal w/ reformulation
Original nine dots	0% (-)	0% (-)	100% (16.61)
Modified nine dots	31% (7.32)	100% (1.5)	100% (14.1)
Blocks World	100% (1.13)	–	–

Table 6: A solution trace of our system solving the nine dots problem without reformulation.

Step #	Current Plan
1	()
2	((place-pen D9))
3	((draw-line d9 d1) (place-pen d9))
4	((draw-line d1 d11) (draw-line d9 d1) (place-pen d9))
5	((draw-line d11 d10) (draw-line d1 d11) (draw-line d9 d1) (place-pen d9))
6	((draw-line d10 d1) (draw-line d11 d10) (draw-line d1 d11) (draw-line d9 d1) (place-pen d9))
	Goal achieved

3. Fast search, due to good heuristics, results in the rapid discovery of a solution.

In this section we show exactly how these claims are supported by these results. For the purposes of supporting our claims, we focus on the results from the modified version of the nine dots problem with no reformulation (shown in Table 7).

Claim 1 When comparing the original goal with the reformulated goal, it is apparent that the level of difficulty between these two formulations is drastic. The only thing that has changed between these two problems is the goal formulations and consequently the heuristics based them. With the original goal, the heuristics are much less accurate than with the reformulated goal, which is fully specified. Thus, we can conclude that the misguided weak heuristic search is the cause of the difficulty experienced when solving the original goal formulation.

Claim 2 The difference in success rates support our claim that impasses are overcome by changing the search heuristic via a reformulation mechanism. With the original goal, the success rate is only 31% apparently due to an impasse in the heuristic search. With the reformulated goal, success rate is 100%–suggesting the nine dots impasse has been overcome by goal elaboration.

Claim 3 The success rates and number of failed attempts for the reformulated goal are qualitatively equivalent to those seen on the blocks world problem. This suggests that solving the reformulated nine dots problem should be almost as rapid as solving the blocks world problem. This is substantially better than the performance on the original goal formulation. Since the only difference between formulations is the goal and the heuristics, we can conclude that the well guided heuristic search results in the rapid solution to the nine dots problem.

In summary, our results provide evidence for our claims regarding the insight process. Naturally, we suggest that additional experiments be conducted on more insight problems to provide evidence for the generality of these claims . This will be the focus of an expanded version of the paper.

Related Work

The topic of insight has a rich history and has been discussed by scientists across many fields. Historically, insight has been studied in the context of scientific discovery (Poincarè 1952; Hadamard 1945; Dreistadt 1968; 1969; Simon 1977) where the famous preparation–incubation–illumination–verification sequence was discovered. Some continued studying insight in this context (Dreistadt 1968; Simon 1977; Langley and Jones 1988) but others began studying insight problems in a laboratory setting. This was because the preparation–incubation–illumination–verification sequence could be consistently reproduced via insight problems in a controlled environment (Knoblich et al. 1999; Isaak and Just 1996; Falkenhainer 1987; Lung and Dominowski 1985). Some of the first researchers to pursue this line of research were the Gestalt psychologists who developed the initial ideas surrounding problem reformulation (Ohlsson 1984b).

From these lines of work, a number of dominating theories began to arise, each telling their own story of scientific insight. Two of the dominant theories were outlined previously, representational change theory (Knoblich et al. 1999) and the criterion of satisfactory progress theory (MacGregor, Omerod, and Chronicle 2001). Some other notable theories that have been developed are Hadamard's (1949) theory of insight, Simon's (1977) theory of selective forgetting, and Langley and Jones's (1988;1995) theory of spreading activation.

The work by Langley and Jones is particularly notable because it takes a computational approach to insight phenomena–implementing a theory of insight in a cognitive system and predicting and observing its behavior. This work has greatly influenced the approach taken in this paper. This contrasts with the the other notable theories we listed which take a psychological approach to insight problem solving–predicting and observing human behavior. We believe the computational approach is preferable because gaps in theories that would normally go unnoticed are quickly discovered during implementation. Furthermore, a successfully functioning insight problem solving system ensures a functional theory.

Future Work

The current implementation of a our theory uses goal elaboration via Answer Set Programming. This is a heavy handed approach as Answer Set Programming only finds fully elaborated goal states. This served the purpose of showing one way in which goal reformulation could be performed but we believe other mechanisms may be preferable. Of particular interest is abductive inference (Bridewell and Langley 2011) which supports plausible reasoning and only maintains one hypothesis, which can be incrementally improved. This mechanism is fast and produces inference similar to that of humans.

We would also like to incorporate reformulation into the ICARUS cognitive architecture (Langley and Choi 2011). This framework already has modules for conceptual inference, skill execution, and problem solving. We hope to add a module which performs problem reformulation. This new module would enable ICARUS to use reformulation to attempt to overcome impasses encountered in the course of problem solving. We believe that this will enable ICARUS to solve problems that it would have great difficulty solving, such as insight problems.

References

Bridewell, W., and Langley, P. 2011. A computational account of everyday abductive inference. In *Proceedings of the Thirty-Third Annual Meeting of the Cognitive Science Society.*

Chronicle, E. P.; MacGregor, J. N.; and Omerod, 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.

de Groot, A. 1978. *Thought and Choice in Chess, Second Edition*. Mouton & Co Publishers.

Dreistadt, R. 1968. An analysis of the use of analogies and metaphores in science. *The Journal of Psychology* 68(1):97–116.

Dreistadt, R. 1969. The use of analogies and incubation in obtaining insights in creative problem solving. *The Journal of Psychology* 71(2):159–175.

Einstein, A. 1982. How i created the theory of relativity. *Physics Today* 46.

Falkenhainer, B. 1987. Scientific theory formation through analogical inference.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3-4):189–208.

Gelfond, M. 2007. Answer sets. In *Handbook of Knowledge Representation*. Elviser. 285–316.

Hadamard, J. 1945. *An essay on the psychology of invention in the mathematical field.* Princeton University Press.

Isaak, M. I., and Just, M. A. 1996. Constraints on thinking in insight and invention. In Sternberg, R. J., and Davidson, J. E., eds., *The Nature of Insight*. 281–325.

Jones, R., and Langley, P. 1995. Retrevial and learning in analogical problem solving. In *Proceedings of the Seventeenth Conference of the Cognitive Science Society*, 466–471.

Jones, G. 2003. Testing two cognitive theories of insight. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 29(5):1017–1027.

Kershaw, T. C., and 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.

Knoblich, G.; Ohlsson, S.; Haider, H.; and Rhenius, D. 1999. Constraint relaxation and chunk decomposition in insight problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 25(6):1534–1555.

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

Langley, P., and Choi, D. 2011. Icarus user's manual.

Langley, P., and Jones, R. 1988. A computational model of scientific insight. In Sternberg, R. J., ed., *The nature of creativity: Contemporary psychological perspectives*. 177–201.

Langley, P. 1992. Systematic and nonsystematic search strategies. In Hendler, J., ed., *Intelligence planning systems: proceedings of the first international conference*. 145–152.

Lung, C.-T., and Dominowski, R. L. 1985. Effects of strategy instructions and practice on nine-dot problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 11(4):804–811.

MacGregor, J.; Omerod, T. C.; and 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* 27:176–201.

Ohlsson, S. 1984a. Restructuring revisited: An information processing theory of restructuring and insight. *Scandinavian Journal of Psychology* 25:117–129.

Ohlsson, S. 1984b. Restructuring revisited: Summary and critique of the gestalt theory of problem solving. *Scandinavian Journal of Psychology* 25:65–78.

Omerod, T. C.; MacGregor, J. N.; and Chronicle, E. P. 2002. Dynamics and constraints in insight problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 28(4):791–799.

Poincarè, H. 1952. *Science and Method*. New York: Dover. *(Originally published in 1908)*.

Segal, E. 2004. Incubation in insight problem solving. *Creative Research Journal* 16(1):141–148.

Simon, H. A. 1977. *Models of discovery: and other topics in methods of science*. D. Reidel Pub. Co.