Heuristics in Problem Solving: The Role of Direction in Controlling Search Space

Yun Chu[1], Zheng Li[2], Yong Su[3], and Zygmunt Pizlo[4]

Abstract:

Isomorphs of a puzzle called m+m resulted in faster solution times and an easily reproduced solution path in a labeled version of the problem compared to a more difficult binary version. We conjecture that performance is related to a type of heuristic called direction that not only constrains search space in the labeled version, but also facilitates solution by presenting an "easy to keep in mind" destination as the goal state that does not tax working memory. Using direction makes it possible to solve a problem by building a path toward the solution rather than eliminating unpromising paths. The latter always involves search, which slows down the solution process and requires storing a large number of intermediate states in memory. Direction allows for smaller search. We speculate that discovering direction in a given search space enables operation selection and guidance in the solution path.

Keywords:

problem solving, heuristics, hill-climbing, direction, isomorphs, m+m puzzle

Zygmunt Pizlo would like to acknowledge support from AFOSR.

[1]State University of New York, Purchase College; [2]Duke University; [3]The Ohio State University; [4]Purdue University

http://dx.doi.org/10.7771/1932-6246.1078

In problem solving, heuristics play a major role in the solution process. Heuristics exist because more often than not, they aid in finding an easy path to the answer in complex problems (Renkl, Hilbert, & Schworm, 2008). However, there are instances when heuristics can be misleading and may even hinder solution (Öllinger, Jones, & Knöblich, 2006). Insight problems are sometimes known for having the appearance of a simple problem, yet commonly used heuristics are not the correct approach to finding the solution. For example, the maximization heuristic in the cheap necklace problem is one of the obstacles to solution (Chu, 2009). In the cheap necklace problem, participants are presented with four chains containing three links each. The goal is to connect all 12 links in a complete and closed necklace with 15¢. Opening a link costs 2¢ and closing a link costs 3¢. How do you make the necklace? The first impulse is to connect two chains end-to-end, making a six-link chain. This move appears to maximize the number of links connected with only 5¢. Problem solvers tend to fixate on the maximizing first move that is incorrect and guarantees failure.

The Criterion for Satisfactory Progress (CSP) theory (Chu, Dewald, & Chronicle, 2007), formerly known as the Progress Monitoring Theory (MacGregor, Ormerod, & Chronicle, 2001), suggests that the difficulty with insight problems is that, initially, general problem-solving (GPS) heuristics are employed. Individuals appear to have a set when faced with difficult problems, that is, they perform the same incorrect sequence of moves over and over again before changing their mental representation of the problem. As long as a move meets a criterion for progress (derived from the problem statement), it will be judged as satisfactory and retained. In a multiple-move problem, perceived satisfactory progress depends on the problem solver's look-ahead ability. If the problem solver can only look one move ahead, she will be satisfied with her progress until one move before the dead-end of the problem, at which point she will be able to see the impending failure. The better look-ahead a problem solver has, the earlier she will realize her current path leads to failure. Thus, she will abandon her plan and explore alternative solution paths earlier than a person with poorer look-ahead. This might not guarantee finding the correct solution, but it moves the person away from the initial set.

What mechanisms could be involved in look-ahead ability? The concept of *direction* seems to be a plausible candidate. This concept was introduced by Maier (1930) and was later used by Tolman (1932). The main idea is related to the commonsensical meaning of the word. When you are in an unfamiliar city, you usually ask for help in terms of direction. The distance to the goal would be desirable information if humans used Newell and Simon's (1972) *heuristic* approach. In their approach, a problem solver computes the sum of the distance from the start state to the current state and the estimated distance from the current state to the goal state (the latter is called the heuristic function). A hill-climbing method is often employed (Chronicle, MacGregor, & Ormerod, 2004). Under some assumptions, the problem solver should always choose the action, which leads to the state that minimizes this sum. However, if the distance from the start is not the minimal distance,

and if the estimated distance to the goal is not accurate, the problem solver may end up searching a large proportion of the problem space. This is not very effective or practical, especially if the problem space is large. Having too many solution paths make it hard to find the answer (Patra, Goswami, & Goswami, 2009). Rules of thumb (often informally called heuristics) limit the search space, thus making the solution more likely (Vrakas & Vlahavas, 2005).

Unless the problem is extremely familiar to the problem solver, at which point she may store the state space in long-term memory,[1] she cannot search large parts of the problem space due to the limitations of short-term memory. In fact, such a search is often not needed. Many problems can be solved quite well by building the solution path, rather than by exploring a large number of alternatives (Pizlo & Li, 2005). When a problem is solved by building a solution path, the solution may not be optimal (in terms of the path length), but it is likely to be economic in terms of the time spent solving the problem. Practically, what matters in everyday life is to solve problems and to solve them quickly. Note that the solution time consists of two times: the time needed to plan the solution and the time needed to execute the plan. If a search space is large, which is the case in most non-trivial combinatorial optimization problems, it may be far more time efficient to keep search to a minimum, or eliminate it altogether. Searching a substantial part of the problem space is likely to lead to arbitrarily large amounts of time spent on planning the solution. If the number of states grows exponentially with the problem size, searching a constant fraction of the problem space will lead to search time that also grows exponentially with the problem size. If a problem can be solved by building a solution path, rather than by search, then even if the solution path is highly suboptimal, the total time can be a linear (or approximately linear) function of the problem size. This is the case with such problems as the traveling salesman problem (TSP) (e.g., Pizlo et al., 2006) and the 15-puzzle (Pizlo & Li, 2005), both known to be NP-hard problems. In both of these problems, the human problem solver builds the solution path using a recursive method. We believe that in both of these cases, the problem solver has a good sense of direction. The problem solver knows where to go next so that every step brings her *closer* to the solution regardless of whether she knows the distance to the goal or not.

We propose that *direction* is a type of heuristic because it guides exploration in the search space. Heuristics, as they are understood in cognitive psychology, are likely to facilitate problem solving by providing hints about what steps to take at each stage of the problem-solving process (Özcan, Bilgin, & Korkmaz, 2008). The only formal definition of a heuristic was provided by Newell and Ernst (1965), as an estimated distance to the goal. This makes sense—if the problem solver knows the distance to the goal, it should be easy to decide what to do next so to make this distance shorter. Methods that use this approach are often called hill-climbing methods (hill-climbing assumes that a maximum of a cost function is sought; note that minimizing a distance to the goal is equivalent to maximizing the negative distance to the goal). Hill-climbing methods in problem solving

inevitably lead to search of the problem space. Search is necessary, even if all distances are known accurately, because one has to try all possible next steps in order to choose the best one. If distances are not known accurately, then even more search will be needed. Direction as a heuristic, on the other hand, is different. The concept of direction assumes that the problem solver may be able to decide the next step without checking alternative steps. This is likely to reduce or even eliminate search altogether. But can "direction" actually work? Can this concept be formalized? Is it actually different from distance?

Consider the relation between direction and distance in a simple case such as navigation on a Euclidean plane. Take two points on a plane and the task of finding the shortest path between these points. It is obvious that from among infinitely many arbitrary curves connecting these two points, a straight-line segment connecting these two points is the shortest. This *shortest* path can be "found" by simply drawing a straight line emanating from the first point in the direction toward the second point. One does not need to try other paths or measure any distances in order to know which is the shortest! Here, direction is trivial to define because the Euclidean plane allows for defining straight lines (straight lines also exist in more general geometries, such as affine and projective). Although this example looks trivial, it does illustrate an important point. If a direction can be defined in the solution space of a given problem (at least approximately), then short solutions paths can be found with a minimal amount of search. The present authors showed how this can be done in the case of the TSP, 15-puzzle, and other problems. In all these cases, hierarchical clustering and a coarse-to-fine solution process based on a multi-resolution pyramid representation of the problem was the key.

A multi-resolution pyramid is not the only way to establish and use direction, but it demonstrates that direction can be used not only in a Euclidean space, but also in abstract spaces representing puzzles, problems, and games (Pizlo & Li, 2003). It is an interesting question how direction is actually defined for a given problem or puzzle. In this paper, we discuss a puzzle, which we call "m+m," which can be presented in two ways, even though the solution is exactly the same. One of these two versions appears to be quite difficult for human problem solvers, while the other is quite easy. We measured the problem solvers' performance and argue that the difference is related to the operation of direction.

The m+m Puzzle

Like the 15-puzzle and the TSP, the m+m puzzle is a combinational problem. We produced this puzzle as a generalization of a puzzle described by Pushkin and Saltykov (1972) for the case of $m = 4$.

Start state:	XXXXOOOO
Goal state:	OXOXOXOX

The start state consists of four X's and four O's in a row. The goal state consists of alternating O's and X's in a row. The task is to rearrange the tiles from the start to the goal state in four moves. Each move consists of moving a pair of adjacent tiles left or right. The pair must be moved to the next pair of open spaces, and the move does not affect other tiles. The correct sequence of moves is shown in Appendix A. Because there are only two different types of tiles (X and O), we call this version of the puzzle *binary*.

Pushkin and Saltykov showed that this puzzle is very difficult for human problem solvers despite the fact that the search space is not very large. We found that the number of different states that can be reached in four moves is equal to 795. If a participant can visit one unvisited state per second, all states can be reached in about 13 minutes. It follows that this puzzle can, in fact, be solved fairly quickly by performing exhaustive search. This contrasts with such problems as the 15-puzzle or TSP, where performing exhaustive search is practically impossible. Pushkin and Saltykov also found that after a participant solves the 4+4 puzzle, she has great difficulty reproducing the solution. It takes a considerable amount of time to solve the puzzle for the second time. They conjectured that performing exhaustive search is the only way to solve this puzzle. That is, there is no heuristic, which can reduce the solution search. They further claimed that humans never perform exhaustive search and that this is why the 4+4 puzzle is so difficult to solve and the solution is so hard to recall. We found that the 4+4 puzzle can be generalized to the m+m puzzle, which consists of 2m tiles, half of which are X's and half of which are O's. For example, the 5+5 puzzle.

Start state: XXXXXOOOOO
Goal state: OXOXOXOXOX

We also found that the solution can be produced in m moves, where $m \geq 3$. In Appendix B, we present the proof of the solution algorithm, and we determine the number of different solutions for each m.

The m+m puzzle seemed especially suitable for addressing the questions of heuristic search in general and of *direction* in particular because we found that a change in the way the puzzle is presented to the participant changes the puzzle from a very difficult to a very easy one. We found that there is an isomorph of the puzzle that we call m+m labeled. These are the labeled version for $m = 4$ and $m = 5$:

Start state: FDHBGAEC
Goal state: ABCDEFGH

Start state: HDJBFGEIAC
Goal state: ABCDEFGHIJ

Here the tiles have unique labels (we used the English alphabet) and the task is to arrange the tiles in their natural lexicographical (alphabetical) order. All other rules are identical to those in the binary version, and one of the solutions is identical as well. Unlike the binary version, the labeled version has only two solutions for every *m*—the two solutions are always mirror symmetric to each other. With the binary and the labeled versions of m+m, we have two puzzles that look different, but are essentially equivalent, except that the binary version has more solutions for larger values of *m*. Interestingly, the labeled version is substantially easier. It seems that different physical representations of the problem lead to different mental representations, and one of them (labeled) allows the use of very effective heuristics in solving the problem. The labeled version is easier despite the fact that the number of different states that can be reached in four moves is larger (for $m = 4$, the number of states is 1844). The labeled version has more states because some states that are different in the labeled version are identical in the binary version due to the indistinguishable nature of the X and O tiles. It follows that the binary version has more cycles in its state graph.

Other studies of isomorphic problems have found that if an isomorph leads to automating the rules of the problem, it does so by lessening the working memory load and, as a result, the solution is more easily found. Kotovsky, Hayes, and Simon (1985) employed isomorphs of the Tower of Hanoi problem and found this to be the case. Using isomorphs of the Chinese ring problem, Kotovsky and Simon (1990) found that the size of the search space was not the main cause of difficulty of the problem. Zhang, Johnson, and Wang (1998) found that transfer across isomorphs is based on strategies, not on problem structures. We will investigate performance in isomorphs of the m+m puzzle in our experiment. A mixed design will present four versions of the m+m puzzle: 4+4 binary, 4+4 labeled, 5+5 binary, and 5+5 labeled. We expect that participants will solve the 4+4 puzzles faster than the 5+5 puzzles. In addition, we expect that participants will solve the labeled versions significantly faster than the binary versions due to their employing the heuristic of direction. We will investigate transfer in the isomorphs in the follow-up study.

Experiment

Method

Participants. Forty undergraduate students were tested. There were 27 female and 13 male participants. The average participant age was 22.6 years old. All participants were naïve about the hypotheses being tested and unfamiliar with the m+m puzzle.

Stimuli. The m+m puzzle was shown on a computer monitor by means of a software interface. The first screen showed the instructions for the m+m puzzle. The start state and the goal state were clearly presented and the operators (i.e., valid moves of the tiles) were

specified. The participants were instructed to click under the tiles with the computer mouse to select which pair they wanted to move. Once the pair of tiles was selected, they were instructed to click either "left" or "right" labels to move the pair of tiles to the next available opening on that side of the row. An example of a legal move was presented in the instructions. The participants were allowed to practice selecting and moving the tiles on the instructions screen. The instructions screen was self-paced. The participant was instructed to press the space bar on the keyboard when she had fully understood the problem and was ready to start attempting the solution. When the space bar was pressed, the time to solution started recording and the next screen was shown. This screen displayed the start state and the number of moves from the start state (not counting cancelled movements). The interface allowed the participants to make only legal moves, undo the moves, or start over. This information allowed the participant to verify whether the current sequence of moves was likely to lead to the solution. For example, if the current state in the 4+4 puzzle was reached in three moves, the participant knew that the next move had to reach the goal state. If the solution was not possible after the next move, the participant did not have to perform the move. Instead, the participant could undo one or more moves, or start over. If the participant did not solve the puzzle in *m* moves, she had to either start over or undo, as no more than *m* continuous moves from the start state were allowed.

Procedure. Participants were tested individually in front of the computer for a two-hour block. Each participant was tested on either a 4+4 binary problem and a 5+5 labeled problem or a 4+4 labeled problem and a 5+5 binary problem. The order of problem presentation was counter-balanced, yielding four groups. Participants were given a single 60-minute attempt (with unlimited undo's and start over's) to solve each problem. Solution time was recorded along with all attempted moves and the times of these moves. If the participant failed to solve the problem, the recorded time was 60 minutes. The total testing session lasted up to two hours; however, most participants were able to solve at least one if not both problems. Average testing time was 3300 seconds (55 minutes) total to complete both problems in one session. Some participants become frustrated when they spent an hour on a problem that they could not solve. A few participants stopped working at some point during the session. Because this was an individual testing environment, the experimenter was able to encourage them to continue trying to solve the problem. The experimenter reminded them to just try their best if they stopped working on the problem for more than a few seconds. Participants were responsive to this encouragement and continued working on the problems.

To summarize the procedure, there were three experimental factors: problem size (4+4; 5+5), presentation type (binary; labeled), and carryover (problem presented first; second). The main hypothesis is that the two different versions (presentation) of the m+m puzzle will lead to different mental representations. If this is the case, we should observe differences in performance across the two presentation types.

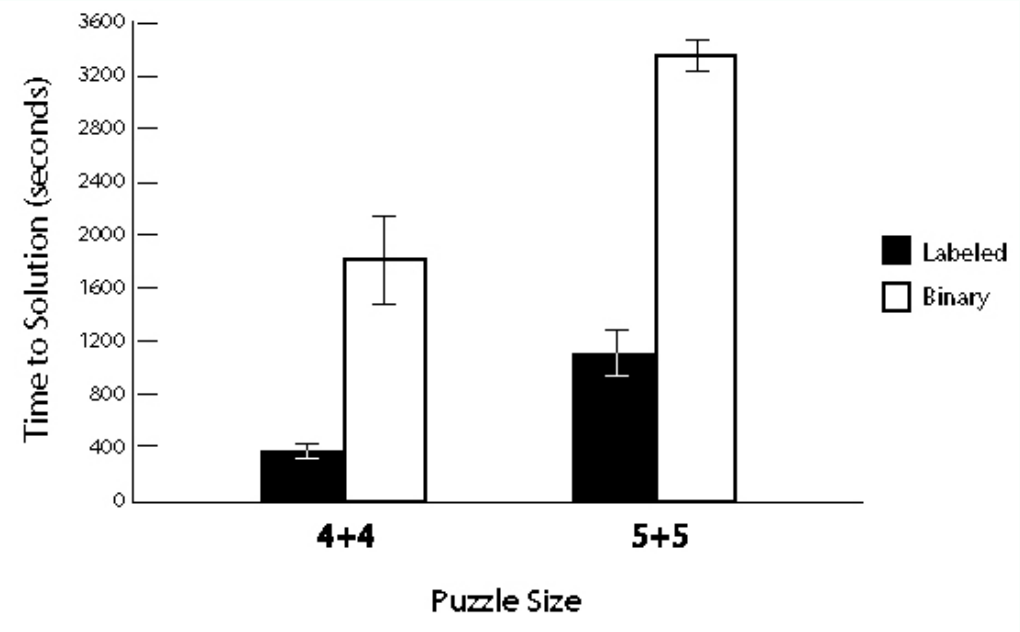
Results and Discussion

Solution Frequency. The 4+4 binary problem was solved by 70% (14/20) of the participants, while the 4+4 labeled problem was solved by 100% (20/20) of the participants. The 5+5 binary problem was solved by 25% (5/20) of the participants, while the 5+5 labeled problem was solved by 95% (19/20) of the participants.

Solution Times. The experiment was a 2 (size) x 2 (type) x 2 (carryover) partial factorial mixed design. A within-subject analysis of variance (ANOVA) was conducted on time to solution by size, type, and carryover. All means below are reported in seconds. If the participant did not solve the problem, recorded time was 3600 seconds (60 minutes). The 4+4 puzzles (mean solution time $M = 1106$) were solved significantly faster than the 5+5 puzzles ($M = 2226$), $F(1, 36) = 40.62$, $p < 0.001$. The labeled versions ($M = 732$) were solved significantly faster than the binary versions ($M = 2600$), $F(1,36) = 114.67$, $p < 0.001$. There were no carryover effects, which means that being exposed to the labeled version of the problem first did not facilitate solving of the binary version (or vice versa), and being exposed to the smaller size first (4+4) did not facilitate solving of the larger size (5+5) (or vice versa). The interactions among the three experimental factors were not significant.

The four types of problems (4+4 binary; 4+4 labeled; 5+5 binary; 5+5 labeled) yielded significantly different mean times to solution (see Figure 1), $F(3, 57) = 47.25$, $p < 0.001$, *partial* $\eta^2 = 0.71$.

Figure 1. Time to solution for the four versions of the m+m puzzle.



The 4+4 labeled problem was solved significantly faster (M = 389) than all other versions of the problem, all $p < 0.05$. The 4+4 binary problem (M = 1824) was solved significantly faster than 5+5 binary (M = 3375), $p < 0.001$, but its mean solution time was not significantly different than the solution time for 5+5 labeled (M = 1076). For the labeled versions, all participants presented with 4+4 labeled attained the solution with the slowest solution time of 900 seconds (15 minutes), and all but one participant attained the solution for the 5+5 labeled version. In contrast, for the binary versions, six participants (30%) were unable to attain the solution in 4+4, and a whopping 15 participants (75%) were unable to attain solution in 5+5. Of the five participants who attained the solution in 5+5 binary, 2 of the participants had recorded solution times of more than 3500 seconds (59 minutes). In summary, the labeled versions were solved significantly faster than the binary versions. The 4+4 puzzle was solved significantly faster than the 5+5 puzzle. The type of puzzle (binary and labeled) appeared to play a larger role on performance than the size of the puzzle (4+4 and 5+5).

Time to Correct Moves. Both the binary and the labeled versions of the 4+4 were coded for time to correct first, second, and third moves in seconds. Time to correct the second move was recorded when the correct first move was followed by the correct second move. The same procedure was followed to calculate time to the correct third move. The final move (correct fourth) was always recorded when solution was attained. Time to correct moves was significantly different among the first, second, and third moves, $F(2, 66) = 88.72$, $p < 0.001$, *partial* $\eta^2 = 0.73$. Time to the correct first move (M = 135) was significantly faster than time to any of the other moves, $p < 0.001$. Time to the correct second move (M = 562) was significantly faster than time to the correct third move (M = 610), $p = 0.032$. Time to the correct final fourth move for the solvers was reached at M = 666. The participants attained the correct first move toward the start of the solution attempt (at about the 1/5 mark), whether they solved the puzzle or not. The second and third moves were attained toward the end of the solution attempt (after the 5/6 mark).

Both the binary and the labeled versions of the 5+5 were coded for time to correct first, second, third, and fourth moves. The same procedure was followed as for the 4+4. The unsolved attempts of all versions of the puzzle were recorded for time to correct move (as many correct moves as the solvers were able to attain). For example, a failed attempt at a 5+5 binary puzzle might only have a correct first and second move. Time to correct moves was recorded for those two moves only, as no data were available for the two remaining moves (third and fourth) that were not attained. Time to correct moves was significantly different among the first, second, third, and fourth moves, $F(3, 69) = 76.16$, $p < 0.001$, *partial* $\eta^2 = 0.77$. Time to the correct first move (M = 386) was significantly faster than time to any of the other moves, $p < 0.001$. Time to the correct second move (M = 1070) was significantly faster than time to the correct third move (M = 1348), $p = 0.001$, and fourth

move ($M = 1306$), $p < 0.001$. Time to the correct final fifth move for the solvers was reached at $M = 1309$. There was no significant difference between time to the correct third move and time to the correct fourth move. In summary, the participants attained the correct first move toward the start of the solution attempt whether they solved the puzzle or not. Since there was no significant difference between time to correct third and fourth moves, the latter three moves of the solution appeared in quick succession at the end of successful solution attempts, similar to the pattern seen in the 4+4 versions.

It was reasonable to expect the 4+4 version of the puzzle to be easier to solve than the 5+5 version, and this was indeed the case. The second hypothesis was that the labeled version is easier to solve than the binary version and the experiment provided support for this hypothesis. Participants reported that after they solved the labeled version, they knew the strategy to solve the problem. However, this was not the case with the binary version. In the follow-up experiment, we investigated whether solving the same version of the problem a second time would clearly demonstrate that the participants had developed a strategy to solve the labeled version. We propose that the strategy employed is a heuristic of direction.

Follow-up Experiment

Method

Participants. Eight graduate students from Psychology and Engineering departments were tested. There were two female and six male participants with an age range of 25-30 years. All participants were naïve about the hypotheses being tested and unfamiliar with the m+m puzzle.

Stimuli. Same stimuli as in the previous experiment.

Procedure. Participants were tested individually in two sessions. Each session consisted of a two-hour block. For session 1, each participant received only one of four possible problem versions: 4+4 binary, 4+4 labeled, 5+5 binary, or 5+5 labeled. Participants were allowed 60 minutes to solve the problem. Time to solution was recorded. If the participant failed to solve the problem, a solution time of 60 minutes was recorded for that attempt. Following this first attempt, the same problem was presented to the participants. They were asked to solve the problem once again with a time limit of 60 minutes. After two attempts to solve the same version, session 1 concluded. Participants were asked to return for a second session several days later. For session 2, each participant received the alternative version of the m+m problem in both problem size and presentation type. For example, if a participant received 4+4 labeled in session 1, she received 5+5 binary in session 2. In other words, after two sessions, a participant would have received one labeled version and one binary version, as well as one 4+4 version and one 5+5 version (Table 1).

Table 1. Design for the Follow-up Experiment

	Group 1	Group 2	Group 3	Group 4
Session 1	4+4 Binary	5+5 Labeled	5+5 Binary	4+4 Labeled
Session 2	5+5 Labeled	4+4 Binary	4+4 Labeled	5+5 Binary

This was the same for the main experiment (see above). Again, the participant was asked to solve the same problem twice in session 2. Although the testing sessions were fairly long, the participants were graduate students, which may have helped the process. The participants in this experiment were more patient and tried harder than the undergraduate participants in the main experiment. The graduate students in this follow-up experiment tended to take the experiment more seriously than the undergraduates and persevered during the long testing sessions.

The participants were divided into four groups, two participants per group. The four groups represent all possible combinations of the size and version orders. To summarize the procedure, there were three experimental factors: problem size (4+4; 5+5), presentation type (binary; labeled), and learning (first attempt; second attempt). We already know that smaller problems are easier and that labeled problems are easier. The main research question was whether the solution on the second attempt was easier in the labeled version and not in the binary version.

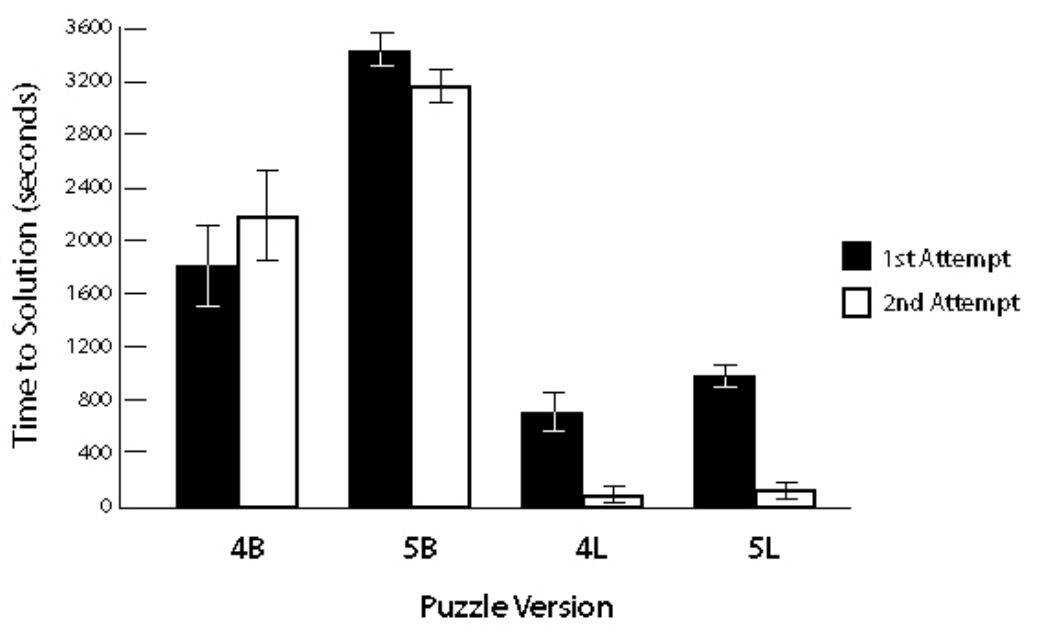
Results

Solution Frequency. The 4+4 binary version was solved 75% of the time and the 5+5 binary version was solved 25% of the time. The 4+4 labeled and the 5+5 labeled were solved 100% of the time. These results were similar to the results in the main experiment. There were no differences in the solution frequency from the first attempt to the second attempt in any of the puzzle versions.

Solution Times. Mean solution time (in seconds) for 4+4 binary was $M = 1826$ in the first attempt and $M = 2170$ in the second attempt. Mean solution time for 5+5 binary was $M = 3467$ in the first attempt and $M = 3190$ in the second attempt (Figure 2). There were no systematic differences among the four groups described in Table 1. This means that being exposed to the labeled version of the problem first did not facilitate solving the binary version (and vice versa), and being exposed to the smaller size first (4+4) did not facilitate solving the larger size (and vice versa). Therefore, the following discussion will ignore the order of conditions.

Consider first the 4+4 puzzle. It can be seen that the binary version was quite difficult and that there was no evidence of reliable learning from the first attempt to the second.

Figure 2. Mean solution time for the puzzle versions in the follow-up experiment.



Specifically, participant S4 did not solve this problem on the first one-hour attempt, and he solved it at the end of the second attempt. S1 and S2 solved it on each attempt, but one of the two participants required less time on the second attempt, whereas the other required more time. Finally, S3 solved the problem on the first attempt, but could not repeat the solution on the second one. These results agree with those reported by Pushkin and Saltykov (1972), and they quite clearly indicate that the participants were not able to figure out a systematic way to solve the problem. Instead, they seemed to perform search and stumbled on a solution accidentally. As a result, repeating the four steps representing the solution was very difficult.

The 5+5 puzzle was substantially more difficult to solve than the 4+4 puzzle. Out of a total of eight attempts made by four participants, the 5+5 binary problem was solved only twice. S6 solved it on the second attempt and S7 solved it on the first. Overall, the solution times show a floor effect in the 5+5 binary version.

Mean solution time for 4+4 labeled was $M = 713$ in the first attempt and $M = 79$ in the second attempt. Mean solution time for 5+5 labeled was $M = 942$ in the first attempt and $M = 100$ in the second attempt (Figure 2). In contrast to the binary versions, results for labeled versions were quite different. All participants found the solution on both attempts, and the solution times on the second attempts were very short. In fact, the second attempt involved checking only a few states that do not belong to the solution path. That is, on the second attempt, participants performed not many more than the number of moves in the solution path (four or five).

General Discussion

We discuss the labeled version in terms of the structure of the problem enabling heuristics for operation selection. The labeled versions of the m+m puzzle were easy to solve due to the fact that the labels provided overall direction and clear cues to the goal position of each tile. It was possible to decide which next move was the right one simply by examining the labels. Trying many different sequences of moves was not necessary. In the binary versions, even though the tiles were not exchangeable, they were indistinguishable. Specifically, each tile in the start state had a unique place in the end state in the 4+4 and 5+5 puzzles, but since all X tiles were identical and all O tiles were identical, the tiles themselves did not provide any information about their goal positions. The uniqueness of the tiles in the labeled version constrained the search space. However, the uniqueness of the tiles in the labeled version was not sufficient to solve the puzzle. The participant had to establish the relation between the structure of the problem and the role of the labels in order to attain the sense of an overall direction. Consider the results of the follow-up experiment. On average, participants spent approximately 10 minutes on the first attempt with the 4+4 labeled and 15 minutes with the 5+5 labeled puzzle. Finding the solution of the labeled version on the first attempt was equivalent to establishing how the relations among labels can be used to decide the correct next move. Specifically, each move should produce at least one pair of letters in alphabetical order. This makes sense, since the start state is a scrambled set of letters and the goal state has the letters in alphabetical order. In the labeled m+m, local alphabetical ordering builds up to global alphabetical ordering. In other words, the alphabetical order represents direction in the search space, in a way completely analogous to how direction is defined in a Euclidean space. Once the importance of the local order was found during the first attempt, it took little time to solve the same puzzle on the second attempt.

Not only did the labeled versions provide a sense of direction with uniquely labeled tiles, but the familiar alphabetical order of the goal state (ABCDEFGH) was easily remembered. It is possible that the ordered goal state in the labeled version provided an advantage over the binary version by lessening the working memory load. For example, if the goal state were to use the English alphabet in a seemingly random order (e.g., ECHAFGDB), would the labeled version still be easier to solve than the binary version? The answer is probably yes, but the problem solver would have to constantly refer back to the goal state, possibly disrupting her solution process and wasting much needed effort on this part of the task. As previously discussed, Kotovsky et al. (1985) found that a larger working memory load could make the solution harder to find. Previous studies have found that working memory plays an important role in problem solving, because the participant often has to keep a number of pieces of information in mind as she is working through the problem (Bühner, Kröner, & Ziegler, 2008). Working memory processes even differ between insight

and non-insight problem (Fleck, 2008). In addition, working memory capacity has been found to influence an individual's problem approach (Beilock & DeCaro, 2007). Therefore, the labeled versions provided an "easy to keep in mind" destination, which we believe is an essential part of direction.

In the binary version, the participants tended to repeat the same erroneous sequences of moves (as observed from the move logs). There was no evidence of learning from the first to the second attempt. The participants were unable to establish the relation between local orders and global orders. Appendix B presents the algorithm for solving the binary version, which illustrates one reason why solving the binary version is difficult. Appendix B also presents a proof of this algorithm. As always, proving a theorem is equivalent to solving a problem. In this case, proving that the solution algorithm is correct is more difficult than finding the algorithm. One of the authors (ZP) found the algorithm about 30 years ago, but was unable to prove it. Another author (YS) found the algorithm a few years ago and proved it within a week.

We conclude by conjecturing that the solution of the labeled version of m+m puzzle represents something fundamental about human problem solving. Specifically, humans are often able to produce solutions to problems rather quickly regardless of the size of the search space. The minimal role (or absence) of search can be explained only by assuming that the participants know what to do next without examining all or even most alternatives. This is accomplished by being able to judge direction in the problem space. By knowing the characteristics of the goal (its position and/or features), the participant can select the next step as the one that increases the similarity (or decrease the dissimilarity) between the current state and the goal. Dissimilarity has conventionally been modeled as a distance in some abstract spaces (e.g., Shepard, 1987). We propose that dissimilarity can also be used as a tool in establishing direction. Problem solvers often do not know how far they are from the goal, even if they are systematically approaching the goal. This was demonstrated in the case of the 15-puzzle (Pizlo & Li, 2005). Once direction is established for a given problem, the problem becomes easy. Is there a relation between discovering direction and insight? We believe that there is, but new experiments are needed to address this question.

There is one other implication from our results. Many studies of problem solving concentrate on problems that are difficult to solve, operating with the assumption that explaining the source of difficulty will tell us something important about problem solving in general. Our results suggest that it is also worth studying problems that are easy to humans, as long as the search spaces are large. Learning how humans define direction for individual problems can be at least as interesting as documenting the occurrence of insight.

Appendix A

Solution sequence for the binary 4+4 puzzle.

Start state:	XXXXOOOO
Move 1:	X___XOOOOXX
Move 2:	XOOX___OOXX
Move 3:	XOOXOXO___X
Move 4 (Goal state):	OXOXOXOX

Solution sequence for the labeled 4+4 puzzle.

Start state:	FDHBGAEC
Move 1:	F___BGAECDH
Move 2:	FGAB___ECDH
Move 3:	FGABCDE___H
Move 4 (Goal state):	ABCDEFGH

Solution sequence for the binary 5+5 puzzle.

Start state:	XXXXXOOOOO
Move 1:	X___XXOOOOOXX
Move 2:	XOOXXOO___OXX
Move 3:	XOOX___OXOOXX
Move 4:	XOOXOXOXO___X
Move 5 (Goal state):	OXOXOXOXOX

Solution sequence for the labeled 5+5 puzzle.

Start state: HDJBFGEIAC

Move 1: H__BFGEIACDJ

Move 2: HIABFGE__CDJ

Move 3: HIAB__EFGCDJ

Move 4: HIABCDEFG__J

Move 5 (Goal state): ABCDEFGHIJ

Appendix B

This appendix mainly provides a proof that the m+m puzzle can be solved in m moves for any $m \geq 3$. First, we prove a lower bound on the number of operations to solve the puzzle. Next, we derive a guideline to achieve the bound with the optimization principle. Then, a recursive solution for the general case is presented and proved by mathematical induction. With elementary combinatorics analysis, we also count the number of different solutions.

1 The Puzzle

The elements in the puzzle are simply two different tiles and we denote them by 0 and 1;* thus the puzzle is binary. The puzzle starts from a binary sequence of contiguous segments of m 0's and m 1's as shown below, where $m \geq 3$.

$$\overbrace{\underbrace{0\ldots0}_{m\ 0\text{'s}}\underbrace{1\ldots1}_{m\ 1\text{'s}}}^{2m\ \text{tiles}} \tag{1}$$

The valid operations of the puzzle are simple. Each operation is defined as shifting two consecutive tiles to the end of the sequence or the vacant places left by previous movements. For example, the following is one operation by shifting two tiles at the center to the left end of the sequence.

0 0 0 1 1 1

0 0 ␣ ␣ 1 1 0 1

* OX tiles in the binary version of the m+m puzzle are represented as 01 in this proof.

The goal of the puzzle is to reach the state in (2) from the state in (1) with no more than m operations.

$$\underbrace{\overbrace{1010\ldots10}^{m\ 10\text{'s}}}_{2m\ \text{tiles}} \tag{2}$$

The reader is cautioned that the movement of the pair of tiles can only be performed by shifting, not rotating. The shifted tiles can only be placed at the end of the sequence or the vacant places left by previous movements, not inserting between two adjacent tiles. A complete solution for the case $m = 3$ is shown in Table I. For $m = 4$, a little bit more effort will lead to the answer shown in Table II. The solutions for $m = 5$ and 6 are presented in Tables III and IV, respectively.

Table I. A Solution with $m = 3$

0	0	0	1	1	1				
␣	␣	0	1	1	1	0	0		
␣	␣	0	1	1	␣	␣	0	1	0
␣	␣	␣	␣	1	0	1	0	1	0

Table II. A Solution with $m = 4$

0	0	0	0	1	1	1	1		
0	␣	␣	0	1	1	1	1	0	0
0	1	1	0	␣	␣	1	1	0	0
0	1	1	0	1	0	1	␣	␣	0
␣	␣	1	0	1	0	1	0	1	0

Table III. A Solution with $m = 5$

0	0	0	0	0	1	1	1	1	1		
0	␣	␣	0	0	1	1	1	1	1	0	0
0	1	1	0	0	1	1	␣	␣	1	0	0
0	1	1	0	␣	␣	1	0	1	1	0	0
0	1	1	0	1	0	1	0	1	␣	␣	0
␣	␣	1	0	1	0	1	0	1	0	1	0

Table IV. A Solution with $m = 6$

0	0	0	0	0	0	1	1	1	1	1	1		
0	␣	␣	0	0	0	1	1	1	1	1	1	0	0
0	1	1	0	0	0	1	␣	␣	1	1	1	0	0
0	1	1	␣	␣	0	1	0	0	1	1	1	0	0
0	1	1	0	1	0	1	0	␣	␣	1	1	0	0
0	1	1	0	1	0	1	0	1	0	1	␣	␣	0
␣	␣	1	0	1	0	1	0	1	0	1	0	1	0

2 The Guideline

To find the solution for large m systematically, we begin with the definition of several terms.

Definition 1. *The relationship between adjacent two tiles is an i-relation if the tiles are identical. That two-tile pair is called an i-pair. The relationship between adjacent two tiles is a ni-relation if the tiles are not identical. That two-tile pair is called a ni-pair.*

Now, let α_i and β_i be the decrement of the number of i-relations and increment of the number of ni-relations in the ith operation, respectively. Clearly, α_i and β_i belong to $\{0, \pm 1, \pm 2\}$. The negative value of α_i refers to the increment of the number of i-relations and the negative value of β_i refers to the decrement of the number of ni-relations. For a general case with $2m$ elements, there are $(2m - 2)$ i-relations and 1 ni-relation at the beginning as shown in (1). There are 0 i-relations and $(2m - 1)$ ni-relations in the end as shown in (2). Thus, to accomplish the transition from (1) to (2) in n operations, we must have

$$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \beta_i = 2m - 2, \tag{3}$$

where $n \leq m$. However, α_i and β_i could never be negative by (3) and the pigeonhole principle. Moreover, we have

Theorem 1. *m is the minimum number of operations that could accomplish the transition from the state in (1) to the state in (2).*

Proof. By the rules of the puzzle, it is obvious that $\beta_1 \leq 1$ since the pair could only be placed at the end of the sequence in the first operation, which creates only one new relationship. Assuming the transition takes n operations, we have

$$2(n-1) \geq \sum_{i=2}^{n} \beta_i \geq 2m - 3,$$

since $\beta_i \leq 2$ and (3). Therefore,

$$n \geq m - \frac{1}{2},$$

which is equivalent to $n \geq m$ since both n and m are integers. □

Next, let us deduce a guideline to solve the puzzle with the minimum m operations. Since the essential goal of each operation is to replace the i-relation with the ni-relation, we have the notation of the score of ith operation as:

$$S_i \triangleq \alpha_i + \beta_i.$$

Because of (3), we have

$$\sum_{i=1}^{m} S_i = 4m - 4. \tag{4}$$

On the other hand $S_i \leq 4$ for all the operations since each operation can create at most two new relationships and remove at most two old relationships. Before going any further, we should give the definition of gaps.

Definition 2. *A gap is a two-tile vacant space left by taking a pair from the inner part of the sequence. If the boundary tiles of a gap are identical, that gap is called an i-gap. If the boundary tiles of a gap are not identical, that gap is called a ni-gap.*

For an operation with an optimal score of $S_i = 4$, we must either shift an i-pair to fill an i-gap and create an i-gap or shift a ni-pair to fill a ni-gap and create a ni-gap. However, the starting state (1) and the ending state (2) do not contain any gap. It is easy to verify that $S_i \leq 3$ for the operation creating the first gap and filling the last gap. Therefore, creating each gap will cost a score of at least 2. Moreover, it is easy to verify that we must shift an i-pair in the first operation and a ni-pair in the last operation. Hence, there is a discontinuity in which we start to shift a ni-pair instead of an i-pair. Such a discontinuity will cost a score of at least 2. Thus,

$$\sum_{i=1}^{m} S_i \leq 4m - 4. \tag{5}$$

Comparing (4) and (5), we draw the conclusion that there is at most one gap at any point of the puzzle. Additionally, we must shift an i-pair in the first operation and continue to do so. Starting from certain number of operations later, we shift the ni-pairs until solution.

To determine when we shift the first ni-pair, we should take the absolute position of each tile in the sequence into account. Since only pairs can be moved, the sequence in (1) will be shifted to the sequence in (2) by even tiles. Hence, the desired tile in each position of the sequence is fixed. If a certain tile is not the desired tile in that position, we call that tile wrongly positioned. It is obvious that the number of wrongly positioned tiles at the beginning is equal to the Hamming distance between the 2m-tuple in (1) and the 2m-tuple in (2), which equals to $2\lceil \frac{m}{2} \rceil$, where $\lceil x \rceil \triangleq \min \{n \in \mathbb{Z} : n \geq x\}$.

Clearly, operation on i-pair does not decrease the number of wrongly positioned tiles. The number of wrongly positioned tiles will be decreased by 2 only if we shift a ni-pair from

a wrong position to a correct position. Therefore, we must have at least $\lceil \frac{m}{2} \rceil$ such operations for a valid solution. Since shifting correctly positioned ni-pair makes the solution not minimum in m moves, we conclude that all the operations on the ni-pair should be on wrongly positioned ni-pair. Accordingly, we have exactly $\lceil \frac{m}{2} \rceil$ such operations.

In summary, the operations in solving the puzzle could be divided into two phases. The first phase takes $\lfloor \frac{m}{2} \rfloor$ operations, where $\lfloor x \rfloor \triangleq \max\{n \in \mathbb{Z} : n \leq x\}$. The second phase takes $\lceil \frac{m}{2} \rceil$ operations. In the first phase, we only shift i-pairs, and fill out the i-gap after the first operation for $m > 3$. We will alternatively shift i-pair with 0's and i-pair with 1's. The i-pair must be picked from the middle of 0000 or 1111 before the $\lfloor \frac{m}{2} \rfloor$th operation. If we cannot find such a pattern in the sequence or there is still such a pattern after $\lfloor \frac{m}{2} \rfloor$th operation, there will be no solution for that trial. In the second phase, we only shift the wrongly positioned ni-pair to fill out the ni-gap. The first tile of wrongly positioned ni-pair must alternatively be 0 and 1. The ni-pair should be picked from the middle of 0011 or 1100. If we cannot find such a pattern in the sequence before the last operation, there will be no solution for that trial. With such a guideline, we can directly find a solution for the case of $m = 7$ and the result is listed in Table V.

Table V. A Solution with $m = 7$

0	0	0	0	0	0	0	1	1	1	1	1	1	1		
0	␣	␣	0	0	0	0	1	1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1	1	1	␣	␣	1	1	0	0
0	1	1	0	␣	␣	0	1	1	1	0	0	1	1	0	0
0	1	1	0	1	0	0	1	1	␣	␣	0	1	1	0	0
0	1	1	0	1	0	␣	␣	1	0	1	0	1	1	0	0
0	1	1	0	1	0	1	0	1	0	1	0	1	␣	␣	0
␣	␣	1	0	1	0	1	0	1	0	1	0	1	0	1	0

3 The Solution

Although the guideline facilitates the procedure to find the solution, there are still ambiguities in both phases. Now, we provide the second main result in Theorem 2. The constructive proof of the theorem provides a recursive algorithm that guarantees a general simple solution.

Theorem 2. *There always exists a transition from the state in* (*1) to the state in* (*2) with the minimum m operations.*

Proof. We prove it by mathematical induction. First, we find the solutions for $m = 3, 4, 5, 6, 7$ in Tables I, II, III, IV, and V, respectively. For $m \geq 8$, let us consider the transition listed in Table VI.

Table VI. A General Solution

0	0	0	0	0	0	0	0	…	1	1	1	1	1	1	1	1	
0	␣	␣	0	0	0	0	0	…	1	1	1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	0	…	1	1	1	1	␣	␣	1	1	0	0
0	1	1	0	␣	␣	1	0	…	1	0	1	0	1	0	1	1	0	0
0	1	1	0	1	0	1	0	…	1	0	1	0	1	0	1	␣	␣	0
␣	␣	1	0	1	0	1	0	…	1	0	1	0	1	0	1	0	1	0

In Table VI, each of the first three ellipses represents a binary sequence of continued segments of (m – 8) 0's and (m – 8) 1's, while each of the last three ellipses represents a binary sequence of continued segments of (m – 8) 10's. The step between two horizontal lines is from the induction assumption on the (2m – 8)-element subsequence situated at the center of original sequence. Such a step could be accomplished in m – 4 operations. Therefore, the whole transition takes exactly m operations. □

With Theorems 1 and 2, we can solve the puzzle in the minimum m operations recursively. One of the solutions for m = 19 is listed in Table VII.

Table VII. A Solution with m = 19

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 ␣ ␣ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ␣ ␣ 1 1 0 0
0 1 1 0 0 ␣ ␣ 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 ␣ ␣ 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 ␣ ␣ 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 ␣ ␣ 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 ␣ ␣ 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 1 1 ␣ ␣ 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 ␣ ␣ 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 ␣ ␣ 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 ␣ ␣ 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 ␣ ␣ 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 0 1 1 0 ␣ ␣ 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ␣ ␣ 0 1 1 0 0 1 1 0 0
0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ␣ ␣ 0 1 1 0 0
0 1 1 0 ␣ ␣ 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0
0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ␣ ␣ 0
␣ ␣ 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

4 The Counting

Furthermore, we can count the number of different solutions for all the cases of $m \geq 3$. Number of solutions for case $m = 3$ is 2. For $m > 3$, suppose the set of all the valid states after shifting all the i-pairs is $\mathfrak{S}$. The counting process is divided into three steps. First, we count how many possible ways we could go from the starting state to a state in $\mathfrak{S}$ as $\mathfrak{m}_1$. Second, we count the size of $\mathfrak{S}$ as $\mathfrak{m}_2$. Third, we count how many possible ways we could go from a state in $\mathfrak{S}$ to the ending state as $\mathfrak{m}_3$. Then, the total number of solutions $\mathfrak{M} = \mathfrak{m}_1\,\mathfrak{m}_2\,\mathfrak{m}_3$.

With the deduced guideline, the transition in solving the puzzle could be divided into two phases. In the first phase, we shift i-pairs from one side of the sequence to the other side, and i-pairs with 0's and i-pairs with 1's must be shifted alternatively. From now on, the gap will refer to the gap created at the end of the first phase of transition. Since the last shifted i-pair must be in the position of the gap, except that i-pair, all the i-pairs with 0's or all the i-pairs with 1's could be interchangeable. Also, the number of all the i-pairs shifted is $\lfloor \frac{m}{2} \rfloor$. Therefore, the number of possible shifting order from the starting state to a state in $\mathfrak{S}$ is

$$\mathfrak{m}_1 = \begin{cases} (q-1)!q! & m = 4q \\ (q-1)!q! & m = 4q+1 \\ (q!)^2 & m = 4q+2 \\ (q!)^2 & m = 4q+3 \end{cases}. \tag{6}$$

To count the number of distinctive states in $\mathfrak{S}$, we first count the number of possible position for the gap as $\mathfrak{m}_{21}$. With the gap fixed, we then count valid arrangements of the tiles as $\mathfrak{m}_{22}$. Thus, $\mathfrak{m}_2 = \mathfrak{m}_{21}\,\mathfrak{m}_{22}$.

First, it is easy to conclude that the outer four tiles must be either 0110 and 1100 or 1100 and 1001 for the left end and right end of every state in $\mathfrak{S}$. Also, the gap could intersect at most 1 tile with those tiles. Since the last shifted ni-pair must be 01, the number of all the ni-pairs shifted is $\lceil \frac{m}{2} \rceil$, and ni-pairs starting with 0's and ni-pairs starting with 1's must be shifted alternatively, we can decide the first ni-pair shifted or the gap which must be the correct position for the first ni-pair. Therefore, for all the cases,

$$\mathfrak{m}_{21} = 4q - 2, \tag{7}$$

where $m = 4q + r$ with $r \in \{0, 1, 2, 3\}$.

With the pigeonhole principle, a side remark here is that the kind of the gap will affect the value of α_i and β_i for $m > 3$ as follows:

$$\alpha_i = \begin{cases} 1 & i = \lfloor \frac{m}{2} \rfloor + \delta, m \\ 2 & \text{otherwise} \end{cases}$$

and

$$\beta_i = \begin{cases} 1 & i = 1, \lfloor \frac{m}{2} \rfloor + \delta \\ 2 & \text{otherwise} \end{cases},$$

where $\delta \triangleq \begin{cases} 1 & \text{The gap is an I-gap.} \\ 0 & \text{otherwise} \end{cases}$.

Next, it is straightforward to conclude that the next outer four tiles must be either 0110 or 1100 and either 1100 or 1001 for the left end and right end of every state in $\mathfrak{S}$. The same restriction applies if we remove those eight tiles and collapse the gap, and so on. Since 0110 or 1001 renders a wrongly positioned 01 ni-pair while 1100 renders a wrongly positioned 10 ni-pair. The number of times 1100 appears in the sequence could be calculated by the number of wrongly positioned 10 ni-pair needed in the second phase of the transition. Therefore,

$$\mathfrak{m}_{22} = \begin{cases} \binom{2q-2}{q-1} & m = 4q \\ \binom{2q-2}{q-1} & m = 4q+1 \\ \binom{2q-1}{q-1} & m = 4q+2 \\ \binom{2q-1}{q} & m = 4q+3 \end{cases}, \tag{8}$$

where $\binom{0}{0} \triangleq \binom{1}{0} \triangleq 1$.

In the second phase, we shift wrongly positioned ni-pairs to the correct positions, and ni-pairs starting with 0's and ni-pairs starting with 1's must be shifted alternatively. The last shifted ni-pair must be 01 which must be at either end of the sequence. Thus, except that ni-pair, all the ni-pairs starting with 0's or all the ni-pairs starting with 1's could be interchangeable. Also, the number of all the ni-pairs shifted is $\lceil \frac{m}{2} \rceil$. Therefore, the number of possible shifting order from a state in $\mathfrak{S}$ to the ending state is

$$\mathfrak{m}_3 = \begin{cases} (q-1)!q! & m = 4q \\ (q!)^2 & m = 4q+1 \\ (q!)^2 & m = 4q+2 \\ q!(q+1)! & m = 4q+3 \end{cases}. \tag{9}$$

Consequently, by (6), (7), (8), and (9), the number of different solutions $\mathfrak{M}$ for case $m > 3$ is

$$\mathfrak{M} = \mathfrak{m}_1\mathfrak{m}_2\mathfrak{m}_3 = \mathfrak{m}_1\mathfrak{m}_{21}\mathfrak{m}_{22}\mathfrak{m}_3 = \begin{cases} (4q-2)(2q-2)!(q!)^2 & m = 4q \\ (4q-2)(2q-2)!(q!)^2\, q & m = 4q+1 \\ (4q-2)(2q-1)!(q!)^2\, q & m = 4q+2 \\ (4q-2)(2q-1)!(q!)^2\, q(q+1) & m = 4q+3 \end{cases}.$$

For $m = 19$, there are 812851200 solutions.

Endnotes

1. This can only work in the case of simple problems such as Tower of Hanoi, where the number of states is fairly small. In the case of moderately complex problems, such as the 15-puzzle, the number of states is about the same as the number of neurons in the brain, making it unlikely that a person can store all states in long-term memory. For more complex problems, such as chess, the number of states is larger than the number of atoms in the universe. Clearly, storing all states of a problem and exploring the entire problem space is unrealistic.

References

Beilock, S. L., & DeCaro, M. S. (2007). From poor performance to success under stress: Working memory, strategy selection, and mathematical problem solving under pressure. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 33(6),* 983–998.

Bühner, M., Kröner, S., & Ziegler, M. (2008). Working memory, visual-spatial-intelligence and their relationship to problem solving. *Intelligence, 36,* 672–680.

Chronicle, E. P., MacGregor, J. N., & Ormerod, T. C. (2004). What makes an insight problem? The role of heuristics, goal conception, and solution recoding in knowledge-lean problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 30(1),* 14–27.

Chu, Y. (2009). *Human insight problem solving: performance, processing, and phenomenology*. Berlin: VDM Verlag.

Chu, Y., Dewald, A. D., & Chronicle, E. P. (2007). Theory-driven hints in the cheap necklace problem: A preliminary investigation. *The Journal of Problem Solving, 1(2),* 18–32.

Fleck, J. I. (2008). Working memory demands in insight versus analytic problem solving. *European Journal of Cognitive Psychology, 20(1),* 139–176.

Kotovsky, K., Hayes, J. R., & Simon, H. A. (1985). Why are some problems hard? Evidence from Tower of Hanoi. *Cognitive Psychology, 17,* 248–294.

Kotovsky, K., & Simon, H. A. (1990). Why are some problems really hard: Explorations in the problem space of difficulty. *Cognitive Psychology, 22,* 143–183.

Larkin, J. H., & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science, 11*, 65–99.

MacGregor, J. N., Ormerod, T. C., & Chronicle, E. P. (2001). Information-processing and insight: A process model of performance on the nine-dot and related problems. *Journal of Experimental Psychology: Learning, Memory and Cognition*, *27*, 176–201.

Maier, N. R. F. (1930). Reasoning in humans: I. On direction. *Journal of Comparative Psychology, 10*, 115–143.

Newell, A., & Ernst, C. (1965). *The search for generality.* Proceedings of the IFIP Congress.

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice.

Öllinger, M., Jones, G., & Knöblich, G. (2006). Heuristics and representational change in two-move matchstick arithmetic tasks. *Advances in Cognitive Psychology, 2(4),* 239–253.

Özcan, E., Bilgin, B., & Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis, 12,* 3–23.

Patra, S., Goswami, S. K., & Goswami, B. (2009). Fuzzy based fast dynamic programming solution of unit commitment with ramp constraints. *Expert Systems, 26(4),* 307–319.

Pizlo, Z., & Li, Z. (2003). Pyramid algorithms as models of human cognition. *Proceedings of IS and T/SPIE Conference on Computational Imaging, 5016,* 1–12.

Pizlo, Z., & Li, Z. (2005). Solving combinatorial problems: 15-puzzle. *Memory and Cognition*, *33(6),* 1069–84.

Pizlo, Z., Stefanov, E., Saalweachter, J., Li, Z., Haxhimusa, Y., & Kropatsch, W. (2006). Traveling salesman problem: A foveating pyramid model. *Journal of Problem Solving, 1,* 83–101.

Pushkin, V. N., & Saltykov, G. F. (1972). Some mechanisms in solutions of problems involving movement of objects under restrictive conditions. In V. N. Pushkin (Ed.), *Problems of heuristics* (pp. 73–86). Jerusalem: Israel Program for Scientific Translations.

Renkl, A., Hilbert, T., & Schworm, S. (2008). Example-based learning in heuristic domains: A cognitive load theory account. *Educational Psychology Review, 21,* 67–78.

Tolman, E. C. (1932). *Purposive behavior in animals and men.* New York: Century.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science, 237*, 1317–1323.

Vrakas, D., & Vlahavas, I. (2005). Hybrid ace: Combining search directions for heuristic planning. *Computational Intelligence, 21(3),* 306–331.

Zhang, J., Johnson, T. R., & Wang, H. (1998). Isomorphic representations lead to the discovery of different forms of a common strategy with different degrees of generality. *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum.