

# Solving combinatorial problems: The 15-puzzle

ZYGMUNT PIZLO and ZHENG LI

Purdue University, West Lafayette, Indiana

We present a series of experiments in which human subjects were tested with a well-known combinatorial problem called the *15-puzzle* and in different-sized variants of this puzzle. Subjects can solve these puzzles reliably by systematically building a solution path, without performing much search and without using distances among the states of the problem. The computational complexity of the underlying mental mechanisms is very low. We formulated a computational model of the underlying cognitive processes on the basis of our results. This model applied a pyramid algorithm to individual stages of each problem. The model's performance proved to be quite similar to the subjects' performance.

During the last several years, there has been increasing interest in studying how humans solve combinatorial problems, such as the traveling salesman problem (TSP; Graham, Joshi, & Pizlo, 2000; MacGregor & Ormerod, 1996; Vickers, Lee, Dry, & Hughes, 2003). We will first briefly describe the main characteristics of the TSP problem, because prior research on it motivated the present study. The TSP task requires the problem solver to find the shortest tour of  $N$  cities. This problem belongs to the class of  $NP$  complete problems, which means that in order to solve it (find the shortest tour), a problem solver would have to, in the worst case, perform an exhaustive search through all possible tours (Garey & Johnson, 1979; Gutin & Punnen, 2002; Lawler, Lenstra, Rinnooy Kan, & Shmoys, 1985). Since the number of tours grows very quickly with the size,  $N$ , of the problem, an exhaustive search is impractical even for moderate-sized problems. Therefore, researchers have concentrated on formulating approximating algorithms that can find a reasonably short tour, where the solution time is a low-order polynomial function of the problem's size. Approximating algorithms are available for the cases in which the distances between pairs of cities satisfy metric axioms (i.e., the distances are nonnegative, symmetric, and satisfy triangle inequality). The most common example of a metric TSP is the Euclidean TSP (E-TSP), in which cities "reside" on a Euclidean plane. That is, the distances between cities are Euclidean distances. Human subjects find the shortest tour in the E-TSP quite often, and when they do not, the tour that they find has a length not much longer than the shortest possible. At the same time, the solution time is proportional to the size of the E-TSP problem, suggesting that the underlying mental mechanism has very low computational complexity (Graham et al., 2000). This implies that solving the

TSP problem involves only a minimal amount of search. These results pose a challenge to modelers of human problem solving—namely, what is the nature of the algorithm that produces good approximations to a global minimum without performing global search?

Human performance in the E-TSP and the underlying mental mechanisms have been modeled by using such concepts as *convex hull* (MacGregor & Ormerod, 1996), *proximities* (Vickers et al., 2003), and *hierarchical clustering with coarse-to-fine sequence of tour approximations* (Graham et al., 2000; Pizlo & Li, 2004). Although we are still far from a full understanding of how humans solve the TSP, it is worthwhile examining human performance in other combinatorial tasks, in order to provide experimental evidence that may, at some point, allow formulating a more general theory. In this article, we report the results of several experiments with human subjects, as well as the results of simulation experiments that employed a well-known puzzle called the *15-puzzle*, which was introduced by Sam Loyd in 1878 (Weisstein, 2003). We also used variants of this puzzle (sizes: 5, 8, and 35).

The 15-puzzle is a sliding-tile puzzle. A set of 15 tiles, numbered from 1 to 15, and an empty space are placed on a  $4 \times 4$  board (see Figure 1). The subject's task is to rearrange the tiles, starting from a *scrambled* arrangement (Figure 1A shows an example), to form an ordered arrangement, called the *goal* (Figure 1B). A tile can be moved only if there is an empty space next to it. So, in the case of the state shown in Figure 1A, four moves are possible: Tile 1 can be moved down, Tile 2 can be moved to the right, Tile 6 can be moved up, and Tile 7 can be moved to the left. The total number of possible arrangements is  $16!$ . These arrangements form two equal and disjoint sets. This means that by performing valid moves, only one half of all the possible arrangements (permutations) can be reached (see Weisstein, 2003, for details).

The 15-puzzle has different-sized variants. The smallest size involves a board  $2 \times 3$  and is called the 5-puzzle. The 8-puzzle involves a board  $3 \times 3$ . The 35-puzzle involves a board  $6 \times 6$  (boards of other shapes—e.g.,  $4 \times 9$ —could

---

Partial support for this research was provided by the Air Force Office of Scientific Research. Correspondence concerning this article should be addressed to Z. Pizlo, Department of Psychological Sciences, Purdue University, 703 Third Street, West Lafayette, IN 47907-2081 (e-mail: pizlo@psych.purdue.edu).

also be used). The family of these puzzles will be called the *n*-puzzle, where *n* stands for the number of tiles. In all of the *n*-puzzles we used, the tiles in the goal state were ordered from left to right and from top to bottom, with an empty space located in the bottom right corner (previous studies of the 8-puzzle have usually used a goal state in which the empty space was in the center and the eight tiles were ordered around the boundary of the  $3 \times 3$  board). Previous research on the *n*-puzzle has concentrated on the 8-puzzle (O'Hara & Payne, 1998; Russell & Norvig, 1995; Schofield, 1967). It is known that the family of *n*-puzzles belongs to the class of *NP* complete problems, which means that the number of paths grows exponentially with the number of tiles and that finding the shortest path from the start to the goal may require performing an exhaustive search (Ratner & Warmuth, 1986).

The present study was designed to characterize human performance in the *n*-puzzle and to answer several specific questions—namely, (1) what is the computational complexity of the mental algorithms? (2) do subjects perform a substantial amount of search? (3) are subjects likely to use distances among the states when they solve the puzzles? and (4) do subjects have a reliable rule for deciding where to go next, without considering alternatives?

Answers to these questions will provide motivation for considering a pyramid algorithm as a possible model of the underlying mental mechanisms. Pyramid algorithms were developed to solve difficult problems in computer vision (Bouman & Liu, 1991; Jolion & Rosenfeld, 1994; Rosenfeld & Thurston, 1971) and were subsequently adopted as models of human vision (Pizlo, Rosenfeld, & Epelboim, 1995; Pizlo, Salach-Golyska, & Rosenfeld, 1997).

In a pyramid algorithm, an image is represented on many layers of scale and resolution. Bottom layers have small receptive fields that are characterized by a high spatial resolution. Higher layers have progressively larger receptive fields with proportionally lower spatial resolution. Pyramid algorithms can efficiently solve some difficult tasks because they can extract global properties from an image by performing only local computations. This is accomplished by analyzing an image in both bottom-up (fine-to-coarse) and top-down (coarse-to-fine) directions. The top-down analysis is especially useful because it allows more global representations to “guide” (or direct) the analysis of more local representations. As a result, a pyramid algorithm can find near-optimal solutions with a minimal amount of search.

Graham et al. (2000) showed that a pyramid algorithm is a plausible model of the mental mechanisms that are involved in solving the E-TSP. In their model, the cities themselves were represented by sharp peaks of intensities on a background of zero intensity. This image was the first layer of the pyramid. Representations on higher layers were produced by blurring the image. More blurring resulted in coarser representations. The modes of the intensity distribution in the blurred image corresponded to clusters of cities: Higher layers had fewer modes. After the representations on all layers were produced, the top-down

5	1	3	4
2		7	8
9	6	10	12
13	14	11	15

A

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

B

Figure 1. 15-puzzle. (A) Start state. (B) Goal state.

solution process started. The solution tour was produced in a coarse-to-fine sequence of approximations in which tours involved modes of intensity distribution, rather than the cities themselves. The algorithm started solving the problem on a layer with a very coarse representation of the original image where there were only three or four modes of the intensity distribution. Solving three-city or four-city TSPs is trivial. This tour of modes was then used as an approximation to a tour of modes on a finer representation. This process was repeated until the bottom layer was reached, where the original problem was represented. Graham et al. showed that their model could quickly produce solutions that were as accurate as those produced by subjects. In the present study, we explore the possibility of generalizing image pyramids to such problems as the 15-puzzle, where a graph, rather than an image, is an adequate representation of the problem space.

## EXPERIMENT 1 Solving the 15-Puzzle

### Method

**Stimuli.** Five instances of the 15-puzzle were used (henceforth, we will refer to instances of the *n*-puzzle as *problems*). The problems were shown on a computer screen. The start states for the problems were produced by applying 80 random moves, starting with the goal state shown in Figure 1B. The sequence of moves that produced each problem (this sequence will be called a *scramble file*) was stored. The scramble files were used in later experiments.

**Subjects.** The 2 authors (Z.P. and Z.L.) and 2 graduate students (J.M. and G.L.), who were naive about the hypotheses being tested, served as subjects.<sup>1</sup> All the subjects were familiar with the 15-puzzle.

**Procedure.** The subjects used a computer mouse to solve the problems. Moves were performed by clicking on a tile to be moved. Each subject solved several problems as practice (the first author solved a large number of problems before he ran in the experimental sessions). The subjects were instructed to “solve the problem by producing as short a path as possible.” The subjects were not explicitly asked to trade planning for actual moves. The issue of such a trade-off was addressed by O'Hara and Payne (1998). They had their subjects, who solved the 8-puzzle, use two strategies involving different degrees of planning. The comparison of the lengths of solution paths produced by their subjects with those produced by ours suggests that the subjects in our experiment did emphasize planning (see the Pyramid Solutions of the 15-Puzzle section for more details). The time required for the solution and the sequence of the subject's moves (henceforth, called a *solution file*) were stored. Each subject solved the same set of problems in the same order. The subjects were not given any feedback about their performance.

**Results and Discussion**

Subjectively, the 15-puzzle, like the TSP, is fairly easy. An examination of the solutions indicated that the subjects solved the problems by putting the individual tiles in place successively. First, the subject put Tile 1 in the top left corner. Then Tile 2 was put next to Tile 1, and then 3 and 4. Completing each row poses additional challenges, as most readers probably know from their own experience. This method is not likely to produce the shortest solution path, but it always allows reaching the goal state. Similar results for the case of the 8-puzzle were reported by O'Hara and Payne (1998).

Figure 2 shows the relation between the time and the length of solution for each subject. The length of the solution ranged between 40 and 160 moves, and the total time of solution ranged between 30 and 180 sec. The performance of each subject varied substantially across the five problems. This does not mean, however, that the five problems were different with respect to difficulty, because each problem led to very different solution lengths across the 4 subjects (see Table 1). Large variability (up to a factor of three) in the solution length among the subjects with the 15-puzzle contrasts with very small variability among subjects with the TSP (see Graham et al., 2000). None of our subjects was clearly better or clearly worse than any other subject: Each subject (as compared with the other subjects) produced the best solution at least once, and each produced the worst solution at least once.

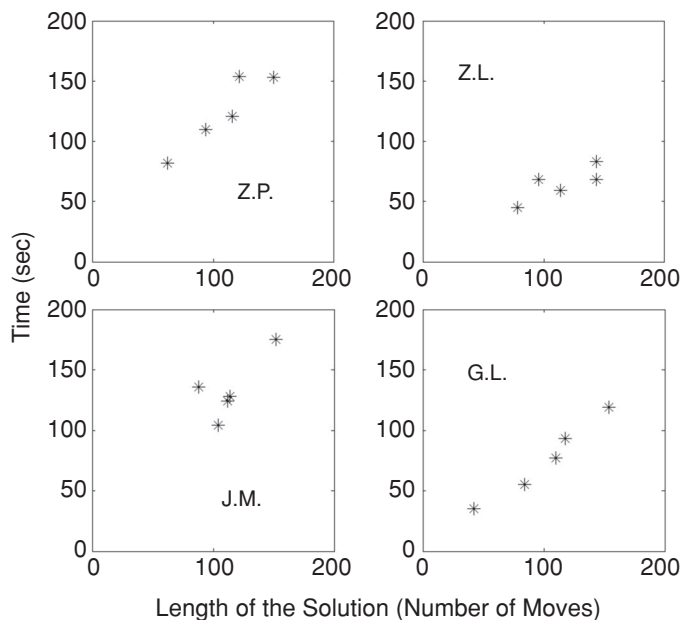
Optimal (shortest) solutions for individual problems were not determined, because finding the optimal solution path for each problem would require searching through a very large number of paths. Optimal solutions for at least some

**Table 1**  
**Solution Lengths (in Number of Moves) in Experiment 1**

	Subject			
	Z.P.	Z.L.	J.M.	G.L.
Problem 1	122	144	88	110
Problem 2	62	144	112	84
Problem 3	94	114	114	154
Problem 4	150	78	152	118
Problem 5	116	96	104	42

15-puzzles have been published in the past. The length of the optimal solution for randomly generated instances of the 15-puzzle was between 41 and 66 states (Korf, 1985), but the number that had to be visited by Korf's algorithm ranged between 540,000 and 6 billion states. Considering the large variability in the solution length for a given problem across subjects in our experiment, it is clear that the solutions produced by the subjects were usually not even close to optimal. However, since the subjects performed almost no backtracking (see below), it follows that the number of states that were visited by the subjects was fewer than 200. Thus, even though the subjects did not produce the shortest solutions, they got to the goal by visiting far fewer states than did the algorithm that produced the shortest path. If *optimal* meant getting to the goal quickly, the human performance would have to be considered far more optimal than the performance of Korf's algorithm.

The slopes in Figure 2 represent the speed with which a given subject generated the moves. Some subjects solved the problem by performing about two moves per second (Z.L. and G.L.), and others performed one move per second (Z.P. and J.M.). The subjects performed almost no backtrack-



**Figure 2. Relation between time and length of the solution for the 15-puzzle in Experiment 1.**

ing (undoing moves). This contrasts with results reported by O'Hara and Payne (1998). This difference between the performance of our subjects and that of O'Hara and Payne's subjects might be related to a different method of controlling the interface. In our experiment, the subjects had more direct control because they simply clicked the mouse on the tile to move it. In O'Hara and Payne's Experiment 1 (for which backtrackings were reported), the subjects used a keyboard, rather than a mouse. In our experiment, backtracking was found in 12 out of 20 solution files (there were 5 solution files per subject). The average number of backtrackings per solution, computed from solution files in which backtrackings occurred, was 1.6. This average means that the number of backtrackings was usually one or two per solution. The average length of backtrackings was 1.7, indicating that when they occurred they were very short, usually of the length of one or two. The fact that backtrackings were rare implies that the subjects were building the solution path by transforming the start state systematically toward the goal state and did not explore more than one path. These results strongly suggest that the subjects did not perform an explicit search while solving the problem.

A search performed by planning moves, without actually executing them (implicit search), was probably not performed either. Implicit search would be likely to translate into a speed-accuracy trade-off. That is, longer solution times would lead to shorter solution paths. Such trade-offs were not observed. Note also that implicit search, even if utilized, could never involve substantial portions of the problem space, due to limitations of working memory (Cowan, 2001). The apparent absence of search (implicit or explicit) implies the use of a low-complexity mental algorithm.

To test the computational complexity of the mental mechanisms involved more directly, we tested the same subjects on the  $n$ -puzzle, where  $n$  was 5, 8, 15, and 35, and we analyzed the length and time of solution as a function of the problem size. What is the lower bound for the complexity of an algorithm that solves the  $n$ -puzzle? Consider a relaxed version of the 15-puzzle in which the moves of each tile are not constrained by other tiles. Consider Tile 1. Assume that this tile has to be moved from the bottom right corner to its goal state in the top left corner. The number of moves (and thus time) is approximately proportional to the length of the diagonal of the board on which the puzzle is presented. For a puzzle with  $n$  tiles, assuming that the puzzle is presented on a square board, the diagonal is approximately proportional to  $\sqrt{n}$ . Since the number of tiles is  $n$ , the total time is proportional to  $\sqrt{n}^3$ . Since we assumed a completely relaxed version of the problem, the actual time cannot grow slower than that with  $n$ .

## EXPERIMENT 2 The Effect of Problem Size

### Method

**Stimuli.** Four sizes of the  $n$ -puzzle were used:  $n = 5, 8, 15,$  and  $35$ . For each size, five problems were used. The start states were produced by applying 1,000 random moves to the goal state. Such a

large number of random moves was used to make sure that the starting states were sufficiently scrambled for all problem sizes.

**Subjects.** The same subjects as those who were tested in the first experiment served here.

**Procedure.** Once again, each subject solved the same set of problems in the same order.

### Results and Discussion

Figure 3A shows the relation between the solution time, averaged over five problems of a given size, and the corresponding average of the solution length for each subject. Log-log coordinates were used. In these coordinates, proportionality between the solution time and the solution length would be represented by a diagonal line with a slope of one. It can be seen that the data points for each subject form an approximately straight line with a slope close to one. There is some variability among the subjects, but note that the performance of the 2 naive subjects is quite similar to the performance of the 2 authors. This means that knowledge of the hypotheses being tested did not seem to affect performance. The slope of one is consistent with the results from Experiment 1, where an approximate proportionality between the solution time and the solution length was also observed. However, the results of the second experiment were not a mere replication of the results of the previous experiment. The range of solution lengths and solution times was much larger in the second experiment simply because problems of different sizes were used. The proportionality between the solution time and the solution length means that each subject had his own "pace" when solving the problems and that this pace did not depend on the solution length or the problem size. This fact suggests the use of a low-complexity mental algorithm with little implicit search.

The results shown in Figure 3B address the complexity of the mental mechanisms more directly. Here, the average solution time is plotted against the problem size in log-log coordinates. A power function with an exponent of  $r$  would be represented in this graph by a straight line with a slope of  $r$ . Theoretical considerations presented in the previous section implied that  $r \geq 3/2$ . The dotted line shown in Figure 3B has a slope of 2, and it can be seen that the experimental points correspond to a slope close to 2. Clearly, the complexity of the mental mechanisms is very low, and it is close to the lower bound estimated from a completely relaxed version of the problem.

The fact that the complexity of the mental mechanisms is low suggests that the subjects did not search much when solving the problems. A question then arises about the nature of the underlying mental mechanisms: Do subjects estimate distances among the states and then use them in deciding which paths are more "promising," as is done in typical AI search algorithms (Korf, 1985; Nilsson, 1971)? These algorithms use an evaluation function, which is the sum of two terms: *depth* and *heuristic*. Depth is the distance between the start state and a given state, and a heuristic is an estimated distance between the given state and the goal state.<sup>2</sup> If both depth and heuristic are accurate estimates of the shortest distances for all states in the problem, using this evaluation function guarantees

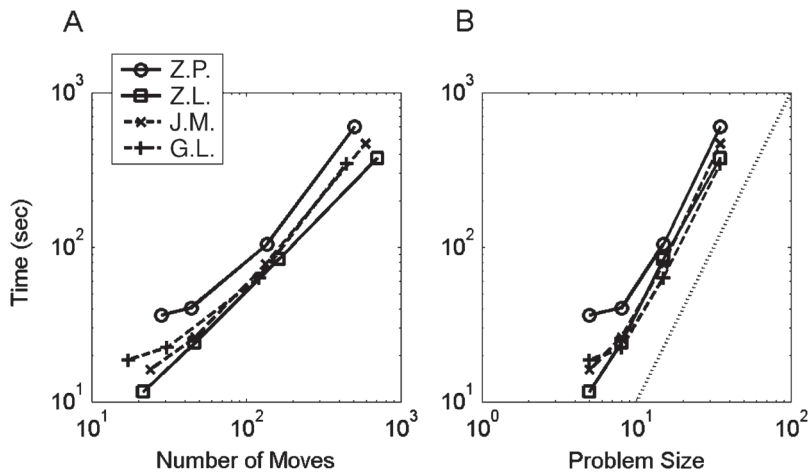


Figure 3. Results from solving the 5-, 8-, 15-, and 35-puzzles in Experiment 2. (A) Time versus solution length. (B) Time versus problem size. The results were averaged for each subject.

finding the shortest path from the start to the goal. Specifically, it was proved that the shortest path is produced if a problem solver always chooses states for which the evaluation function has a minimal value. Obviously, the evaluation function is never known exactly. But if good approximations are known, the evaluation function may still lead to good solutions (see Nilsson, 1980, pp. 72–88, for examples of this approach in the case of the 8-puzzle). Do humans use an evaluation function, which critically depends on estimated distances, or do they use other features characterizing the problem representation that allow them to build the solution path without considering other alternatives?

### EXPERIMENT 3

#### Judgments of Global Distances and Directions

In this experiment, we asked the subjects to estimate the distance between a randomly selected state from their solution file and the goal state. If the subjects can make these judgments reliably, one could conclude that the distances were (or at least could have been) used to solve the problems. However, if the subjects cannot reliably estimate distances, it means that they used a method that did not involve distances. Since the solution process did not involve search, the subject must have had a reliable rule (criterion) to decide whether a given state should be on the solution path from the current state to the goal state. To measure the reliability of such a rule, we performed a second test. The subject was shown three states: A, B, and X. States A and B were taken from the subject's solution file, and X was either a state from the same solution file or not. The subject was asked whether X was in the *direction* from A to B. By *direction* from A to B was meant *on the path from A to B*.

The concept of direction has been mentioned a number of times in the literature on problem solving (e.g., Bartlett, 1958; Maier, 1930), but it has never been for-

mally defined. Students of problem solving probably have assumed that its intuitive meaning was clear enough. Giving someone a direction for how to solve a difficult problem seemed analogous to giving spatial directions for how to find a place in an unfamiliar city. When the task itself involves spatial relations, such as maze learning, these two meanings of direction become equivalent. As a result, when a rat finds a shortcut to the goal, it makes perfect sense to claim that the rat has figured out the direction to the goal (Tolman, Ritchie, & Kalish, 1946). The concept of direction is perhaps clearest in the case of a Euclidean plane. Let us choose two points on a plane and set the task of finding the shortest path from one point to another. Despite the fact that there are infinitely many lines that can be drawn between these two points, one need not try any of them; in fact, one need not measure any distances either. The shortest path can be produced easily by drawing a straight line between the two points. In this case, the distance minimization problem is solved without measuring any distances! In this article, *direction* refers to some property inherent in a problem that allows a problem solver to determine where to go next, without considering alternatives (i.e., without performing search). One of the goals of this study is to verify whether human problem solvers have the ability to use direction (as opposed to distance) in solving problems and to propose a plausible theory for how direction is determined by humans. In this experiment, we tested the subjects' ability to judge distances and directions for states that were far apart. Hence, we call these judgments *global*. In the next experiment, we tested local judgments.

#### Method

**Stimuli.** The stimuli were states of the 15-puzzle from Experiment 1.

**Subjects.** The 4 subjects who were tested in Experiments 1 and 2 also served in this experiment.

**Procedure.** Two tasks were used: heuristic and direction (following the AI terminology, we define a heuristic as a distance to the

goal; see Nilsson, 1971). In the heuristic task, the subject was shown a randomly selected state from his solution file and was asked to estimate the distance (number of moves) to the goal state.<sup>3</sup> The first author was also tested in a *depth* task, in which he estimated the distances from the start state. The results in the depth and the heuristic tasks were very similar. Therefore, we decided that the other subjects would be tested in the heuristic task only. Each subject completed five sessions—one session per problem. Each session contained 80 randomly selected states, or all states if the solution was shorter than 80 moves. The subject responded by typing the number representing the estimated distance to the goal (no feedback was given to the subject). For comparison, each subject performed an identical task with the scramble file. Recall that the scramble file contains the sequence of 80 states that were produced by starting with the goal state and performing 80 random moves. If the scramble file is “played” backward, it can be viewed as a solution of the corresponding puzzle. However, this solution was quite different from the solutions produced by the subjects.

In the direction task, the subject was shown two states randomly selected from one of his solution files, with the constraint that the number of moves (distance) that separated the two states was 10, 20, 30, 40, or 50. These two states were reference states. In 50% of the trials, the test state was selected from the same solution file, subject to the constraint that the test state was between the reference states. In the remaining 50% of the trials, the test state was randomly selected from one of the subject’s other four solution files. The subject’s task was to judge whether the test state was on a solution path connecting the two reference states. No feedback about the correctness of the response was provided. Each subject completed five sessions, 50 trials per session. Each session included reference

states from all five solution files of the subject. Restricting a given session to one solution file was not desirable, because the repeated exposure to the reference states from a given file would likely help determine whether or not the test state came from the same file. As with the heuristic task, each subject ran another set of five sessions in which he was tested in the direction task, using scramble rather than solution files.

## Results and Discussion

The heuristic task proved very difficult and unnatural to the subjects—so much so that it was practically impossible for the subjects to estimate the absolute distance from the current state to the goal. This required trying some alternative. This took the form of suggesting that the subjects estimate the relative distances, instead of trying to estimate absolute distances. For example, assuming that the puzzle was solved in 80 moves, the subject tried to estimate how many moves would have been needed to get from a given state to the goal. Figure 4 (left panels) shows the relation between the estimated distance to the goal for states from a solution file for Problem 4 and the actual number of moves that the subjects performed while solving this problem. The squared correlation coefficient between the estimated and the actual distance to the goal ranged between .7 and .95 (Figure 5 shows correlation coefficients for all five problems). However, these correlations do not imply that the judgments of the relative

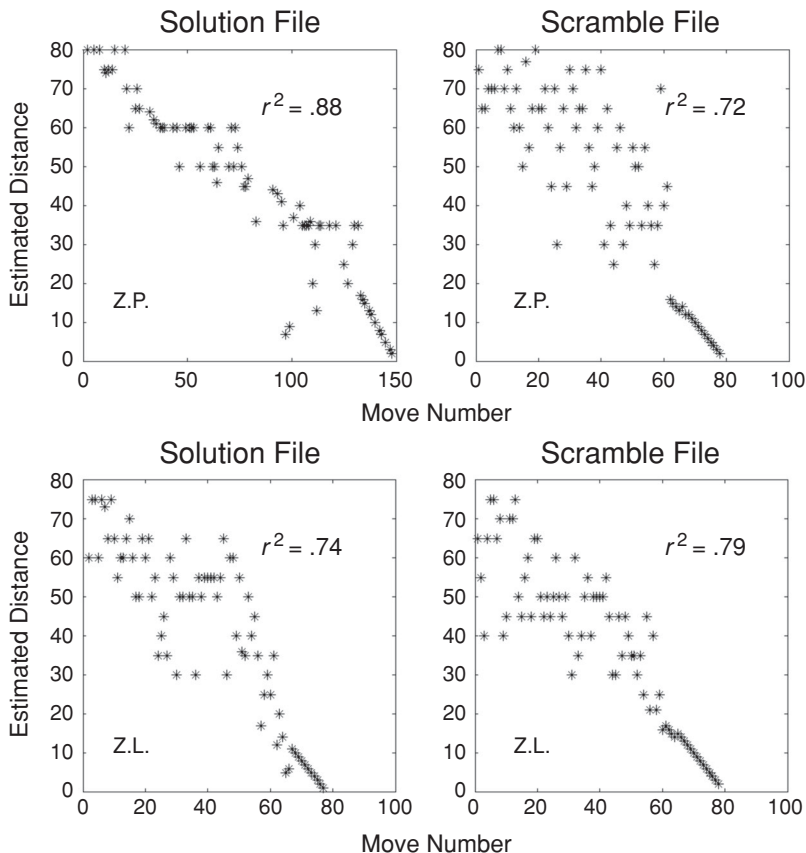


Figure 4. Judgments about the distance to the goal (heuristic) in the case of Problem 4 in Experiment 3.

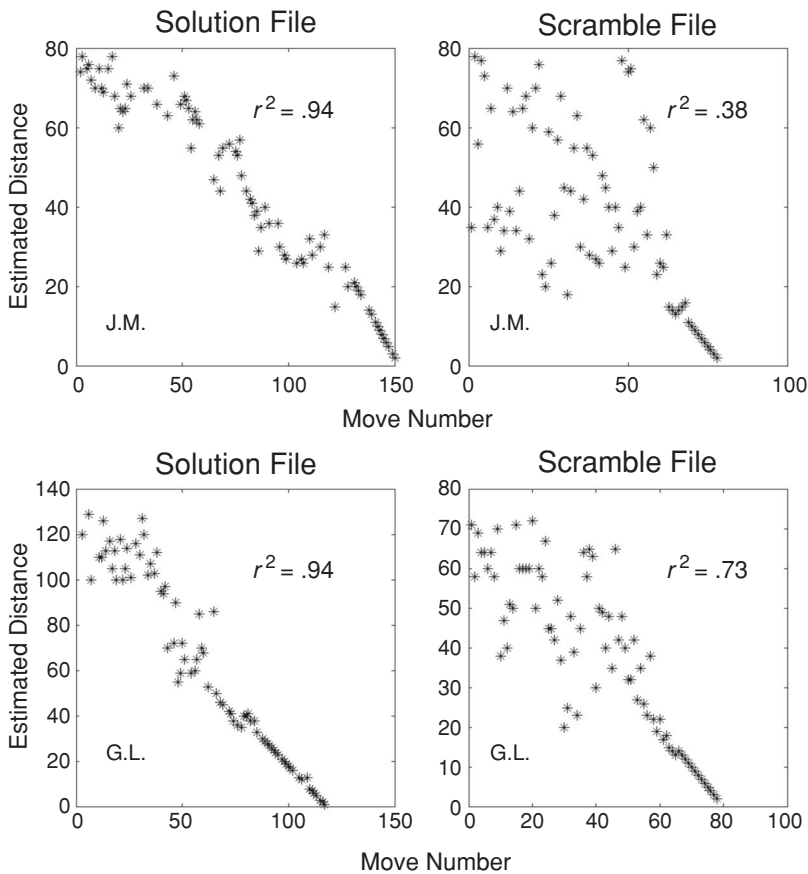


Figure 4. (Continued).

distance to the goal were very reliable. By looking at Figure 4, it can be seen that states that were very close to one another in the actual solution were often judged as being 10 or more moves apart. This represents an error of about  $\pm 10\%$ . Such poor accuracy could not be the basis for a search-free solution. For states that were very close to the goal, the subjects were able to count the number of moves that were needed to complete the puzzle. Error-free counting was performed for states as far as 20 moves from the goal. The subjects were able to do so probably because the last 20–30 moves usually involved not more than five tiles and the moves produced quite systematic transformations of the tiles. For all other states, the subjects simply relied on the number of tiles that were in place, and on the basis of this information, they judged the proportion of the puzzle yet unsolved. Note, however, that this strategy could produce only an *order* of distances to the goal state, not the actual distances. But ordinal transformation of distances does not guarantee satisfying distance axioms. In other words, numbers representing the order of distances are not distances themselves.

In the case of scramble files, the heuristic task was substantially more difficult. The subjects were quite uncertain about the distance to the goal, except for the states that were close to the goal state. This is shown in Figure 4 (right panels), where performance of the subjects

in the case of Problem 4 is presented. The squared correlation coefficient between the estimated and the actual distances to the goal ranged between .25 and .8 (see Figure 5). Since the scramble file was produced by applying random moves, the subjects were usually unable to infer the order of distances to the goal from the proportion of a problem solved. Without the information about order, the distance judgments were almost completely random. *This fact strongly supports the claim that the subjects are not able to judge distances in the 15-puzzle.* They can only judge the order of distances, provided that the states came from the subjects' solutions.

Figure 5 compares the squared correlation coefficients computed between the judged and the actual distances to the goal for solution and scramble files. It can be seen that the correlations are substantially lower in the case of scramble files. Furthermore, there does not seem to be any correlation among the subjects, nor is there an indication of a relation between the correlation coefficients shown in this figure and the solution lengths shown in Table 1.

Next, consider the direction task. The direction task was natural and easy: The proportion of correct responses was quite high for all five problems (see Figure 6). Interestingly, the performance with the scramble files was not much worse than that with the solution files. Furthermore, for both the solution and the scramble files, performance

did not depend strongly on the distance between the reference states. The small difference in performance between the solution and the scramble files in the case of the direction task contrasts with a large difference in performance between the solution and the scramble files in the case of the heuristic (distance) task. Thus, even though both direction and distance judgments are based on similarities among the states (only similarities among the states are available), they must involve different aspects of similarity. It has been known for a long time that similarities allow reconstruction, at least in principle, of distances (Shepard, 1962). Our results show, however, that *humans are not very reliable in reconstructing distances*. Furthermore, our results show that *there is another aspect characterizing the problem space, different from distances, which can be reconstructed from similarities, and this other aspect (called direction) is, in fact, reconstructed by humans quite reliably*. This observation is analogous to that formulated by Richards and Koenderink (1995) in the case of color and texture spaces.

To summarize, the results of this experiment showed that distance was not a likely source of information that could be used by subjects to solve the 15-puzzle, simply because distances were not reliably estimated by subjects. It is more likely that subjects used a different property, called here *direction*, to build the solution path, because

judgments of direction are very reliable. Note that in order to build the path to the goal, while at the same time avoiding search, the subject has to have a reliable cue for "where to go next." To verify whether subjects do have such a cue, the next experiment tested direction judgments for small distances among the states.

#### EXPERIMENT 4

##### Judgments of Local Distances and Directions

The previous experiment tested the ability of the subjects to judge distances and directions for states that were far apart (global judgments). This experiment tested the subjects' ability to judge distances and directions for states that were close to one another (local judgments). Obviously, the distinction between global and local is not clear-cut. We defined local as the range of distances over which the systematic decrease in the global judgment of distance to the goal was small, as compared with random variability of this judgment. For each subject and each solution file, we performed a piecewise analysis of the relation between the estimated and the actual distances to the goal (these relations are illustrated in Figure 4). The estimated distance was the dependent variable. Starting with the interval size of zero and containing the data point corresponding to the lowest move number used in the pre-

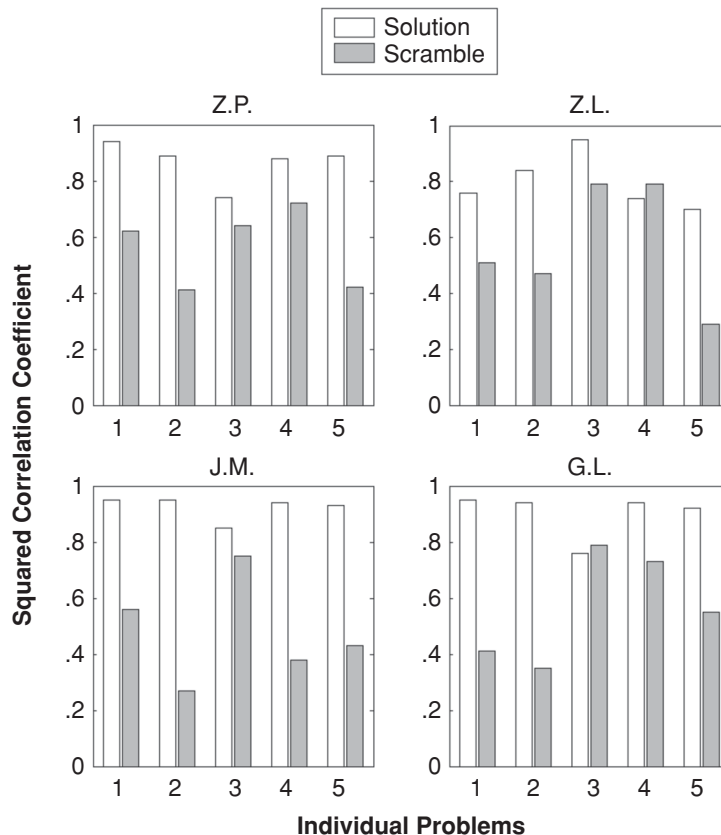


Figure 5. Reliability of judgments about the distance to the goal (heuristic) for solution and scramble files in Experiment 3.

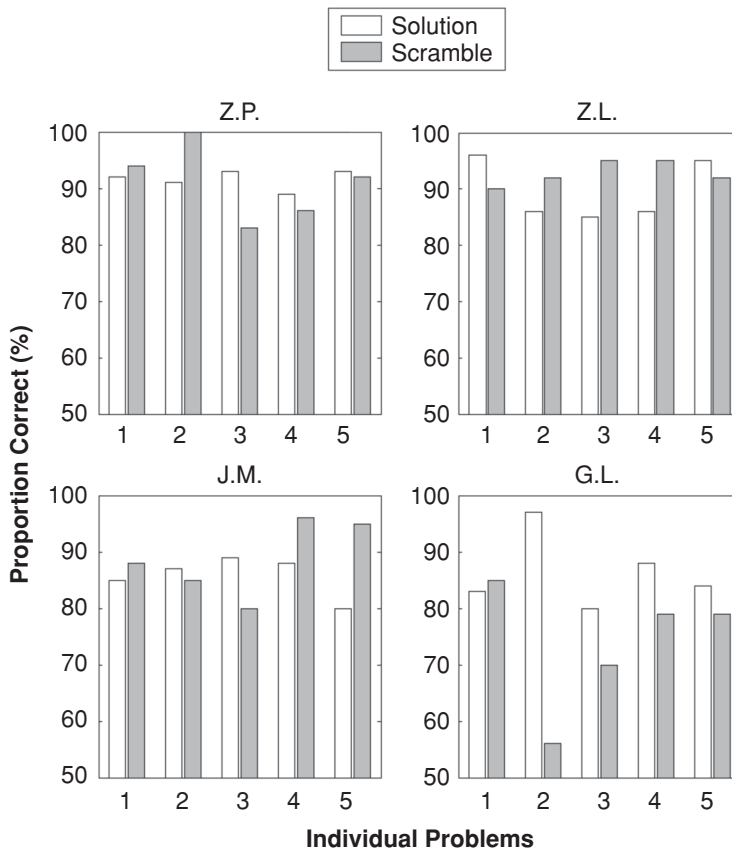


Figure 6. Accuracy of global direction judgments for solution and scramble files in Experiment 3.

vious experiment, we increased the width of the interval systematically by adding point after point and computing a linear regression over this interval. We stopped at a maximal interval width, for which the slope of the regression line was not less than  $-0.1$  (i.e., the regression line had a positive, zero, or small negative slope). Then we repeated this process, starting at the next data point, not included in the previous interval. These intervals defined *local* regions. Note that these intervals did not necessarily have the same width across problems or even within a single problem.

## Method

**Stimuli.** The stimuli were states of the 15-puzzle from Experiment 1. In the distance task, all three states were selected randomly from the same solution file of a given subject. From trial to trial, the solution file from which the states were selected was chosen randomly. The state shown on the top of the screen was the reference state. The test state on the bottom left was  $n$  steps ahead of (or behind) the reference state, and the test state on the bottom right was  $n \pm 1$  steps behind (or ahead of) the reference state. The test state on the left was closer to the reference state with a probability of .5. The distances between the reference state and the test states were determined randomly, subject to the constraint that all three states were within the same local range (defined above) and that no distance was smaller than 4. The average distance  $n$  was 6.2 moves. After running some preliminary tests, we realized that the city block distance

between the positions of the empty space in the reference and each of the test states could be used as a cue in the distance task: A larger distance between the positions of the empty space was associated with a larger distance between the states of the 15-puzzle. To eliminate (at least partially) this cue, we equalized the average distances between the positions of the empty space in the two test states, relative to the position of the empty space in the reference state.

In the direction task, the reference state and one of the test states were selected randomly from the same solution file, subject to the same locality constraint as in the case of the distance task. This test state was  $n$  steps ahead of the reference state (the average distance  $n$  was 6.6 moves). The other test state was obtained by performing  $n$  random moves, starting at the reference state. So both test states were  $n$  moves ahead of the reference state, but only one of the test states (chosen randomly from trial to trial) was on the subject's solution path in the direction toward the goal. All other aspects of the direction task were identical to those of the distance task. The distance and direction tasks involved a session with 50 questions each. All questions for a given subject involved solution files of that subject. In other words, the subject was tested on his solutions files. We assumed that performance in this task would not be affected by the memory of the states. The control experiment (reported below) showed that this assumption was justified.

**Subjects.** The 4 subjects who were tested in Experiments 1–3, served in this experiment.

**Procedure.** In the previous experiment, the response time was unlimited, so the subject was allowed to take as much time as needed to answer each question. As a result, some response times were quite long, especially in the distance task and in trials involving states that

were close to the goal. Recall that for such states, the subjects were trying to count the number of steps, rather than to infer the distances from the similarities between states. Counting the number of steps is an unlikely mechanism for solving the 15-puzzle, simply because the actual speed of solution (less than 1 sec per move) is much higher than the speed of counting the moves. To make the present test more closely related to the actual solving of the 15-puzzle, we introduced a deadline for the exposure duration of the stimuli in both tasks. The deadline was 10 sec. In preliminary tests, we found that this deadline was sufficient to produce reliable performance in the direction task. At the same time, it was not long enough for the subjects to count the number of steps between the reference state and each of the test states.

The subject's task was to judge which distance is greater (in the distance task) or which state is on the solution path (in the direction task). The subject's performance was evaluated by computing the proportion of correct responses. The overall proportion correct is known to be fairly insensitive to the bias for choosing one of the two possible responses. The lower bound for the proportion correct in both tasks was .5 (chance level).

## Results and Discussion

The results for both tasks are shown in Figure 7, where it can be seen that all 4 subjects were able to judge direction, but not distance, reliably. The subjects' performance, except for J.M.'s, in the distance task, was close to chance level. J.M. reported that he explicitly tried to count the number of moves between the three states and that he was sometimes able to perform the count before time had elapsed. Other subjects also tried to count the number of moves, but they almost never succeeded. Without counting, the subjects were unable to infer the distance precisely between states on the basis of comparing their similarities. It follows that it is rather unlikely that the subjects relied on distance judgments to solve these problems.

The high level of performance observed in the local direction task clearly suggests that subjects have a reliable rule for deciding about the next move(s) and, thus, for building a path to the goal state. This rule is not likely to be based on distances. Instead, it is based on some property related to what we call *direction*. Now, *if direction is actually reconstructed from similarities among the states of the problem, we would expect a high degree of consistency among subjects in judging the direction*. To verify this prediction, we performed the following control experiment.

### Testing the Consistency Across Subjects in Using Direction

#### Method

Each of the two authors ran the direction test on the other 3 subjects' solution files. Specifically, each subject ran three sessions in which he judged directions, using the states and paths from solutions produced by the other 3 subjects. In short, we were asking whether a subject is able to "understand" another subject's rule for where to go next, despite the fact that their solutions were quite different.

As a comparison, we also used scramble files in this control experiment. Recall that scramble files contain the sequence of 80 random moves that were applied to the goal state in order to produce the start states in the problems that were solved by the subjects. Even though the scramble file contains random moves, when the scramble file is viewed in a reverse order, the moves represent a solution of the problem, except that this solution was not produced by a human

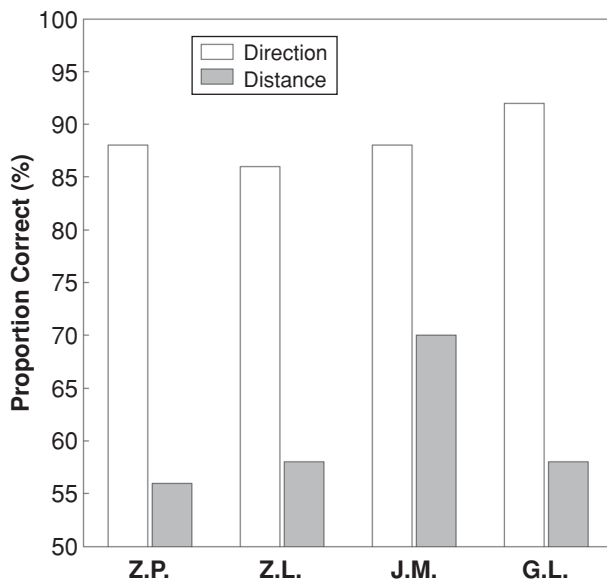


Figure 7. Discrimination of directions and distances in Experiment 4.

subject. In Experiment 3 we found that the subjects were able to judge global directions in the case of scramble files almost as reliably as in the case of their solution files. Are they able to produce equally high performance in a local direction task? To make the comparison between the performances on solution and scramble files as close as possible, the distances between the test and the comparison states in the case of scramble files were kept very close to those of the solution files.

## Results and Discussion

Figure 8 shows that the subject's performance on the others' data was not worse than his performance on his own data. But, the subject's performance on the solutions represented by the scramble files was substantially lower (although it was still above chance level). These results show that one subject can "understand" another subject's solution, as well as his own solution, but that this high level of performance does not generalize to an arbitrary solution, such as the one represented by a scramble file. These results also show that the high level of performance in the direction test shown in Figure 7 was not due to memorizing the states from the subject's own solutions.

To summarize, results of this control experiment strongly support the claim that all our subjects used a very similar (perhaps even identical) rule to solve the 15-puzzle. Our results from previous experiments reported in this article also suggest that this rule is not related to distance.

Having collected experimental evidence suggesting that human problem solvers use similarities among states to infer some abstract property (direction), which is quite different from distances, a theoretical question arises about the nature of this abstract property. What can be computed from a graph such that it does not necessarily satisfy metric axioms but is strong enough to effectively guide the human problem solver from the start to the goal, without backtracking and search?

**GRAPH PYRAMIDS**

We conjecture that the main aspects of human performance described above can be adequately modeled by a pyramid algorithm. Recall that conventional pyramid algorithms operate on images. But one may also apply a pyramid algorithm to a graph, rather than to an image (Kropatsch, Leonardis, & Bischof, 1999; Pizlo & Li, 2003, 2004). Note that if the representation of a visual image is based on discrete sampling (which is the case in biological systems, as well as with cameras), the image can be treated as a graph, whose nodes are positions of pixels or receptors. In our study, the nodes in the graph represent the states in the  $n$ -puzzle, and the edges represent legal moves.

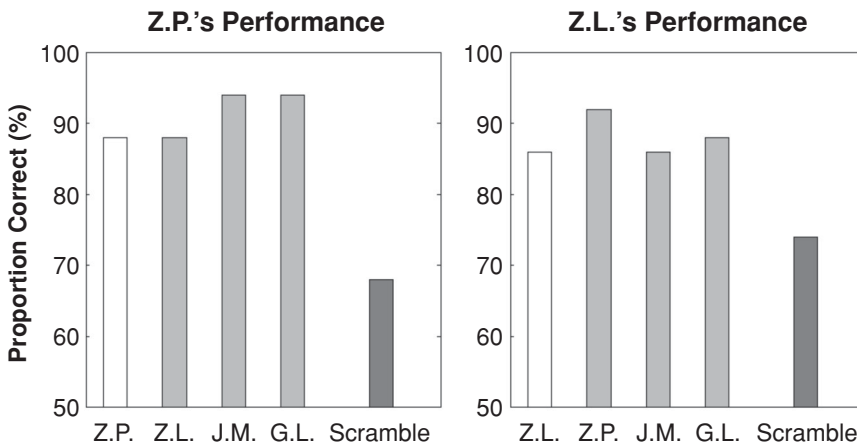
**Pyramid Solutions of the 15-Puzzle**

The first question is how to build a pyramid representation of the 15-puzzle. Recall that a pyramid algorithm stores multiple representations of a problem, organized in a stack of layers. The layers differ from one another with respect to scale and resolution. More exactly, each layer has nodes that store information about the original graph. The number of nodes on layer  $k$  is  $N/b^k$ , where  $N$  is the number of nodes in the original graph and  $b$  is called a *reduction ratio*. It follows that the number of nodes decreases exponentially with  $k$ . This means that nodes on higher layers have larger receptive fields; that is, they represent larger parts of the original graph. Because the “memory” of nodes is limited, nodes on higher layers do not store complete information about the part of the graph that falls within their receptive fields. In our algorithm, each node stores information about direct connections to neighboring nodes (called *siblings*) on the same layer and about shortest connections among its *children* that are within its receptive field on a lower layer.

The next question is how the representations on higher layers should be produced. In the case of graph pyramids, nodes on higher layers represent clusters of nodes on

lower layers. There is more than one way to form clusters (group states together). The choice is important because the graph representing the 15-puzzle is very large: The number of nodes is equal to  $16!/2 \approx 10^{13}$ . This number is greater than the number of neurons in the human brain. Clearly, it is impossible for a human subject to store all states of this puzzle simultaneously in memory. The clustering should be performed in such a way that the problem solver can work on a very small fraction of the entire problem at a time. In fact, the method commonly used by the subjects indicates such an efficient clustering. Consider the 16 positions of Tile 1 on the  $4 \times 4$  board, with an arbitrary arrangement of the remaining Tiles 2–15, and the empty space. Each of those 16 positions of Tile 1 represents a cluster of  $15!/2 \approx 10^{11}$  states. Furthermore, those 16 clusters do not overlap. The goal state is in the cluster represented by Tile 1 being on the correct (goal) position. So one can begin the solution of the problem by changing the position of Tile 1 until it ends up in its goal position. This will correspond to finding a path through the 16 clusters. In this approach, the cluster corresponding to Tile 1 being in the top-left corner of the board is a subgoal. The subjects in our experiments always started with achieving this subgoal. Note that this subgoal is not a particular state but, instead, a cluster of states that should be visited on the way to the goal.

Now, how does the model determine the sequence of moves that bring Tile 1 to the top left corner? If Tiles 2–15 are considered indistinguishable, the 15-puzzle has only  $16!/14! = 240$ , rather than  $10^{13}$ , states. Our model builds a complete pyramid representation for this graph by performing hierarchical clustering of nodes and then finds a path by using a top-down process of successive approximations (see the Appendix for a pseudocode, and Pizlo & Li, 2003, for examples in which this algorithm is applied to random graphs). The hierarchical clustering consists of constructing smaller and smaller graphs in such a way that (1) nodes on higher layers represent clusters of neighboring nodes on the lower layers and (2) edges on higher lay-



**Figure 8.** Performance of 2 subjects in the direction task on other subjects' solution files in the control experiment.

ers represent direct connections between pairs of clusters on lower layers.

We tried several ways of choosing the neighboring nodes to form a cluster and found that the solution paths were not very sensitive to the method of clustering. The top-down process begins at a coarse (global) representation of the graph, and in this representation, there are only a few nodes (each node corresponds to a cluster of states in the graph). As a result, a decision about a path is easy. This path is then projected to the next lower layer, where the representation is less coarse. This representation has more nodes, but not all nodes have to be considered. The path found on the higher layer provides information about which nodes on the lower layer should be considered for possible inclusion in the next approximation to the solution path. This path is then projected to the lower layer and so on until the bottom layer of the pyramid is reached. At that point, the solution path for Tile 1 is found. This process of successive approximations involves a minimal amount of search, because the more global representation is used to guide (direct) the solution process on the more local representations. We believe that this *pyramid-based algorithm is a possible model of the concept of direction in human problem solving*.

Next, the process is repeated for Tile 2, except that Tile 1 is not allowed to be moved. The same is repeated for Tile 3. In the case of Tile 4, the graph is larger, because the placement of this tile on its goal position requires that Tile 3 be moved. The placement of Tiles 5–8 is completely analogous to that for Tiles 1–4. Then Tiles 9 and 13 are put in their goal positions. The remaining tiles (10, 11, 12, 14, and 15) that are not necessarily in their goal positions correspond to a 5-puzzle, which is the smallest nontrivial  $n$ -puzzle. All five tiles must be involved in solving this part of the problem. The graph representing the 5-puzzle has  $6!/2 = 360$  nodes.

We first applied our pyramid algorithm to the same five problems on which our subjects were tested in Experiment 1. The solutions produced by the algorithm in the case of Problems 1–5 had lengths of 112, 116, 92, 112, and 82, respectively. It can be seen that the lengths of the paths produced by our model are quite similar to the lengths of the paths found by the subjects (see Table 1). Next, we applied the algorithm to the same 20 problems for the 5-, 8-, 15-, and 35-puzzles, on which our subjects were tested in Experiment 2. The solution length ranged between 10 and 18 for the 5-puzzle, between 22 and 42 for the 8-puzzle, between 108 and 132 for the 15-puzzle, and between 456 and 606 for the 35-puzzle. These results of the model are quite similar to the results of the subjects. The main difference is that the model does not seem to produce solution paths that are as long as those that the subjects occasionally produce. Specifically, the solution length in our Experiment 2 ranged between 10 and 54 for the 5-puzzle, between 22 and 72 for the 8-puzzle, between 98 and 218 for the 15-puzzle, and between 322 and 996 for the 35-puzzle. For comparison, the solution length produced by the subjects in O'Hara and Payne's (1998) experiment in the case of the 8-puzzle ranged between

24 and 61 when the subjects performed more planning and between 52 and 184 when the subjects performed less planning (the planning strategy was manipulated by the authors by changing the method of controlling the interface). It appears that the version of the pyramid model implemented by us adequately captures the human performance that involves more planning, although a direct comparison between our model and O'Hara and Payne's experimental results is not possible, because our examples of the 8-puzzle were different. Note that the performance of our model can be modified simply by changing the size of the clusters in the pyramid representation: Smaller clusters would correspond to less planning and would be likely to lead to longer solution paths.

### Psychophysical Test

If our pyramid model actually simulates the mental solution process, subjects should be able to "understand" solutions generated by the model. This was verified by testing our subjects in a direction task, using states taken from the model's solution. Three subjects were tested (Z.P., Z.L., and J.M.). The test and reference states were selected in such a way that the distances between these states were very similar to those in our Experiment 4. Performance of each of the 3 subjects on the model's solution data was 96%. This was slightly better than the subject's performance on his own data! This result provides strong support for the model's ability to simulate the subjects' mental mechanisms. The fact that the subjects understood the model's solution slightly better than they were able to understand their own solutions can perhaps be explained by the absence of occasional random decisions (errors) in the model's solutions that were likely to occur in the subjects' solutions. In other words, the model's performance is more predictable than the subject's performance.

## SUMMARY AND CONCLUSIONS

We tested human performance in a class of combinatorial problems called the  $n$ -puzzle. This puzzle is considered easy by human subjects, although the solution length for a given instance of this puzzle varies substantially across subjects. Despite the large variability in the solution length, the solution method seems to be quite consistent across subjects. Specifically, humans solve the puzzle by breaking it into a sequence of subproblems corresponding to moving the individual tiles into their goal places. Subjects do not appear to use distances among the states of the problems, and they are able to produce a solution path without performing search. Our model produces very similar solutions by using a top-down sequence of approximations based on a pyramid representation of graphs corresponding to subproblems of the puzzle.

The main difference between the  $n$ -puzzle and the E-TSP (the E-TSP was discussed in the introduction) is that humans produce near-optimal solution lengths in the latter, but not in the former. The main similarity is that the subjects do not seem to use distances and they do not

perform much search, even though both problems have enormous search spaces. Furthermore, the computational complexity of the underlying mental mechanisms is very low for both problems. As a result, the solution times are very short. Finally, in both problems, a pyramid-based algorithm, which uses hierarchical clustering, seems to be an adequate model.

To summarize, it appears that combinatorial problems are solved by humans very quickly, despite the fact that the search spaces are very large. It follows that the search of the problem space cannot be the main feature of the underlying mechanisms. Instead, an effective mental representation, which allows building the solution path in the direction toward the goal, seems to be the critical factor. We conjecture that this representation takes the form of hierarchical clustering.

## REFERENCES

- BARTLETT, F. C. (1958). *Thinking: An experimental and social study*. London: Allen & Unwin.
- BOUMAN, C., & LIU, B. (1991). Multiple resolution segmentation of textured images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, **13**, 99-113.
- COWAN, N. (2001). The magical number 4 in short-term memory: A consideration of mental storage capacity. *Behavioral & Brain Sciences*, **24**, 87-185.
- GAREY, M. R., & JOHNSON, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: Freeman.
- GRAHAM, S. M., JOSHI, A., & PIZLO, Z. (2000). The traveling salesman problem: A hierarchical model. *Memory & Cognition*, **28**, 1191-1204.
- GUTIN, G., & PUNNEN, A. P. (2002). *The traveling salesman problem and its variations*. Boston: Kluwer.
- JOLION, J.-M., & ROSENFELD, A. (1994). *A pyramid framework for early vision: Multiresolutional computer vision*. Dordrecht: Kluwer.
- KORF, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, **27**, 97-109.
- KROPATSCH, W. G., LEONARDIS, A., & BISCHOF, H. (1999). Hierarchical, adaptive, and robust methods for image understanding. *Surveys on Mathematics for Industry*, **9**, 1-47.
- LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., & SHMOYS, D. B. (1985). *The traveling salesman problem: A guided tour of combinatorial optimization*. New York: Wiley.
- MACGREGOR, J. N., & ORMEROD, T. (1996). Human performance on the traveling salesman problem. *Perception & Psychophysics*, **58**, 527-539.
- MAIER, N. R. F. (1930). Reasoning in humans: On direction. *Journal of Comparative Psychology*, **10**, 115-143.
- METCALFE, J., & WIEBE, D. (1987). Intuition in insight and noninsight problem solving. *Memory & Cognition*, **15**, 238-246.
- NEWELL, A., & ERNST, G. (1965). The search for generality. In W. A. Kalenich (Ed.), *Information processing: Proceedings of IFIP Congress* (Vol. 1, pp. 17-24). Washington, DC: Spartan.
- NILSSON, N. J. (1971). *Problem-solving methods in artificial intelligence*. New York: McGraw-Hill.
- NILSSON, N. J. (1980). *Principles of artificial intelligence*. Palo Alto, CA: Morgan Kaufmann.
- O'HARA, K. P., & PAYNE, S. J. (1998). The effects of operator implementation cost on planfulness of problem solving and learning. *Cognitive Psychology*, **35**, 34-70.
- PIZLO, Z., & LI, Z. (2003). Pyramid algorithms as models of human cognition. In C. A. Bouman & R. L. Stevenson (Eds.), *Computational imaging* (Proceedings of SPIE-IS&T Conference on Electronic Imaging, Vol. 5016, pp. 1-12). Bellingham, WA: SPIE.
- PIZLO, Z., & LI, Z. (2004). Graph pyramids as models of human problem solving. In C. A. Bouman & R. L. Miller (Eds.), *Computational imaging II* (Proceedings of the SPIE-IS&T Conference on Electronic Imaging, Computational Imaging, Vol. 5299, pp. 205-215). Bellingham, WA: SPIE.
- PIZLO, Z., ROSENFELD, A., & EPELBOIM, J. (1995). An exponential pyramid model of the time-course of size processing. *Vision Research*, **35**, 1089-1107.
- PIZLO, Z., SALACH-GOLYSKA, M., & ROSENFELD, A. (1997). Curve detection in a noisy image. *Vision Research*, **37**, 1217-1241.
- RATNER, D., & WARMUTH, M. K. (1986). Finding a shortest solution for the  $n \times n$  extension of the 15-puzzle is intractable. *Proceedings of the Fifth National Conference on Artificial Intelligence* (Vol. 1, pp. 168-172). San Mateo, CA: Morgan Kaufmann.
- RICHARDS, W., & KOENDERINK, J. J. (1995). Trajectory mapping: A new nonmetric scaling technique. *Perception*, **24**, 1315-1331.
- ROSENFELD, A., & THURSTON, M. (1971). Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, **C-20**, 562-569.
- RUSSELL, S. J., & NORVIG, P. (1995). *Artificial intelligence. A modern approach*. Upper Saddle River, NJ: Prentice-Hall.
- SCHOFIELD, P. D. A. (1967). Complete solution of the eight puzzle. In E. Dale & D. Michie (Eds.), *Machine intelligence* (Vol. 2, pp. 125-133). New York: Elsevier.
- SHEPARD, R. N. (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function. Part I. *Psychometrika*, **27**, 125-140.
- TOLMAN, E. C., RITCHIE, B. F., & KALISH, D. (1946). Studies in spatial learning: I. Orientation and the short-cut. *Journal of Experimental Psychology*, **36**, 13-24.
- VICKERS, D., LEE, M. D., DRY, M., & HUGHES, P. (2003). The roles of the convex hull and the number of potential intersections in performance on visually presented traveling salesperson problems. *Memory & Cognition*, **31**, 1094-1104.
- WEISSTEIN, E. W. (2003). *CRC concise encyclopedia of mathematics*. New York: Chapman & Hall.

## NOTES

1. We follow the tradition adopted in psychophysics, where subjects are identified by initials and the authors themselves are included among the subjects.
2. This definition of a heuristic was proposed by Newell and Ernst (1965).
3. This task is quite similar to the task used by Metcalfe and Wiebe (1987). The main difference is related to the fact that in their experiments, subjects were making judgments while solving problems. As a result, their experiments were directly related to metacognition (cognitive judgments about cognitive acts). Our experiment, on the other hand, seems to be less directly related to metacognition, because our subjects judged distances to the goal in a separate experiment from the one in which they solved the problems.

**APPENDIX**  
**Pseudocode of the Pyramid Algorithm**

---

This pseudocode describes building the pyramid representation and solving a sub-problem of 15-puzzle.

Data structures (data type names are capitalized):

A Pyramid is an array of Layers.

A Layer is an array of Clusters.

A Cluster (or Node, if it is on the 1st layer) represents one or a collection of states in the n-puzzle state-space.

A Cluster data structure contains information about which Nodes or Clusters it contains (its Children), which Cluster on the layer above it belongs to (its Parent), which Nodes or Clusters are adjacent to it (Neighbors), and which State it represents in the n-puzzle state-space.

A Path is an ordered list of Clusters.

Function Build\_Relaxed\_Nodes (integer Step) returns Layer {

# This function builds the 1st Layer of the Pyramid for each step in 15-puzzle

# (as described in section 6.1).

# The 1st layer consists of Nodes, which represent states in the state space,

# and edges, which represent transformations, or valid moves in 15-puzzle,

# that change one state into another.

```

l = new empty Layer
while (there are distinct states remaining) {
    state = (generate a state from the state-space of step Step)
    n = new Node
    n.State = state
    n.Children = none
    n.Parent = none
    for each node n' in l {
        if (there is a valid move between n and n') {
            add n' to n.Neighbors
            add n to n'.Neighbors
        }
    }
    add n to l
}
return l
}

```

# Each Layer generated from the above function represents a sub-problem.

# We build a pyramid and solve using the pyramid for each Layer separately,

# then stitch together the solution.

Function Build\_Pyramid(Layer bottom) returns Pyramid {

p = new empty Pyramid

add bottom to p

```

do {
    new_layer = Build_Layer(top layer of p)
    add new_layer to p
    sum = (sum of degrees for each Cluster in new_layer)
    while (sum > 0)
}

```

Function Build\_Layer(Layer previous) returns Layer {

average\_degree = (compute the average degree of Nodes in previous)

l = new empty Layer

for each cluster c in previous { #arbitrary order

## APPENDIX (Continued)

```

if (c has no parent and degree of c > average_degree) {
    c' = new Cluster
    c.Parent = c'
    add c to c'.Children
    add c' to l

    for each neighbor n of c {
        if (n has no parent) {
            n.Parent = c'
            add n to c'.Children
        }
    }
}

for each cluster c in previous { #arbitrary order
    if (c has no parent) {
        if (none of c's neighbors have parent) {
            c' = new Cluster
            c.Parent = c'
            add c to c'.Children
            add c' to l

            for each neighbor n of c {
                if (n has no parent) {
                    n.Parent = c'
                    add n to c'.Children
                }
            }
        }
    }
}

for each cluster c in previous { #arbitrary order
    if (c has no parent) {
        # one of c's neighbors must have a parent, because of the loops above
        c' = (parent of any arbitrary neighbor)
        c.Parent = c'
        add c to c'.Children
    }
}

# Now to propagate connections
for each Cluster c in previous {
    for each neighbor n of c {
        if (c.Parent not equal n.Parent) {
            add n.Parent to c.Parent.Neighbors
            add c.Parent to n.Parent.Neighbors
        }
    }
}

# Once pyramid of a sub-problem is built, we can use it to solve the
# sub-problem

Function Corridor_Search (Pyramid p, State start, State goal) returns Path {
    bottom = (bottom layer of p)
    s = (Cluster in bottom with State start)
    g = (Cluster in bottom with State goal)
    while (s.Parent not equal to g.Parent) {
        s = s.Parent
        g = g.Parent
    }
}

```

**APPENDIX (Continued)**

---

```
# Add additional code to save intermediate s and g
}

# Since s and g share same parents, they are very nearly adjacent

l = (layer containing s and g)
l' = (subgraph of l containing only children of s.Parent=g.Parent)
do {
  path = BFS(l', s, g) #BFS is any standard Breadth-first search algorithm
  if (l is bottom Layer in p) return path
  l = (layer below l)
  s = (Cluster in l containing start, using intermediates saved above)
  g = (Cluster in l containing goal)
  l' = (subgraph of l containing only children of Clusters in path)
}
# Execution never reaches here
}
```

---

(Manuscript received July 14, 2004;  
accepted for publication October 8, 2004.)