Ricochet Robots - A Case Study for Human Complex Problem Solving

Nicolas Butko*, Katharina A. Lehmann†, Veronica Ramenzoni‡

September 15, 2005

1 Introduction

At the beginning of the Cognitive Revolution, stimulated by the development of computers, the Information Processing approach aimed to uncover the cognitive processes responsible for human problem solving capabilities. However, despite the initial interest and due to little success identifying the general processes that underlie human problem solving, the field scattered in different areas of study that employed different methodologies and responded to distinct goals. Nevertheless, in recent years and for reasons that seldom overlap, the processes involved in real-life complex problem solving rouse the interest of both the psychological and the artificial intelligence community. Despite the theoretical and methodological differences, the articulation of both approaches opens an interesting avenue of research for the study of processes involved in real-life complex problem solving. On the one hand, the computer simulation of real-life tasks offers an important tool for psychological study, which contributes to the development of novel techniques for measuring of human performance. On the other hand, a better understanding of how human subjects solve everyday-life problems provides new insight into how to design more effective computer solutions.

The aim of this project was to integrate both approaches and explore aspects in which they could be mutually informative. We consider a computer-generated game, "Ricochet Robots," as a starting point for the investigation of both human and computer-generated solutions. This game is interesting for several reasons. The first is that it lies on both sides of the boundary of tractability for computers. Some boards can be solved optimally by modern personal computers, others cannot. Because of the exponentially growing nature of the

*University of California, San Diego, Cognitive Science Department, Complex Systems and Cognition Laboratory, 9500 Gilman Drive La Jolla, CA, USA

†Universität Tübingen, Wilhelm-Schickard-Institute für Informatik, Sand 14, 72076 Tübingen, Germany

‡University of Cincinnati, Department of Psychology, PO BOX 210376, 429 Dyer Hall, Cincinnati, OH, USA

game, even with computer power doubling in accordance with Moore's law, this game will span the boundary of computability for many years to come. A second reason Ricochet Robots is an intriguing problem domain is that, while it is easy for human experts to find some solutions to almost every solvable board configuration, it is often quite difficult for them to find the optimal solution. This points to something interesting about human problem solving beyond just exploring a search space blindly. It may indicate that some avenues are easily searched, and others are heavily suppressed. In that respect, the analysis of human performance in solving the game provides interesting information about the different strategies used in solving complex problems.

A final point of interest about this particular task is that it follows notions of complexity both from a computer science point of view and from a psychological point of view. The former criterion is satisfied because of the exponential nature of the search space. The latter is satisfied because the structure underlying the game is changed every time a move is made: by moving one robot, the possibilities that are available to the other robots may change considerably. In order to explore these notions, we approach the study of the game using four distinct but complementary strategies. After explaining the game in Sec. 2, we discuss an optimal solution finder, and the difficulties faced in the creation thereof in Sec. 3. In Sec. 4 we study human learners, and try to glean insights about their strategies for solving novel problem solving tasks. This aspect of the project involved the experimental analysis of a pilot sample of human subjects and a detailed case study of a more complex version of the game. Third, we will describe in Sec. 3.2 our attempts to build a solution finder based on intuitions about the way 'expert human subjects solve the game: one that finds solutions quickly, though not necessarily the optimal ones. Finally, Sec.5 describes the changing underlying structure of the game's landscape to explore the self-organization of agents.

2 Ricochet Robots - The Game

Ricochet Robots has been created by Alex Randolp, and published by *Hans im Glück* in 1999 as *Rasende Roboter*. In the UK and US it is distributed by *Rio Grande Games*. The classic board game is played on a 16×16 grid of *cells* C. A cell can thus be represented by a pair of numbers denoting its x- and y-position, respectively (s. Fig. 1). A cell can have walls at the upper, right, lower, or left border. The grid is bound by a wall, such that all cells at x-position 0 have a wall on their left border, all cells at y-position 0 have a wall on their upper border, all cells at x-position 15 have a wall on their right border, and all cells at y-position 15 have a wall on their lower border. We will call walls on the upper border 'northern' walls, walls on right borders 'eastern' walls, and so on. The four cells in the middle of the board are not accessible and also bound by walls. Additionally, some of the inner cells of the board show walls.

Four so-called *robots* R_1, R_2, R_3, R_4 can be placed arbitrarily on the grid, as long as no two robots are standing on the same cell and none is placed on the

four cells in the middle. Every robot can be moved by any player. A *move* of any robot is defined as moving vertically or horizontically over the cells until an obstacle is met. An obstacle is either defined as a wall that is perpendicular to the direction the robot moves or any other robot standing on a cell in the direction the robot moves. For example, if robot 1 moves horizontically to the left, it can reach cell $(5, 6)$ because it is blocked by robot 4 on cell $(4, 6)$. Horizontically to the right it will stop at cell $(15, 6)$, stopped by the eastern wall on this cell. Vertically upwards it will stop at cell $(9, 4)$, stopped by the southern wall of cell $(9, 3)$. Vertically downwards it will stop at cell $(9, 15)$.

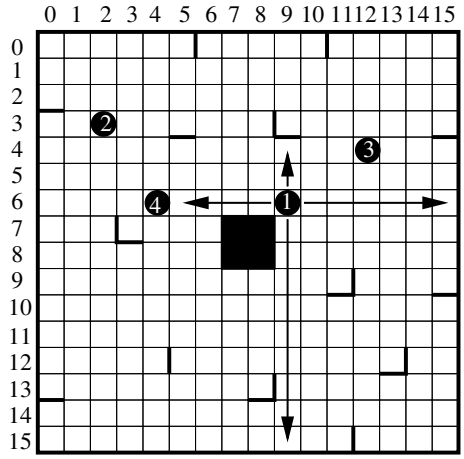

Figure 1: A typical board of *Ricochet Robot* is based on a 16×16 grid. Four robots, represented by labelled black dots, are placed on it arbitrarily such that no two robots occupy the same cell and none is placed on the four cells in the middle of the board. Cells can have walls. A robot has to move either vertically or horizontically until it is stopped by a wall or another robot, i.e., no robot can stop without bouncing into an obstacle. The four possible moves of robot 1 are indicated by arrows in the figure.

A robot can **only** stop at those cells where either a wall blocks it move or another robot.

The original game is played in 17 rounds: In each round a pair of robot R and cell C is chosen. The task is to find the minimal number of moves such that robot R will stop at cell C. All robots can be moved in order to help robot R to reach that cell and stop there. Two examples are given in Fig. 2.

For our experiments in Sec. 4 we used a self-implemented computer version of the game.

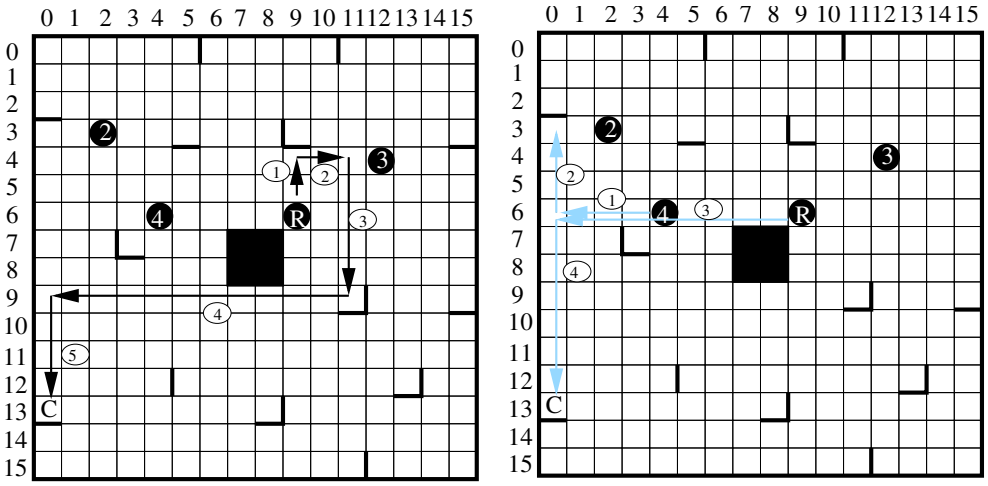

Figure 2: The task is to move robot R to cell C with as little moves as possible. On the left side it can be seen that the robot can reach cell $(0, 13)$ with five moves by first going upwards, than rightwards, downwards, leftwards, and downwards. Another solution is to move first robot 1 to the left and upwards (2 moves) plus two moves of robot R, yielding four moves in total.

3 Complexity of Optimal Solution Finder

We have indicated in the introduction that *Ricochet Robots* is a 'complex' game because it 'grows exponentially'. What does it mean to say that Ricochet Robots has an exponentially growing nature? At any board position in the game, a robot will have at most four moves available to it. Usually the robot will be standing against at least one obstacle, and so will have at most three moves available, but board situations can be created fairly easily where a given robot is not against a wall. In the example board in Figure 3, every piece has three moves available to it, and so there are 12 possible moves.

After we make each of those 12 moves, each robot will have up to three moves to make for 12 more possible moves. In order to see where we can arrive in two moves, we must consider 144 possibilities, consequently. For three moves, we must consider 12×144, or 1728 possibilities. To see where each robot can be in six moves, we must consider almost three million possibilities. In general, if we consider that each robot has three moves available to it at a given time, the number of possibile outcomes n moves in the future is 12^n.

We call 12 a "branching factor" because we can represent the possible sequences of moves as a tree, where each node has 12 "branches." This is also illustrated in Figure 3. In reality, the branching factor is slightly smaller, because often a piece will only have two, and sometimes only one or zero options available. As we will see, with smart algorithms and large memory usage, we

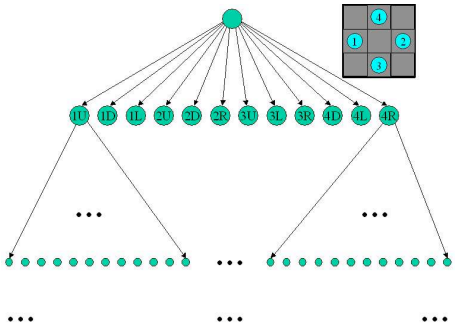

Figure 3: The board depicted in this picture allows three (independent) moves for every of the four robots, represented by 12 points in the search tree. Depending on the choice of each of these robots, the moves and or the number of possible moves of the others may be influenced. Also, the board shows that it makes a difference in which order the robots will move: If robot 3 moves to the right or left, robot 4 can now move on more fields downwards than before. If robot 4 had first moved downward, then robot 1 had moved to the right, it would take three moves in total to get robot 4 to the former field of robot 1. Thus, the search tree must hold all possible orders of all possible moves at any given time step.

can reduce the branching factor even further, but the fundamental problem will still remain exponential.

To completely understand search trees, we need to review some terminology. A "tree" is a connection of "nodes." In our case, each node corresponds to a situation on the board, i.e., the positions of all four robots. Each node (current board situation) is connected to another node (possible next board situation) by a "link" if there exists one move from one of the robots that will transform the first situation into the latter. The situation on the starting board is called the "root node." A "level" of the tree is the set of all nodes a fixed *distance* away from the root, where the *distance* is here defined as the number of moves needed to transform the situation on the starting board to the wanted situation. Thus, level n is the set of all board situations that can be reached in exactly n moves from the beginning. A "child" of a node is a node in level $n + 1$ that is linked to by a node in level n. A node represents a *final state* if the designated robot R is on the cell C of the board, i.e. if we have won.

Notice that since it will be possible to access the same board position at different times, nodes corresponding to identical board positions may exist at different levels in the tree.

3.1 Comparison of Search Algorithms and Time/Memory Tradeoffs

There are two basic classes of algorithms for searching through possibilities of moves such as we have in this case. These are *Breadth First Search* (BFS), and *Depth First Search* (DFS). We will explain these in more depth shortly. Over the years, there has been little improvement to these basic algorithms for searching entire trees. Even fancy search techniques like A* search provide no improvement in our case because each move (and so each link in the tree) is weighted equally.

3.1.1 Breadth First Search

In Breadth First Search (**BFS**), we start at the root node, and add all the children of the root node to a list of moves we are considering. We take each of these in turn, checking to see if it is a final state or not. If not, we take all of the children of that node and add them to the end of the list. After we've considered all nodes in level 1, our list will consist only of nodes from level 2. After we consider all nodes in level 2, our list will consist only of nodes from level 3, and so on.

Breadth First Search is time efficient, but costly in terms of memory. One reason that it is time efficient is that we will find an optimal solution before any suboptimal solution. By searching levels in order, we guarantee that the first winning board position we find will be the one with the least number of moves. The price we pay for this is that we need to keep up to B^n possible moves in memory at a given time, where B is the branching factor that we discussed above.

Luckily, we can reduce this branching factor quite a bit with an efficient implementation. For a relatively small amount of extra memory, we can maintain not only the current board positions, but all previous ones in memory. This prevents us from having to add board positions we have already seen to our search tree, and having search them again. The cost of this improvement is small. For a branching factor of 2, we need only twice as much memory; for higher branching factors, we need less than twice as much. The time needed to search through old moves to find duplicates is proportional to the level and not the total number of moves, and so it takes little time.

Thus, with a sophisticated algorithm, for a little bit of extra memory and a very small amount of extra time, we can reduce our branching factor and substantially increase performance. This will ultimately allow us to consider a few extra moves for the optimal board position.

3.1.2 Depth First Search

In contrast to BFS, Depth First Search (**DFS**) sacrifices time in favor of memory efficiency and facility of implementation. In Depth First Search, we start at the root node and search its first child's first child's ... until we come to a node with

no children. Then we go back to the parent of the later node and search its second child, and so on. Now, instead of always having to remember every node in a level, we only have to remember which node is the parent of the current node and which are its children. This takes virtually no memory.

Unfortunately, we don't get the same guarantees about finding optimal solutions first, or even at all. For example, if a sequence of moves just repeats board positions, DFS could search that sequence ad infinitum because it has no memory of what it has seen before.

Luckily, we can circumvent this problem with an algorithm called Depth Limited Depth First Search (DLDFS). DLDFS searches only nodes of level less than l, a depth limit. By increasing l to $l+1$ after all nodes of level up to level l have been searched, we can guarantee that the optimal solution will be found, and will be found first.

The cost of this guarantee is repetition: since we have no memory of level $l-1$, we have to go through the entire tree until we get to level $l-1$ before we can consider level l. So, to consider level l, we must consider level $l-1$ twice, level $l-2$ three times, and so on. As we saw before, with a branching factor of 12, level 6 contains three million elements, which we will have to re-search for each level $l > 6$. This seems like a large sacrifice, but it is always small compared to the size of the next level: with a branching factor of 12, if level 6 has three million elements, level 7 will contain 36 million. Compared to the cost of exponential growth, the cost of redundancy is small.

Still DLDFS has a drawback, which is that while we were able to use our memories in BFS to reduce our branching factor in some sophisticated way, we must continue to use a naive branching factor with DLDFS. This produces a very real tradeoff between a time-expensive algorithm and a memory-expensive algorithm. This tradeoff is illustrated in Figure [n+1].

3.2 Optimal Solutions

As argued above it is difficult to implement an optimal solution finder for Ricochet Robots. Nonetheless many humans love that game and do agree, normally after some minutes of intensive starring at the board, that they have indeed found an optimal solution. In the next section we will discuss a different way how a computer could be used to find solutions and then start our discussion on the human problem solver.

3.3 A Human inspired Problem Solver

Speaking with some experienced players we tried to derive their personal problem solving strategies. It seems that most players decide within a short time span to either

1. try to solve it 'forwards' by making some moves with the designated robot R. Most stop after 4-5 steps.

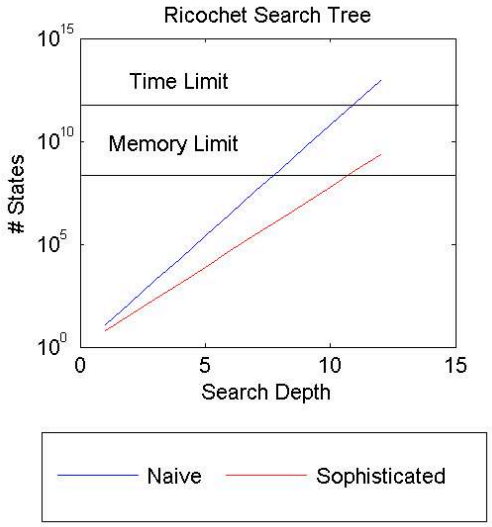

Figure 4: Time/memory tradeoffs for modern processors. Using a naive search algorithm, we have a high branching factor, and are limited by time rather than memory. Using a sophisticated search algorithm, we have a lower branching factor and are limited by memory, and not time. Reasonable limits for current personal computers are given.

2. try to solve it 'backwards' by finding out from where R has to come in order to reach the cell C.

It seems that this first decision is made very quickly. Some players report that they just 'knew' that the overall move sequence will be hard so it made more sense to try it backwards. Sometimes C is very near to the current position of R and then the forward strategy seems to be a better starting point.

If the 'backward' strategy is used, a player will look from where R has to come from to reach C. Sometimes it is clear that R cannot reach C with any 'helper robot' that is placed beneath the path to C (see Fig 5).

To find the solution, players often use forward and backward strategies in alternation. The knowledge when to abort the planning and start a new way throught the search space seems to corellate with experience of the player. This

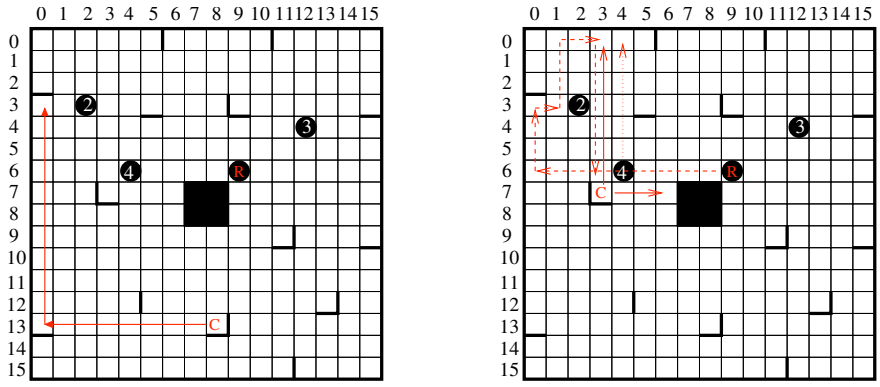

Figure 5: On the left, a backward analysis reveals that it would be good to place R somewhere between cell (0,3) and cell (0,13). Luckily, it is easy to move R there, once robot 4 has gotten out of its way. Thus, the following move is optimal: move robot 4 upwards, move R leftwards, downwards, rightwards. On the right, backward analysis reveals that R has to stop at either one of the cells between (3,0) and (3,6) to reach C or at one of the cells between (4,7) and (6,7). The fastes possibility to stop it is to move robot 4 upwards and to follow the dotted arrows.

'knowledge' is also acquired by playing the same board for 17 rounds in a row, so experienced players will memorize how to get any robot from the left upper corner to the middle of the right side, or will know that it is hard to find a way to all the cells in a special part of the board, and so on.

How can this be mimicked by a computer? We first implemented a module that mimics the 'forward' strategy: In this part of the program the computer will find all cells that can be directly reached by R. In a further step the computer evaluates all cells that can be reached by R if the other robots were not blocking him, i.e. for all possible combinations, the other robots were temporarily removed from the board and again the computer evaluated which cells were now reachable from R.

Comparing this solutions with human players it is quite obvious that human players would not regard a direct solution in the first step if it requires more than 10 steps. Thus, humans seem to make no more than 10 forward steps in a first try. To mimic different stages of expertise it would be interesting to parameterize the depth of this strategy and compare solutions with solutions found by human players.

The backward strategy is still hard to implement because also here we have a high complexity. The basic idea is to find all cells from which C can be reached directly. It is then clear that R has to stop on one of these fields to get to C. If there is a 'natural' obstacle, i.e., a wall that could stop it on one of these cells, the next backward analysis step would try to find all cells such that this

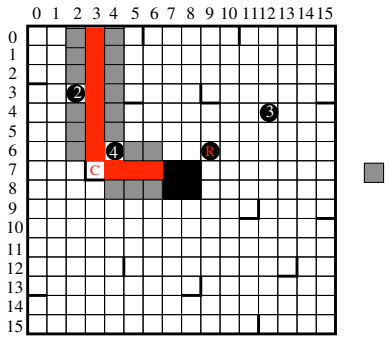

Figure 6: The darker fields show where R has to stop to reach C directly. The grey cells indicate where a helper robot has to be positioned in order to let R stop on one of the dark fields.

obstacle will be touched. Thus, the procedure to find all cells from which C can be directly reached has to be called recursively for all obstacles that can stop R on those cells already discovered.

If R can stop on any of the discovered cells, it can reach the goal C directly, thus, we would have found this solution with the forward analysis. If there is no more 'natural' obstacle on its way, we have to try, to move one of the three other robots to a cell neighbored to the cells we want R to stop. Looking at the examples from Fig. 5 it becomes obvious that this it is a major decision with which of the neighbored cells to begin.

In Fig. 6 we have coloured in grey all those cells we would have to examine further. The question for all of these cells is: Can we find a robot that can stop on one of these grey cells **and** will this help R to stop on one of the dark cells? As we now, robot 4 can be put on one cell that is reachable by R. So, we have to evaluate all the grey cells with respect to the improvement they *could* bring before really testing them, because otherwise we would have a very similar devastating search tree as for the optimal solution finder. It is certainly possible to build such an evaluating function, but it requires a very efficient implementation, anyhow. It would be interesting to actually build this backward analysis part of the solution finder. Then, it could be parameterized similarly to the suggestions made for the forward analysis. Computer generated solutions could then be compared to solutions found by human players on different expert levels and maybe parameters for beginner and experts could be abstracted.

The next section will give some results on how human players solve *Ricochet Robots* on our computer version of the game.

4 Human Complex Problem Solving

Scientific approaches to the study of psychological phenomena have traditionally attempted to mimic the physics approach to the study of complexity, by studying simple systems and then extrapolating the results to complex systems. This is particularly true for the study of human problem solving. Early efforts in problem solving research focused on the study of simple novel tasks. The primary motivation behind this line of research was that simple tasks have clearly defined optimal solutions which allow for subjects' problem solving steps to be easily traced. This approach assumed that the cognitive processes underlying the subjects' performance on simple tasks were representative of the processes involved in 'real-life problem solving tasks. However, empirical findings on the non-generalizability of theoretical models to different knowledge domains discredited this hypothesis [10]. Research on problem solving in the following decades took two distinct approaches. The information processing perspective in North America centered its efforts on the study of problem solving in different knowledge domains (e.g. chess playing, calculation, managerial and political problem-solving, computer skills and game playing), while the Complex Problem Solving (**CPS**) perspective in Europe focused on the study of human performance in real-life tasks.

As mentioned in the introduction, information processing research separated into two main lines of research among others, with little overlap on the theoretical models used in both areas: the development of expertise in specific problem solving domains and the development of computer solutions to domain specific problems. Following the tradition in the field, both approaches oriented their efforts to the analytic study and description of domain-specific close-system problems (e.g. Tower of Hanoi), which in most cases did not reflect real-life problems. The complexity if a problem in this context was in general equated to the complexity of the problem's search space.

Artificial intelligence attempts, although successful in some cases, failed to provide important insights on how humans solve similar problems. For instance, human chess experts could consistently outplay computers that relied heavily on brute-force searching techniques. The realization that humans could even function at all in tasks for which the search spaces were astronomical led researchers like Alan Newell to study heuristic search in his 1972 paper outlining his General Problem Solving System (**GPS**) [2]. One of the "features" of the GPS system was that, as with people, if it did not find a correct solution to the problem, it would just "give up" rather than continue the Sisyphean task of searching an impossibly large search space. The realization that human problem solving does not entertain entire search spaces was important among other reasons because it demonstrated the lack of face validity of the deterministic models used for the characterization of problems. Critically, the tasks used were not representative of real-life problems and provided limited information on the strategies and models used by human subjects in solving them. Furthermore, this research also demonstrated the limitations of the computational notion of complexity for the characterization of psychological phenomena [6].

The complexity posed by a problem is not necessarily equivalent to the complexity of it's search space, but to other properties of the problem as well as to the subject's mental representation of it.

The CPS perspective offers an alternative -more ecologically valid- conceptualization of 'complexity' [5, 4, 3]. CPS studies human problem solving from a holistic point of view; it employs computerized laboratory tasks constructed to mimic 'real-life problems and focuses on the motivational, as well as social variables that affect subjects' performance. In spite of the fact that the literature in CPS does not provide a unified definition of 'complexity[9], a task must assume certain characteristics in order to be considered a complex problem. The tasks used are novel, relatively complex, semantically rich, and dynamic (e.g. forest fires and ecologies). A task is considered dynamic if early actions taken in solving the problem determine the environment of the task in which further decisions must be taken, and in some cases features of the environment might change independently of the subjects' actions. Complex tasks are also time-dependent in the sense that subjects are constrained to make decisions by time limits, and some decisions must be made at particular moments in time depending on environmental demands. Finally, a task is considered complex when the variables that determine task environment do not relate to one another in a one-to-one fashion.

The 'complexity' stressed by this perspective emphasizes not only the computational properties of a problem, but also the psychological demands placed on a human solver [9]. The psychological complexity does not necessarily equate its perceived difficulty; modifications to the semantic content of a problem often yield significant changes in its perceived difficulty, even though the analytic structure of the problem remains constant. Due to this reason measures of complexity must rely on other factors besides self-report. Relational complexity refers to the number of units of information that must be considered in parallel in order to arrive at a successful solution. This quantity is limited by restrictions in working memory capacity. The meaningful units of information to be considered are in general identified with the arguments presented by the problem. Those in turn, refer to inferred relationships between variables in the game. Complex tasks require to establish novel relationships between variables, which in general results in high relational complexity. Despite its importance, the notion of relational complexity has only been empirically studied for relatively simple problems [7].

In the context of this project, we studied the psychological as well as the computational complexity of the "Ricochet Robots" game. We did not only consider the computational complexity of each puzzle, but we also analyzed the game according to a CPS model. *Ricochet Robot* represents an instance of complex problem solving in several respects. Arriving to a solution requires a long series of decisions, in which early moves determine and limit the following moves. The game can be set to restrict time-constraints depending on implementation goals. In the implementation of the game used in this project each puzzle was time constrained for the experimental study. Time constraints were removed for the case study in order to explore other characteristics of the game. *Ricochet Robots*

can also be considered a complex task computationally, as has been reviewed in Sec. 3, and psychologically. Although the psychological complexity of the game cannot be measured in an straightforward fashion, the number of arguments necessary to obtain an optimal solution can be experimentally manipulated. As a result of an increase in relational changes in participants' performance and/or in the strategies used to approach the task are expected. We investigated this possibility empirically by studying human subjects performance using two distinct experimental techniques. The objectives, implementation and results of them are described in detail in the following sub-sections.

4.1 Group Experiment

In order to explore how human subjects solve *Ricochet Robots* we conducted a pilot experiment with a small sample of subjects. The aims of the experiment were to test the importance of two factors which were hypothesized to play a role in solving the game. First, we inferred that subjects considered in order to find a solution the number of possible cells all robots could reach independently at all moments in the game. Based on this inference, we hypothesized that providing this information visually would result in an improvement in subjects performance. For each puzzle used in the experiment participants were presented with two versions, a version with no aids, and a version in which in which additional information was visualized upon clicking on a robot. This action changes the color of all those cells that the robot can move to at that specific moment in the game.

Second, we predicted that the complexity of the space of the problem would affect subject's performance. We expected that participants would have more difficulty solving puzzles in which the number of moves of the optimal solution was larger. As a result, we expected the initial move time and the overall time to increase. Furthermore, we predicted that participants would produce more recurrent moves (moving a robot and then returning it to it's starting cell) for this puzzles.

Finally, we considered that the functional role played by moving other robots - besides the target robot- in finding an optimal solution could provide some insight into the difficulty posed by each puzzle. We use the relationship between the robots in each board as a measure of the relational complexity of each puzzle. Four different puzzles were designed in order to represent four relationships between the robots in the puzzle. We predicted that performance would differed depending on the relational complexity of the puzzle. The puzzles require to achieve the optimal solution that the participant considers one (puzzle 1) or 2 arguments (puzzles 2 through 4). In detail, the arguments for each puzzle were: 1) consider only the designated robot, 2) consider the relation between the target robot and other robot that behaves as a obstacle, 3) and 4) consider the relation between the target robot and other robot that behaves as a helper.

4.1.1 Methods

Participants

Five students (three males and 2 females) from the Santa Fe Institute Complex Systems Summer School agreed to participate in this study. Participation was voluntary; participants received no monetary compensation for their participation. All participants were treated in an ethical manner (American Psychological Association, 1992).

Materials and Apparatus

The different board games were programmed in Java, using only standard Java libraries. Each Java GUI displayed a board game (X x X cm) at the center of the screen; to the right of the board four buttons displayed in vertical order: number of moves entered, undo option, reset option and the overall time elapsed since the beginning of the puzzle. Two personal computers were used for experimental material presentation (PowerPC G4, 12.1-inch TFT Display with1024x768 resolution and an IBM T30, 14-inch TFT Display with 1024x768 resolution). In both cases participants used a mouse pad to navigate the game's environment.

Procedure

Prior to the beginning of the experimental session participants received a tutorial. The first part of the tutorial consisted of a handout with instructions detailing the characteristics of the game, which included one example of each, a puzzle board with and without visual aids. Participants were instructed to carefully read the handout and to ask any questions referring to the game. The second part of the tutorial consisted on a training session during which an experimenter walked each participant through a puzzle while restating the rules of the game. Later the experimenter observed as the participant solved the same puzzle. Participants were allowed to explore the board -turning the visual aids on and off- until they declared themselves to be confident to begin with the experimental session.

Each experimental session consisted of 4 different puzzles, which were presented twice (with and without visual aids). The order of visualization conditions was counterbalanced across participants. The four puzzles differed on their degree of difficulty. The criteria for difficulty were: a) the minimum number of moves necessary to reach the goal (3, 5 or 6), and b) the number of relational arguments to be considered in finding an optimal solution. Participants were instructed to try to solve each puzzle in the minimum number of moves and in the least amount of time. A maximum time of 5 min was allowed for each of the four puzzles. If the participant reached a solution before the maximum time, he/she was instructed to restart the game and look for a better solution. Similarly, participants were allowed to reset the game to the initial board, if they could not find a solution. In addition, participants were allowed to undo their recent moves. After 5 min the screen froze and participants moved to the following puzzle. Each participant's moves and the time elapsed between moves, as well as total time, were recorded for later analysis. At the end of each experimental session participants answered a set of open questions about the

strategies used in solving the game and the relevance of visual aids in reaching the best solution.

4.1.2 Results and Discussion

Participants' cognitive performance on each board and the efficacy of visual aids were evaluated using three results-oriented indicators: *overall solution time*, *number of solutions per puzzle* and *solution quality*. Solution quality was operationalized as the total number of moves per solution. Good performance was determined according to the number of moves between the starting state and the goal state for each puzzle. On average participants showed larger mean times for the first move on each puzzle compared to the following moves. This effect is more pronounced for the first puzzle and decreases over time, which suggests a learning effect. Contrary to experimental predictions, results showed that the introduction of visual aids impaired participants' performance. T-test comparisons showed that on average significantly more moves and larger overall times were employed for boards that displayed visual aids compared to the same puzzle without the aids ($t(4) = 3.80$, $p < 0.05$ and $t(4) = 2.41$, $p < 0.05$, respectively). Furthermore, results showed significantly larger mean times for the initial move for the first puzzle with aids compared to the same puzzle without the aids ($t(4) = 2.68$, $p < 0.05$). ANOVA (**An**alysis **o**f **Va**riance) analysis comparing overall time for the different puzzles with and without visual aids showed increased overall times for puzzles with visual aids, but no significant differences between puzzles ($F(3, 1) = 9.94$, $p < 0.05$ and $F(3, 1) = 0.419$, $p > 0.05$, respectively). This result indicates that the effect of the visual aids on performance remained stable across the experiment.

Analyses performed on the ratio of the overall time employed in solving a puzzle divided by the puzzle's difficulty indicated that participants' performance was not significantly affected by the complexity of the space of the problem. However, the comparison of participants' performance on the different puzzles according to their relational complexity shows a preference for puzzles that do not require the help of additional robots to reach a solution. Qualitative analysis of the solutions produced in solving puzzle number two indicates a preference for solutions that do not require using other robots as helpers, even though moving other robots would lead to a better solution. Difficulties in solving puzzle number four provide additional support to this observation. Four of the participants attempted to solve the puzzle only by moving the target robot and reseting the puzzle upon realizing that a solution could only be reached by moving an additional robot; the fifth participant failed to find a solution to the puzzle.

Finally, due to the characteristics of the games implementation (puzzle difficulty), no process-oriented indicators were derived from the experiment. However, some intuitions about the nature of the cognitive strategies employed were obtained from the qualitative analysis of the sequence and timing of moves. In order to qualify the strategies used by the participants in finding a solution, the first solution obtained for each puzzle was analyzed separately in a sub-

sample. Temporal transitions between moves were analyzed independently for this sub-sample of the data. Within-subjects ANOVA was performed on the data obtained from this sub-sample, using the time for the initial, second and third move as main factors. Post-doc analyses indicated that participants took significantly longer times for the initial move compared to both the second and third moves ($F(3,1) = 7.35$, $p < 0.05$ and $F(3,1) = 9.40$, $p < 0.05$, respectively). These results suggest a tendency to use more of a planning strategy than a search strategy. Furthermore, participants' reports confirm this intuition; three of the participants reported the use of an inverse strategy, working backwards from the goal state to the initial state, before implementing a solution. However, after finding a satisfying solution to the board, at least two of the participants switched from a planning to a search strategy.

Overall, results suggest that subjects have a tendency to use a planning strategy to solve *Ricochet Robots*. Problem solving performance in this environment seems to be largely based on cognitive abilities to operate on a mental representation of the game. The inefficacy of the visual aids in improving subjects performance also indicates that the capacity to plan a successful strategy might be negatively influenced by the visual information presented. However, the role of visual aids cannot be qualified based on this evidence. Participants reported that the visual aids were confusing; they did not provide useful information and were obstacles to finding a solution. Finally, it can be argued that participants had more difficulty solving puzzles that require more than one argument. This indicates that the relational complexity more so than the computational complexity (space of the problem) of the task affects human capacity to solve the game.

4.2 Case Study

Case studies are typically used in computer modeling and simulation research more so than in psychological studies in the problem solving literature. However, some authors have argued in favor of their use, because they provide rich information about the cognitive activity and problem solving behavior of the individuals than do average data [8, 1]. In this case, we decided to further explore subject's behavior in solving the game by conducting a case study of one subject's performance.

We used for this study the puzzle for which subjects in the group experiment had significantly more difficulty finding an optimal solution. The experimental conditions differed from those of the group experiment. A single participant who had never played the game, was given brief tutorial on the game's rules and no written instructions or training examples. The participant only solved one puzzle and was allowed unlimited amount of time to complete it. We expected that the lack of time-constrains combined with the brief training would provide a finer grain picture of the learning process that underlines strategies development.

Qualitative analysis of the participant's performance suggests that rather than utilizing either planning or search strategies, the subject seemed to operate on a continuum between the two. The original data for the time between moves

is shown in Figure 7 **a**. Most often, the subject waited only a short time before making the next move, indicating an active search strategy. However, on several occasions the participant produced long pauses before deciding on a move, which suggests a planning strategy (Figure 7 **b**).

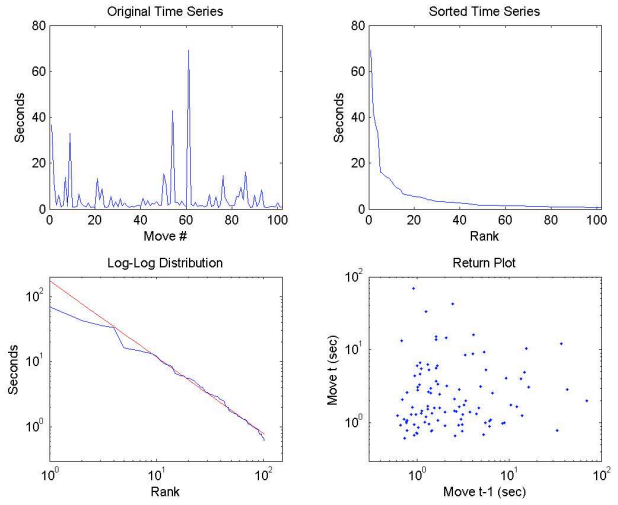

Figure 7: Case Study single-subject data. A) The original time series showing the delay after each move. B) The sorted time series, showing the proportion of long and short waiting times. C) The distribution in B seems to follow a power law, which may be indicative of Self-Organized Criticality phenomena. D) Each waiting time is plotted against the next waiting time. The lack of correlation shows that it is difficult to predict the next waiting time from the current one.

The subject did not, however, simply go back and forth between long waiting

times and short waiting times, and the planning regions are not readily distinguishable from the active searching regions. As Figure 7 **d** shows, the current waiting time is little indication of the next waiting time, and so the regions of planning strategies and the regions of search strategies are not well defined, but seem to lie on a continuum.

It would seem then that the language of problem-solving literature is inadequate to capture the dynamics of actual human problem solving. What seems to be happening is not that people use one of two strategies, but that they follow a complex decision-action cycle. The length of decisions seems to follow a power law distribution (Figure 7 **c**), which has been associated with Self-Organized Criticality phenomena.

Four phenomena have been identified as being essential for self-organized criticality:

- Constant Drive
- Storage
- Threshold Non-Linearity
- Energy Release.

One possible analogy between SOC and Problem solving would be to associate Constant Drive with the desire to reach the problem goal, Storage with the memory of the plan, Threshold with the confidence in a solution, and Release with action and forgetting. That is, when a subject is trying to reach a goal, they do some planning until they are reasonably confident in a direction of action, at which point, they act in accordance with their plan until their memory of the plan degrades, they reach the end of their planned trajectory with or without reaching the goal, or decide that their plan is not working. At this time, unless they have reached the goal (while the drive is present), they begin to plan again.

This is one of many possible analogies. While it seems pleasing, plausible, and to fit the SOC framework well, future tests will need to be performed to assess its veracity. Nevertheless, our early results indicate that human problem solving is indeed much more complex than had been previously asserted, and that the language of planning versus search strategies is insufficient to capture these complexities.

5 Self-Organization

In this section, we briefly explore how independently acting agents can cooperate to solve this problem. The agents were allowed to know everything about the world that was externally observable, but nothing about the plans of the other agents.

We began with a simple rule for the agent, and a simple rule for the helper robots.

- If you're an agent, move along the shortest path to the goal. If no such path exists, remain still.
- If you're a helper, find the point on the board where the sum of your moves to get there and the agent's subsequent distance to the goal is minimized, and move along the shortest path to that point. If you cannot reach a point which will allow you to help the agent reach the goal, remain still.

These two simple rules allowed the agents to solve many puzzles, though not all of them. Also, a few interesting interactions emerged. In one puzzle, the agent moved one step along a long path to the goal, and then another agent moved out of its way along a shorter path. The agent then retraced its steps and took the shortest path to the goal. In another puzzle, two helpers competed with each other to be the first to help the agent. This created confusion among them, as seemingly good moves became bad with the move of the other. Ultimately one helper got into a position where it could aid the agent and the second helper moved out of the way, allowing the agent to reach the goal. In at least one case the optimal solution was found.

A limitation of this scheme is that it only can find solutions in which one robot is sufficient to help the agent reach the goal. To get helper robots to coordinate would require one of two things: Communication among the robots, or knowledge of the other robots' strategies. In the latter case, the helper robots knew how other robots would act, and so they could each compute an optimal solution. This is not feasible because of the limitations of optimal solutions finders, discussed earlier. Even if they could act in a heuristic way, this would be equivalent to the heuristic solution finder. The interesting self-organization case is the former, in which helpers can communicate.

One example of a scheme that would allow two helpers to work together could be as follows: If you cannot help the agent, and no other helper has asked you to stop, move randomly. If you can help the agent, request that all other helper robots stop and help the agent as before.

When only one helper robot is needed, the probability of one robot wandering into a helping spot randomly is fairly high. When two helper robots are needed, the probability of them wandering into their respective helping spots simultaneously is low. A more sophisticated self-organization and communication scheme would be necessary for agents to solve the most difficult puzzles, in which all three robots are needed simultaneously.

6 Summary and Discussion

The project showed that certain games are a good starting point for exploring the complexity of human and computer problem solving. Although the project is still at its beginning we hope to do some more experiments and finally implement the digital solution finder to compare its results with solutions found by humans, to learn more about computers and brains and how they can learn from each other.

References

[1] D. Dörner and A.J. Wearing. *Complex Problem Solving: The European Perspective*, chapter Complex Problem Solving: Toward a (Computersimulated) Theory, pages 65–99. Hillsdale, NJ, Lawrence Erlbaum, 1995.

[2] G. Ernst and A. Newell. *GPS: A Case Study in Generality and Problem Solving*. Academic Press, 2969.

[3] P. Frensch and J. Funke. *Complex Problem Solving: The European Perspective*, chapter Definitions, Traditions, and a General Framework for Understanding Copmlex Problem Solving. Hillsdale, NJ: Lawrence Erlbaum, 1995.

[4] J. Funke. Computer-based testing and training with scenarios from complex problem-solving research: Advantages and disadvantages. *International Journal of Selection and Assessment*, 6(2):90–96, 1998.

[5] J. Funke. Dynamic systems as tools for analyzing human judgement. *Thinking and Reasoning*, 7(1):69–80, 2001.

[6] W. D. Gray. Simulated task environments: The role of high fidelity simulators, scaled worlds, synthetic environments, and laboratory tasks in basic and applied cognitive research. *Cognitive Science Quarterly*, 2(2), 2002.

[7] G.S. Halford, W.H. Wilson, and S. Phillips. Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Behavioral & Brain Sciences*, 21(6):803–864, 1998.

[8] R. H. Kluwe. *Complex Problem Solving: The European Perspective*, chapter Single Case Studies and Models of Complex Problem Solving, pages 269–291. Hillsdale, NJ: Lawrence Erlbaum, 1995.

[9] J. Quesada, W. Kintsch, and E. Gomez. Complex problem solving: A field in search for a definition? *Theoretical Issues in Ergonomic Science*, 6(1), 2005.

[10] J. R. Sternberg and P. Frensch, editors. *Complex Problem Solving*. Hillsdale, NJ: Lawrence Erlbaum, 1991.