Exploring the Effect of Animation and Progressive Revealing on Diagrammatic Problem Solving

Daesub Yoon[1], N. Hari Narayanan[2], Soocheol Lee[1], Oh-Cheon Kwon[1]

[1]Telematics & USN Research Division,
Electronics and Telecommunications Research Institute, Korea
{eyetracker, juin, ockwon}@etri.re.kr
[2]Department of Computer Science
& Software Engineering, Auburn University, AL 36849, USA
naraynh@auburn.edu

Abstract. We conducted eye-tracking studies of subjects solving the problem of finding shortest paths in a graph using a known procedure (Dijkstra's algorithm). The goal of these studies was to investigate how people reason about and solve graphically presented problems. First, we compared performance when the graphical display was animated to when the display was static. Second, we compared performance when the display was initially sparse, with detailed information being progressively revealed, to when the display presented all information simultaneously. Results suggest that while animation of the procedure or algorithm does not improve accuracy, animation coupled with progressively revealing objects of interest on the display does improve accuracy and other measures of performance.

1 Introduction

There is increasing research interest in the intelligent user interface community on interfaces that track and respond to the user's attention. To develop such interfaces, we have to understand how people view, comprehend and respond to visual displays of information. One aspect of this issue that our research has addressed is how people solve problems drawn from domains with the following five characteristics: (1) objects of the domain are spatially distributed; (2) the domain is dynamic, i.e. objects and their properties change over time; (3) objects interact with each other; (4) such interactions can be traced along chains of dependency relationships that branch and merge in spatial and temporal dimensions; and (5) predicting the future evolution of a system or object configuration requires reasoning from a given set of initial conditions and inferring chains of events along paths of dependency.

Understanding the cognitive processes underlying such reasoning can provide insights into the design of information displays that actively aid the problem solver and enhance performance. Given the increasing use of large format interactive displays for tasks such as weather forecasting, emergency management and military planning,

In *Diagrammatic Representation & Inference*, Lecture Notes in Artificial Intelligence, Vol. LNAI 4045, Springer, pp. 226-240.

results from such research may find significant practical application. In this context, we report on two experiments that investigated how people reason about graphs. In addition to determining the accuracy of their answers, we measured their response times and collected data on their eye movements. Our goal was to understand the relations between accuracy, patterns of visual attention allocation across the display and response times.

The rest of this paper is organized as follows. First, we summarize earlier work on diagrammatic reasoning that has a bearing on the present research. The next section explains Dijkstra's algorithm or procedure for finding shortest paths in an undirected weighted graph with an example. This section also illustrates the operation of animated and progressively revealing displays of this procedure. The following section describes the experiments. Section 5 presents results and discussion.

2 Prior Research

Diagrams are often used as external representations in problem solving. Problem solving with a static pictorial representation such as a diagram sometimes involves mental simulation of the behavior of the system that it depicts, requiring the reasoner to mentally manipulate the configuration or situation depicted in the diagram. In this case, diagrammatic reasoning may involve several cognitive processes. Prior research on diagram-based problem solving falls into three groups: theoretical analyses, empirically based cognitive modeling efforts, and empirical investigations of the problem solving process.

Larkin and Simon undertook a theoretical analysis of diagrammatic versus sentential representations and described features of diagrams that aid reasoning (Larkin & Simon, 1987). Extending this line of inquiry, Cheng proposed twelve functional roles of diagrams in problem solving (Cheng, 1996). These analyses suggest that diagrams of mechanical devices can aid a problem solver by explicating the structure, components, relations, states and causal dependencies in the devices.

Based on several experiments, Narayanan and Hegarty developed a cognitive model of multimodal comprehension and suggested guidelines for designing information displays (Narayanan & Hegarty, 2002). They describe the process of how people construct a mental model during multimodal comprehension. First, a static mental model is constructed by diagram decomposition, making representational connections, building referential connections, applying knowledge about basic laws, and hypothesizing causality. Then, people construct a dynamic mental model by mental animation and inference. Based on several experiments, they showed that multimedia systems designed according to their cognitive model were better in terms of facilitating comprehension and learning. They suggest six principles for information display design: decomposition principle, prior knowledge principle, co-reference principle, basic laws principle, lines of action principle, and mental simulation principle.

Several studies have focused on uncovering characteristics of the process of diagram-based problem solving. Narayanan et al (1994) employed a protocol analysis approach to discover characteristics of subjects' behaviors during diagrammatic rea-

soning. To explain how diagrams are used in reasoning, they proposed a process model of how people might solve mechanical reasoning problems from diagrams. They interpreted experimental results within the framework of this process model and found that diagrams aid the indexing and recall of relevant inferential knowledge. This research suggests that diagrams of systems are decomposed into components by reasoners, and that this decomposition is guided by spatial adjacency and causality. The model predicts that attention shifts across device components are mediated by spatial connections or contacts and causal dependencies. They found that diagrammatic reasoning about mechanical devices proceeded along the direction of causality.

In another experiment (Narayanan et al., 1995), these researchers further explored the role of the diagram in guiding the reasoning process. They used the diagram of an impossible or contradictory mechanical device. In that problem, the causal chain of events in the operation of the device split and merged at certain points. They postulated that reasoning trajectories of diagram-based problem solving were influenced by device structure, inferred causation, search strategy, verification goals, and short-term memory needs. They found evidence in verbal protocols that search strategy, verification goals and short-term memory support were factors that influenced focus shifts.

One limitation of those experiments is that no eye movement data was collected to track focus shifts. Instead, the researchers used verbal reports and gestures to infer the components that subjects were focusing on. This is an indirect measure. Eye movement data can provide direct information on focus shifts during diagrammatic problem solving.

Rozenbilt and his colleagues (Rozenbilt et al., 1998) conducted three experiments to determine if eye movement data could provide crucial information about moment-by-moment cognitive processes when subjects were reasoning about diagrammatic problems. They found that independent raters who did not participate in the experiments were able to actually predict principal axes and principal directions of the visual stimuli presented to subjects simply by looking at subjects' scan paths without the stimuli. In case of mechanical reasoning problems, raters predicted principal axes and principal direction more accurately than for non-mechanical reasoning problems. In another experiment, independent raters tried to predict subjects' accuracy by observing their eye positions overlaid on the diagram of the device; more than 75% of the time, the raters correctly predicted subjects' accuracy. This suggests that eye movement patterns contain crucial information about how people are reasoning with a diagram.

Grant and Spivey conducted two experiments to find out how people look at a diagram and how their eye movement patterns correlate with inference making (Grant & Spivey, 2002). In the first experiment, they discovered that when people solve a diagrammatically presented medical problem, those who produced the correct solution looked at a particular area of the diagram for a long time. This area contained information critical to solving the problem. They considered only the fixation time (i.e. how long someone looked at a specific area of the diagram), not gaze patterns. They did a second experiment with three different conditions, one of which was designed to attract subjects' attention to this critical area using a blinking technique on the display. During the second experiment, they did not track subjects' eye fixations.

Subjects who saw the blinking diagram performed better than those who saw a static diagram. This result indicates that appropriate attention guidance can improve diagrammatic reasoning. But a deficiency of their experimental method is that they did not know whether the subjects actually looked at the blinking area during the experiment because eye tracking was not done.

Yoon and Narayanan conducted an eye tracking study of diagrammatic problem solving (Yoon & Narayanan, 2004) in the domain of mechanical devices. Their study revealed eye movements suggestive of the use of mental imagery to solve mechanical reasoning problems presented as diagrams. These researchers found that subjects who employed mental imagery during problem solving exhibited more eye fixations, looked at more components of the problem, spent more time looking at important components, and gazed across the diagram in a more systematic fashion. The implication is that interfaces that help users better guide their visual attention may improve problem solving performance.

3 The Shortest Path Problem and Its Solution Procedure

Many problem solving situations admit well-defined procedures that specify components or elements of the problem that a successful problem solver must pay attention to. So it is reasonable to hypothesize that a display that guides the attention of the problem solver to these components, in a particular order if so specified by a procedure that is available, can positively influence the user's problem solving process and its results.

The problem we chose to investigate this hypothesis is the shortest path problem: given a graph G with edge costs and a starting node s, find the shortest cost paths from s to every other node in G. Many practical problems, such as route planning and project planning, can be represented as shortest path problems. An approach to solving such problems is given by Dijkstra's algorithm. This algorithm, invented by Edgar Dijkstra, solves the problem in stages. In the first stage, it considers edges to all nodes adjacent to s one at a time. The cost of the path to each adjacent node (from s), which is nothing but the cost of the edge connecting it with s, is then attached to that node. In the second and following stages, the algorithm picks one node, say j, from among all nodes with a finite cost attached to them – j will be the node with the smallest cost. Then the algorithm declares that the shortest path from s to j is now known, and proceeds to consider, in turn, each node adjacent to j to see if a smaller cost path from s through j to this adjacent node can be found. If so, the attached cost of that node is appropriately updated. Then, as before, the algorithm picks the next node j to declare that the shortest path to it is now known. When this declaration has been made for all nodes in the graph, the algorithm terminates. This is an efficient and systematic way to find shortest paths from s to all other nodes. It can be shown that n to the power 3 is an upper bound for the number of steps needed to solve this problem for a graph with n nodes.

It is obvious that the algorithm considers elements of the problem (i.e. nodes and edges of a graph) in a systematic fashion, and produces and updates relevant informa-

tion (changing path costs from s to various nodes), during the course of problem solving. What we investigated was the question of whether users trained on applying (i.e. mentally simulating) Dijkstra's algorithm on graphs are helped to different extents by different kinds of information displays that present shortest path problems to them.

One possible display to present a shortest path problem to a user is a static one – the picture of the graph with all relevant node and edge information included. A second possible display is one that graphically animates the various steps of the algorithm. Figure 2 shows various snapshots of an animated display that illustrates the operation of Dijkstra's algorithm on an undirected weighted graph (shown in Figure 1).

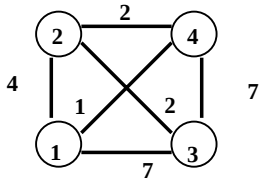

Fig. 1. A simple undirected weighted graph

Read Figure 2 column by column (left to right) and each column top to bottom. It shows a 4 node graph with node 1 being the start node s. Each edge has a cost attached to it. Each node has an information triple attached to it. The first item of this triple is either 0 (indicating that the shortest path to it is not yet known) or 1 (indicating that the algorithm has declared that the shortest path to this node is known). This item is 1 for node 1 since it is the start node, and therefore the shortest path from it to itself is already known. This node is also colored blue to indicate this (in the black and white figures we use a horizontal pattern to indicate a blue colored node). The second item of the triple is the cost of the shortest path to this node found thus far. This item is 0 for node 1 since it is the start node. The third item is the previous node in this shortest path. This item is S for node 1 since it is the start node.

In the first stage, the algorithm considers edges to all nodes adjacent to node 1 one at a time. So it first considers the edge 1-2, which is indicated by that edge turning red from its default black color. Next, the algorithm computes the cost of the path from node 1 to node 2 (which, in this case, is simply 4). This is indicated by node 2 turning red (in the black and white figures we use a diamond pattern to indicate a red colored node). Now the cost of this path is attached to node 2. This is indicated by the information triplet [0,4,1] appearing beside node 2, indicating that while the shortest path to node 2 is not yet known, a path of cost 4 is known and the previous node in that path is node 1. All these can be seen in the second graph of Figure 2. After these operations, edge 1-2 reverts to black color. When this process is repeated for edge 1-3 and node 3, and edge 1-4 and node 4, the graph will appear as in the third graph of Figure 2. Now there are three nodes, 2, 3 & 4, with finite path costs. So in the second

stage the algorithm picks node 4, which has the smallest path cost of 1, and declares that the shortest path from node 1 to node 4 is now known. This is indicated by the edge 1-4, node 4 and its information triplet turning blue, with the first item of the triplet changing from 0 to 1, as can be seen in graph 4 of Figure 2.

Now the process repeats for node 4. Its edges 4-2 and 4-3 and adjacent nodes 2 and 3 are considered in turn. Note that the edge 4-1 and node 1 are not considered further as these are parts of a known shortest path already. The cost of a path from node 1 to node 2 through node 4 is 1+2=3, which is less than the cost of the previously computed path to node 2 (of cost 4), so the information triplet of node 2 is updated from [0,4,1] to [0,3,4]. The cost of a path from node 1 to node 3 through node 4 is 1+7=8, which is more than the cost of the previously computed path to node 3 (of cost 7), so its information triplet is not updated. These operations can again be illustrated by the appropriate edges, nodes and information triplets changing color and the values changing. This situation is depicted by graph 5 in Figure 2.

Now there are two nodes, 2 and 3, whose shortest paths from node 1, the start node, are not yet known. So in the third stage the algorithm picks node 2, which has the smallest path cost of 4, and declares that the shortest path from node 1 to node 2 is now known. This is indicated by the edge 4-2, node 2 and its information triplet turning blue, with the first item of the triplet changing from 0 to 1, as can be seen in graph 6 of Figure 2. Now the algorithm considers nodes adjacent to node 2. In this case there is only one node, 4, that is left and the rest of this process, therefore, should be obvious. Graph 7 in Figure 2 indicates the consideration of edge 2-3, and the updating of the information triplet of node 3. Graph 8 shows the final state, when the shortest paths from node 1 to all other nodes are found (these paths are 1-4, 1-4-2 and 1-4-2-3). What we have illustrated here are both Dijkstra's procedure and how an animated display of the procedure might operate.

One might conceive of an alternate display that not only shows this animation of the shortest path finding procedure but also takes a "just in time" approach by only revealing components of the graph that the animation is illustrating at any moment. So, initially, only node 1 and its state are displayed (Figure 3; read this figure also left to right column-wise and each column top to bottom). Then, when the procedure considers node 2, it and the edge to it are revealed with appropriate color codings as described above. Similarly, other nodes and edges are also revealed only when they are considered for the first time by the procedure. This will result in a progressive revealing of information as shown in Figure3.

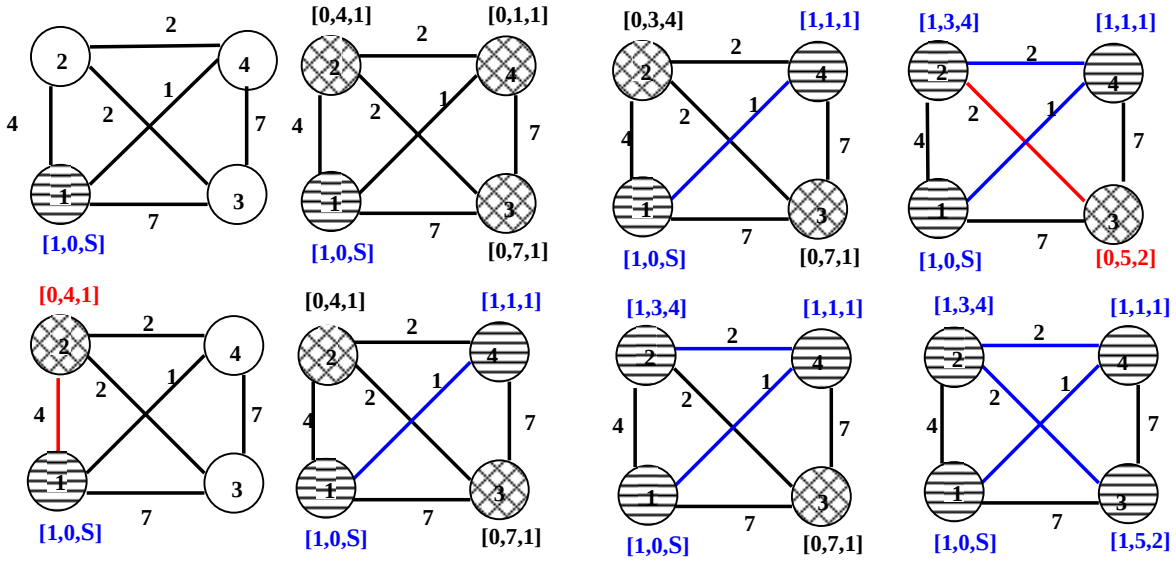

Fig. 2. Illustration of Dijkstra's algorithm operating on a simple graph, showing stages of an animated display; discussed in text column-wise left-to-right and top-to-bottom in each column.

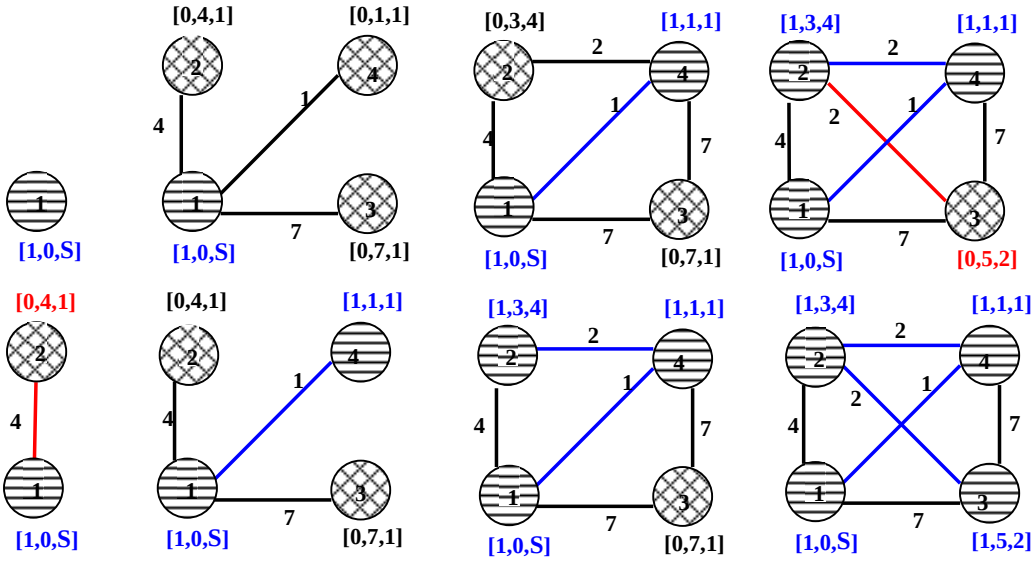

Fig. 3. Snapshots of a progressively revealing animated display of Dijkstra's algorithm operating on a simple graph; discussed in text column-wise left-to-right and top-to-bottom in each column.

4 Experiments

4.1 Comparing an animated display to a static display

Cognitive and computational modeling in the mechanical domain (Hegarty, 1992; Narayanan, Suva & Motoda, 1994; 1995) suggest that construction of a dynamic mental model of a system is often accomplished by considering components individually, inferring their behaviors due to influences from other connected or causally related components, and then inferring how these behaviors will in turn affect other components. This can involve both rule-based inferences that utilize prior conceptual knowledge and visualization processes for mentally simulating component behaviors (Narayanan, Suwa & Motoda, 1994; 1995; Schwartz & Black, 1996; Sims & Hegarty, 1997). In the domain of machines, a spatial visualization process called mental animation (Hegarty, 1992) is involved in the simulation of component behaviors. Mental animation appears to be constrained by working memory capacity such that people are only able to mentally animate one or two component motions at a given time (Hegarty, 1992). Working memory demands are imposed when several mechanical components constrain each other's motions, so that the motion of components cannot be inferred one by one (Hegarty & Kozhevnikow, 1999) or if imagining the motion of a component changes the configuration of components so that it no longer corresponds to the external display (Narayanan, Suwa & Motoda, 1994). Furthermore, this type of mechanical reasoning is particularly demanding of spatial working memory processes (Sims & Hegarty, 1997). This evidence suggests that an adaptive display that can provide local animation or other kinds of visualizations showing the behaviors of individual components of a system is likely to improve problem solving performance by reducing working memory demands and freeing up more mental resources to meet processing demands of the problem at hand. This prediction was tested.

In this study, we had two conditions. Subjects in both conditions were given training on Dijkstra's algorithm prior to start of the experiment. The experimental condition showed subjects an animation of how to find shortest path in a weighted undirected graph according to Dijkstra's algorithm as explained in Section 3. Figure 4 shows the stimulus used. The animation did not run to completion. Instead, it was stopped in the middle and subjects were asked to predict the next step (the problem to be solved appears in Figure 4). The control condition showed a static diagram of the state of the graph at the point at which animation stopped (Figure 4 shows the state of the graph at this point).

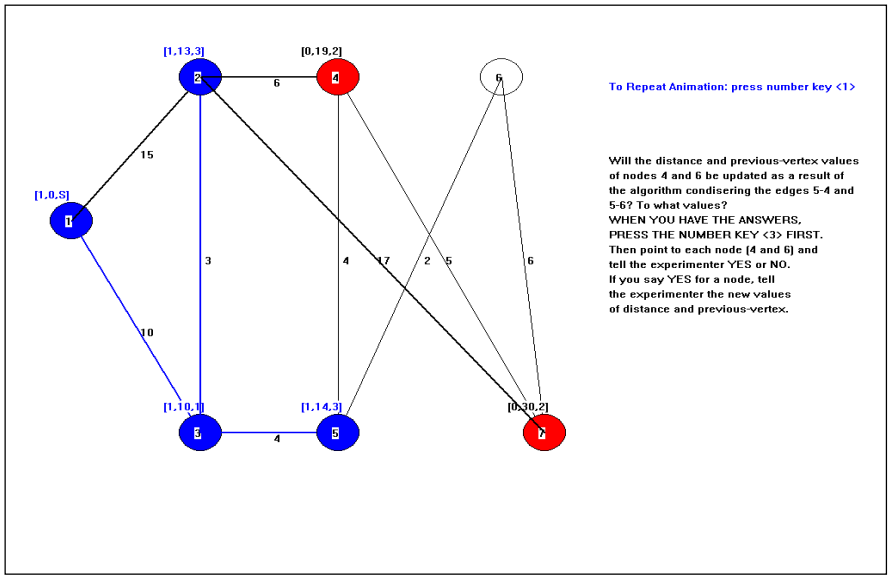

Fig. 4. Graph used in the experiments

4.2 Comparing a progressively revealing animated display to an animated display that presents all information simultaneously

The cognitive process model of Narayanan and Hegarty predicts that successful decomposition of a system into its constituents is a necessary precursor to accurate mental model construction. This process can be hindered if the display is dense, with many variables shown, or if the problem solver lacks the necessary background knowledge of the domain and representational conventions to successfully parse the display. For example, Hegarty and Shimozawa (2001) reported the following findings from the meteorology domain: (1) it takes longer to verify a fact about a specific variable from a weather map if the map shows multiple variables; (2) the number of variables displayed on a weather map influences both encoding and inferences from weather maps; and (3) performance is slower and less accurate for maps that included irrelevant variables. Visual clutter is a problem in at least two ways. When both relevant and irrelevant visual information are present, separating the two and focusing only on the relevant is a comprehension challenge. Even when most of the information present in the display is relevant (as is the case in our stimulus), focusing only on the right display objects at the right time during various stages of the problem solving process is not an easy task. Therefore, a display that progressively reveals relevant

information instead of showing all information at once is likely to improve problem-solving performance. This prediction was tested.

We used the graph in Figure 4. The control condition display was the animated display from the previous experiment. The experimental condition display showed the same animation, but with the additional property that an initially blank display progressively revealed each component (in a fashion similar to the illustration in Figure 3) as the animation progressed.

4.3 Procedure

Fifty five undergraduate students of engineering volunteered to participate in the experiment, in return for a nominal payment. They were recruited from an undergraduate algorithm class in the Computer Science & Software Engineering Department at Auburn University. Subjects were assigned to two matched groups based on their GPA, one assigned to a control condition and the other to an experimental condition. The experiment was conducted one subject at a time in an eye tracking laboratory equipped with the SMI head-mounted eye tracker, eye tracking computer, and a stimulus display computer.

First, subjects studied Dijkstra's algorithm for finding the shortest path with a printed tutorial, and then watched an animation of the procedure (similar to Figure 2) as many times as they wanted. No time limit was imposed in this training phase. When a subject indicated that he or she was ready, the person was asked to sit on a chair, and watched the stimulus display on a 20-inch monitor at eye level at a distance of approximately 3 feet. The experimenter sat behind the subject and controlled the experiment through the eye-tracking computer. The experiment proper began by the subject pressing number key 1. After watching the stimulus display corresponding to the experimental or control condition, the subject depressed number key 3 to indicate that he or she had the answer, and then wrote it on a printed picture of the graph that the experimenter provided. Eye movement data was collected and recorded between these two key presses.

4.4 Process and outcome measures

We collected two process measures (eye movements and response times) and one outcome measure (accuracy of answer provided to the question in Figure 4, scored on a scale of 0-6). Response time and accuracy are commonly used metrics of problem solving performance. However, not all problems in visuo-spatial domains have answers that can be unequivocally classified as correct or incorrect. A case in point is developing an action plan for an emergency evacuation from an information display that shows factors such as population distribution, layout of roads, features of the terrain and weather conditions. Here it is as important to ensure that the problem solver has considered all critical elements of the domain as it is to create a feasible plan. We employed two measures called *coverage* and *order*, besides accuracy and response time, to characterize the quality of problem solving. Coverage and order are derived from eye movement data, as explained below.

Coverage is defined as the percentage of objects in the display that were attended to for more than a certain threshold. We set the threshold to 200 milliseconds, approximately equal to two fixations. Coverage is therefore a number between 0 and 100.

A good strategist will not only attend to all relevant objects in the display, but also consider them in the order that best supports reasoning. For example, a crucial feature that separates expert and novice problem solving in meteorological reasoning with weather maps is that novices attend to objects that are perceptually salient whereas experts attend to objects that are thematically relevant (Lowe 1999). Therefore, we developed a metric called order that measures how systematically a subject attended to causally related elements of the display. This metric is explained next.

Let S be an ordered sequence of display objects that a user attended to during a problem solving session. So S begins with the first display item attended to, and ends with the last item attended to before the solution to the problem is produced. This sequence is generated from eye movement data. In this sequence, if object j appears immediately after object i, and if i can influence j in the event chains of the system, then i-j represents an ordered dependent pair in the sequence S. Consecutive ordered dependent pairs represent ordered subsequences of S. The length of a subsequence is the number of dependent pairs in it. Order of S is defined as the sum of squares of the lengths of subsequences in S. This captures the correctness of the sequential order in which the user visually scanned the display (i.e. each dependent pair indicates that the problem solver considered one pair of display objects in the direction of dependency or causality) weighted by the number of consecutive dependent pairs that have been considered (i.e. if subjects A and B both considered the same number of dependent pairs, but if A looked at longer subsequences than B, the value of order will be higher for A than B). Order is a number greater than or equal to zero.

From the raw eye movement data we also computed the total fixation duration on each component using a bounding box technique. We additionally determined the total number of fixations of each subject for each problem (excluding fixations on the question and on blank areas of the screen).

5 Results and Discussion

5.1 Comparing an animated display to a static display

In this experiment, the experimental condition showed subjects an animation of how to find the shortest path in a graph and the control condition showed a static picture as described in Section 4.1.

We expected that, though subjects in both conditions knew of Dijkstra's procedure, the animated display that showed the operation of the procedure until a certain point would produce better accuracy than a static display that showed only the state of the graph that resulted from partial operation of the procedure. We expected response time for the animated display to be more since the animation was likely to encourage more systematic scans of the display than a static picture. We also expected that the animated display would produce more coverage and higher order than the static display.

Table 1 shows results of answer accuracy (Ans), response time (RT), coverage (Co), number of focus shifts (F/S) and order. We used only 48 subjects' eye movement data (24 in each condition) for the data analysis due to problems with collecting good quality eye movement data from all subjects. In terms of accuracy, there was no significant difference between the two conditions. In terms of response time, the animated display increased response times (T-test = 2.113, p-value = 0.04). It also produced higher coverage (T-test = 7.253, p-value = 0.0001). Also, the animated display produced more focus shifts (T-test = 7.363, p-value = 0.00001) and a higher value of order (T-test = 4.257, p-value=0.00001).

Even though there was no significant difference in accuracy between the animated and static displays, there were significant differences in response time, coverage, number of focus shifts and order. This suggests that animated displays induce viewers to look longer at them, look at and across more display objects, and follow the animations systematically, but these visual behaviors do not necessarily lead to more accurate problem solving. One possible explanation for the animated display not improving accuracy is that subjects did not have control over the animation once it started (except to repeat it if desired). So there could have been a speed mismatch between the external animation and the internal simulation of Dijkstra's procedure. This result therefore adds to the extant literature (e.g., Tversky et al. 2002) indicating that animations do not necessarily improve comprehension.

Table 1. Overall results of experiment 1

		Ans	RT	Co	F/S	Order
Exp 1	Mean	1.42	102.6	74.27	166.7	7.29
N(24)	SD	0.5	30.8	13.07	60.49	5.36
Con 1	Mean	1.27	67.89	46.08	61.2	2
N(24)	SD	0.68	74.35	13.84	35.61	2.89
T-test		0.848	2.113	7.253	7.363	4.257
P-value		0.401	0.04	0.0001	0.00001	0.00001

5.2 Comparing a progressively revealing animated display to an animated display that presents all information simultaneously

Here the experimental condition showed an animated graph in which information was progressively revealed and the control condition was an animated display that showed the complete graph.

We expected that the progressively revealing display would produce better accuracy and lower coverage (because this display followed a just in time approach to revealing information), and higher order, than the animated display.

Initially, we assigned 28 subjects to the experimental condition and 27 subjects to the control condition but we could only use data from 19 subjects in the experimental condition and data from 24 in the control condition due to bad calibration of the eye tracker. Table 2 shows results of comparing answer accuracy, response time, coverage, number of focus shifts and order. The progressively revealing display produced better accuracy (T-test = 2.06, p-value = 0.0458). There was not a significant differ-

ence in response time between groups. The control condition had significantly higher coverage (T-test = 2.067, p-value = 0.0451). There was no significant difference in the number of focus shifts between groups. The experimental condition had a significantly higher value of order (T-test = 2.238, p-value = 0.0307).

These results suggest that progressively revealing information helps viewers to be more accurate (compared to static and animated displays), scan the display more systematically (compared to static and animated displays), and be more efficient (improve accuracy while viewing a smaller percentage of display objects compared to an animated display), without increasing response time (compared to an animated display).

One can use the notion of visual search (Larkin & Simon, 1987) to explain the better performance observed with a progressively revealing display, i.e. less information on the display requires less search to find what is relevant. But note that at any stage of solving the shortest path problem, one actually needs to focus only on a few adjacent nodes and their edges regardless of how big the graph is. Therefore, scope of visual search is not affected by whether the rest of the graph is visible or not. Thematic relevance of parts of the graph varies as one executes Dijkstra's procedure, and it is possible that subjects in the animated display condition wasted attentional resources on parts that are irrelevant whereas the progressively revealing display helped them identify relevant display objects at different stages of problem solving. The implication is that animation does not necessarily induce efficient allocation of visual attention, but accuracy and efficiency can be improved through display techniques such as progressive revealing.

Table 2. Overall results of experiment 2

		Ans	RT	Co	F/S	Order
Exp 2	Mean	5	160.6	64.54	209.2	9.26
N(19)	SD	2	69.41	9.9	82.83	6.43
Con 2	Mean	3.71	161.5	71.05	213.4	6.08
N(24)	SD	2.07	86.27	10.52	115.1	2.41
T-test		2.060	0.038	2.067	0.134	2.238
p-value		0.0458	0.97	0.0451	0.894	0.0307

5.3 Discussion

This paper describes experiments on diagrammatic reasoning with graphs in which we compared process and outcome measures of problem solving when the display showed local processes through visualization techniques such as animation, to when the display was static; and when the display was initially sparse, with detailed information being progressively revealed, to when the display presented all information simultaneously. One outcome measure (accuracy) and four process measures (response time, coverage, number of focus shifts and order), three of which were derived from eye movements, were analyzed to compare problem solving performance. Results indicate that animations induce viewers to look longer at the display, look at and across more display objects, and follow the animations systematically, but these visual behaviors do not necessarily lead to more accurate problem solving. On the other

hand, an information display that, besides being animated, also progressively reveals relevant information improves problem solving performance in terms of accuracy, systematicity and efficiency.

This research has implications for the design of information displays that actively track the viewer's visual attention in order to support reasoning and problem solving. Grant and Spivey (2002) report that merely attracting the problem solver's attention to relevant regions of a display through an attention attracting mechanism can dramatically improve accuracy. The problem they studied was Duncker's radiation problem, which was not a problem that required reasoning from initial conditions along pathways of dependencies unlike the shortest path problem. In complex domains where objects participate in spatially distributed events or operations, our results provide guidance on how dynamic displays could be designed to better support diagrammatic reasoning.

However, additional research is required before concrete design recommendations can be made. One limitation of current work is that while the reasoning process that subjects engaged in (a mental simulation of Dijkstra's procedure) is dynamic, the underlying graph does not change, and the procedure itself is well-defined. Real-life problems such as planning the evacuation of a city while viewing traffic maps can be modeled as finding shortest paths in a weighted graph, but such problems are solved under competing constraints and rapid changes in underlying conditions. Therefore, our future work will focus on diagrammatic reasoning with less well-defined, heuristic approaches and more flux in relevant variables.

Acknowledgment: This research was supported by the Office of Naval Research under contract N00014-03-10324. Preparation of this paper was supported by the National Science Foundation (NSF) through an independent research & development contract to the second author, and by the Electronics and Telecommunications Research Institute (ETRI) in case of the other authors. The views and opinions expressed in this paper are those of the authors, not of NSF or ETRI.

References

1. Cheng, P. C-H. (1996). Functional roles for the cognitive analysis of diagrams in problem solving. *Proceedings of the 18th Annual Conference of the Cognitive Science Society* (pp. 207-212). Hillsdale, NJ: Lawrence Erlbaum Associates.
2. Grant, E. R., & Spivey, M. J. (2002). Guiding attention produces inferences in diagram-based problem solving. In M. Hegarty, B. Meyer & N. H. Narayanan (Eds.), *Multidisciplinary Studies of Diagrammatic Representation and Inference*, LNAI 2317, Berlin: Springer-Verlag.
3. Hegarty, M. (1992). Mental animation: Inferring motion from static diagrams of mechanical systems. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 18(5), 1084-1102.
4. Hegarty, M. & Kozhevnikov, M. (1999) Spatial ability, working memory and mechanical reasoning. In J. S. Gero & B. Tversky, Eds. *Visual and Spatial Reasoning in Design*, pp.221-240. Sydney, Australia: Key Center for Design Computing and Cognition, University of Sydney.

5. Hegarty, M. & Shimozawa, N. (2001) Interpretation of weather maps by non-experts. *Presentation at the ONR METOC Workshop*, Applied Physics Laboratory, University of Washington, Seattle, August.
6. Larkin, J., & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-100.
7. Lowe, R. K. (1999). Extracting information from an animation during complex visual learning. *European Journal of the Psychology of Education*, 14, pp.225-244.
8. Narayanan, N. H., & Hegarty, M. (2002). Multimedia design for communication of dynamic information. *International Journal of Human-Computer Studies*, 57, 279-315.
9. Narayanan, N. H., Suwa, M., & Motoda, H. (1994). A study of diagrammatic reasoning from verbal and gestural data. *Proceedings of the 16th Annual Conference of the Cognitive Science Society* (pp. 652-657). Hillsdale, NJ: Lawrence Erlbaum Associates.
10. Narayanan, N. H., Suwa, M., & Motoda, H. (1995). Diagram-based problem solving: The case of an impossible problem. *Proceedings of the 17th Annual Conference of the Cognitive Science Society* (pp. 206-211). Hillsdale, NJ: Lawrence Erlbaum Associates.
11. Rozenbilt, L., Spivey, M., & Wojslawowicz, J. (1998). Mechanical reasoning about gear-and-belt diagrams: Do Eye-movements predict performance? *Proceedings of Mind III: The Annual Conference of the Cognitive Science Society of Ireland* (pp. 158-165).
12. Schwartz, D.L. & Black, J.B. (1996). Analog Imagery in Mental Model Reasoning: Depictive Models. *Cognitive Psychology*, 30, 30, 154-219.
13. Sims, V. K. & Hegarty, M. (1997). Mental animation in the visual-spatial sketchpad: evidence from dual-task studies. *Memory & Cognition.*
14. Tversky, B., Morrison, J. B. & Betrancourt, M. (2002). Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57: 247-262.
15. Yoon, D., Narayanan, N. H. (2004). Predictors of success in diagrammatic problem solving. In A. Blackwell, K. Marriott & A. Shimojima (Eds.), *Diagrammatic Representation & Inference*, LNAI 2980, Berlin: Springer-Verlag, pp. 301-315.