



UNIVERSITEIT UTRECHT

Automated Rating of Level Difficulty for Puzzle Games

MASTER THESIS GAME AND MEDIA TECHNOLOGY

Author:

PAUL MUTSER

Thesis Number:

ICA-3484785

Email:

PAULMUTSER@GMAIL.COM

Supervisors:

PROF. DR. M.J. VAN KREVELD

DR. M. LÖFFLER

October 2014

Abstract

In this thesis, a method for automatically rating the difficulty of puzzle game levels is introduced. Our method takes multiple aspects of the levels of these games, such as level size, and combines these into a difficulty equation. The method can simply be adapted to most puzzle games. We test this method on three different puzzle games: Flow, Lazors and Move.

We conducted a user study to find out how people rate the difficulty of a set of levels. We then use this data to train the difficulty equation to match the user-provided ratings. Our experiments show that the difficulty equation is capable of rating levels with an average error of approximately one point in Lazors and Move, and less than half a point in Flow.¹

¹Levels are rated on a 1 to 10 scale.

Contents

1	Introduction	1
1.1	Problem Description	1
1.2	Goals	1
1.3	Thesis Overview	2
2	Related Work	3
3	Difficulty Equation	5
3.1	Models	5
3.2	Choosing Variables	5
3.3	Determining the Weights	6
4	Games for our Experiments	8
4.1	Games	8
4.1.1	Flow	8
4.1.2	Move	9
4.1.3	Lazors	9
4.2	Implementation	10
4.3	Levels	11
4.3.1	Level Generation in Move	11
4.3.2	Level Selection in Move	13
4.3.3	Level Solving in Move	13
4.4	Difficulty Measures	14
4.4.1	Flow	15
4.4.2	Lazors	15
4.4.3	Move	16
5	User Study	18
5.1	User Data	18
5.2	Study Setup	19
5.3	Level Rating	20
6	Results	21
6.1	Flow	21
6.2	Lazors	23
6.3	Move	26
6.4	Discussion	28
7	Conclusion and Future Work	30
7.1	Conclusion	30
7.2	Future Work	30
	References	32
	List of Figures	33

List of Tables	34
A List of All Levels	35
A.1 Flow	35
A.2 Lazors	37
A.2.1 Initial Configuration	37
A.2.2 Solved	40
A.3 Move	43
B Level Properties	47
B.1 Flow	47
B.2 Lazors	48
B.3 Move	50
C User Study Data	52
C.1 Flow	52
C.2 Lazors	53
C.3 Move	55
D Difficulty Equation Weights	57
D.1 Flow	57
D.2 Lazors	58
D.2.1 Original	58
D.2.2 Adjusted	59
D.3 Move	60
D.3.1 Original	60
D.3.2 Adjusted	61

1 Introduction

1.1 Problem Description

With the rise of smartphones over the past few years, the market for apps is growing rapidly. Especially small *pick up and play* games have seen a large growth in popularity. This group of games includes a large group of puzzle games. These puzzle games require a large number of levels that challenge the players, so that they will continue playing.

The creation of puzzles is a time consuming process that is usually performed by hand. A large part of this process consists of determining how challenging a puzzle level is. To keep players engaged with the game, it is desirable to present them with levels that fit their skill level. It is therefore important to know the difficulty level of each puzzle. A common method for determining this difficulty level is by large amounts of playtesting. By having a group of players play each level, it can be determined how hard a level is. This is a long and expensive process. By automating the evaluation of level difficulty, this process can be made cheaper and more efficient.

This automation is not trivial, as it needs to be determined what factors play a role in the difficulty of a level. Once these factors have been found, they need to be combined into a *difficulty equation* that has a level as its input and a difficulty rating as its output. The difficulty equation will allow level designers to determine the difficulty rating of new levels without going through a playtesting phase.

Difficulty is of course subjective, meaning that there is not one true difficulty rating for each level. The goal of the difficulty equation is to be able to rate levels as close to the average rating given by humans as possible. This means that it has to be trained to match these human ratings. Some data on the difficulty of levels in a game has to be acquired to be able to create an effective difficulty equation. This can be done using playtesting, but has the advantage that it only needs to be done once on a small set of levels. This data can then be used to create new levels which would not require playtesting.

A side effect of using a difficulty equation is that the formula itself can be used to determine what makes a level difficult. This data can then be used to create levels for specific difficulty levels. An added advantage is that this makes automatic level generation for a given difficulty level possible. This means that game designers no longer need to worry about level design, and can create a game where levels are generated in real time, at the skill level of the player. This keeps players more engaged, as they are not getting bored by levels that are too simple, or frustrated by levels that are too hard. They also will not run out of levels after playing the game for a longer time.

1.2 Goals

Our goal is to create an effective difficulty equation. This equation should be able to accurately assign difficulty ratings to puzzle levels. We aim to create a generic method that can be applied to most puzzle games. To be able to test how our method applies to different games, we perform our

experiments on three different games. These games are:

- *Flow*, a game where players must connect pairs of dots with non-intersecting paths.
- *Lazors*, a game where players move mirrors to direct laserbeams to certain targets.
- *Move*, a game where players move a set of balls to their matching goals, but can only move all balls at once.

As these are three very different games, we assume the difficulty equation will work well on most puzzle games, if it works well on these three games.

1.3 Thesis Overview

The rest of this thesis is structured as follows. Section 2 gives a short overview of previous work in the area of automated difficulty rating. Our model for the difficulty equation is presented in Section 3. Section 4 introduces the games we use for this research. An overview of the implementation, and how we apply the difficulty equation to these games is also given in this section.

We perform a user study to train the difficulty equation on actual data, and to be able to compare the results to real data. Details on this user study are presented in Section 5. The results of our research are shown in Section 6, and this section also includes an evaluation of these results. We conclude in Section 7, and show possibilities for future work.

2 Related Work

Some previous work has been done in the area of automated difficulty assessment, although it is not a lot. This section provides an overview of the related work that we were able to find.

Jarušek and Pelánek [1] attempt to calculate the difficulty level of Sokoban puzzles. User study data is used to find the difficulty, measured as the average time taken to solve a level, of a number of levels. They use various metrics based on the shortest solution to find the correlation between those metrics and the difficulty of a level. They also use more abstract measures that decompose the problem of solving a puzzle into smaller subproblems.

Ashlock and Schonfeld [2] assess the difficulty of Sokoban puzzles. They do this by attempting to solve levels automatically. The average time taken before a solution is found, and the probability that an attempt to find a solution fails, are used as difficulty measures. These measures are subsequently used to order the levels by difficulty.

Browne [3] uses linear equations and features of board games to generate enjoyable games. His goals are not directly related to our work, but his method is also applicable to determining difficulty. Using a user study, Browne finds the aspects of board games that makes those games enjoyable. This data is then used to create a linear equation, which determines the enjoyability of other board games. An evolutionary algorithm is then used to create a very enjoyable game, Yavalath. The same process could be used to rate the difficulty of levels, and create levels at a certain difficulty.

We use a similar method to Browne in our research by also combining several aspects of a game into a linear equation. Our equation is also trained to real data that we obtain from a user study. We use this equation to define the difficulty of levels, instead of the enjoyability.

Mantere and Koljonen [4] use genetic algorithms to solve, generate and rate Sudoku puzzles. They make use of their genetic Sudoku solver to rate the difficulty of the puzzles. Their assumption is that puzzles that take longer to solve by the algorithm are also perceived as more difficult by humans. This assumption is tested by comparing their solving times to the provided difficulty ratings of the puzzles. They conclude that their results support this assumption.

Aponte, Levieux and Natkin [5] split games into a series of challenges. They calculate the probability of succeeding in each challenge, and consider this probability as the difficulty of that specific challenge. The probabilities are calculated using the available information on the players history, such as if the player has seen a similar challenge before. By combining the difficulties of the challenges, they are able to calculate the total difficulty of the game for a certain player. They test their method using the game Pacman, and conclude that this method can provide great insight into the experience of players. They suggest that game designers can use the data acquired by this method to improve the quality of their games.

András, Sipos and Sóos [6] analyse the difficulty of Happy Cube puzzles. They believe that the difficulty ranking of the puzzles by the manufacturer is wrong, and attempt to find a better ranking. A user study is performed to find the time taken by participants to solve each puzzles. Four new rankings are created by using the average time, the average time without the top and bottom 10%, the median time, and a combination of the three. These rankings suggest that the

ranking by the manufacturer is indeed incorrect, and the authors attempt to find a method capable of better ranking. Multiple mathematical models are devised to attempt to match the new rankings. They create a puzzle solver using a backtracking approach, and data from this solver is then used to create new rankings. This data includes the average number of backtracks, the probability of solving a puzzle without backtracking, and the number of dead ends requiring backtracks. Their method allows new rankings to be created with high correlation to the rankings from the user study.

We note that multiple authors suggest that the difficulty of a level can be found by measuring the time players take to solve that level. We believe that this is not necessarily true, as a level consisting of one hard challenge may require less time to solve than a level consisting of multiple simple challenges. It can also be noted that the number of moves required to solve a level is generally used to model the difficulty of that level. While we agree that there may be a high correlation between the difficulty of a level and the number of moves, we believe that this might not always be enough to accurately rate the difficulty. In our research we let our players rate the difficulty of a level themselves, as we think this may produce more reliable results. We also model the difficulty of a level as a combination of multiple metrics, to attempt to get more accurate ratings of the difficulty.

3 Difficulty Equation

To automatically determine the difficulty of a level, we require a formula that takes a level as its input, and outputs a score. As every game is different, and has other elements defining the difficulty, each game requires its own formula. Our goal is to find a generic model, which can easily be adapted to a large number of different games.

3.1 Models

For our difficulty equation we considered multiple different models. These models have to combine a level's variables in such a way that they determine the difficulty of that level. There are two possibilities here, either treat these variables independently and just add them together, or combine multiple variables into single terms by dividing or multiplying them with each other. The second option is more versatile, but is more complicated to use and fine-tune than independent variables. We therefore chose to keep the variables separate.

There are still multiple options when we use separate variables. We considered using these variables in a polynomial equation, to be able to have a high versatility. This does have the disadvantage that there needs to be a decision made on the degree of the polynomial. Using a high degree polynomial increases the number of weights that need to be trained, resulting in a higher complexity. Using a lower degree decreases the versatility of the equation.

After considering these options, we opted to use a simpler model. Equation (1) shows our final choice of a linear model for our difficulty equation for level L .

$$\text{Difficulty}(L) = \sum_{i=1}^n (w_i * V_i(L)) + w_0 \tag{1}$$

Each game is assigned a number of variables V_i which influence the difficulty of a level in the game. This number, n in the equation, may vary between games. The choice of these variables is explained in Section 3.2. These variables are multiplied with a constant weight w_i , and added together to form a difficulty score. As our model is linear, it enables us to use linear regression to set the values for the weights. This process is detailed in Section 3.3.

3.2 Choosing Variables

For our difficulty equation to be effective, each game requires a set of variables that together are enough to obtain a good estimate of the difficulty. These variables are usually chosen by playing the game, or observing others playing the game, and looking for properties that are common in difficult levels, and uncommon in easy levels, or the other way around. This is mostly a trial and error process, as occasionally the chosen variables are in reality a poor representation of level difficulty. This can be seen in our choice of variables in Section 4.4, where our initial choices did not prove

to be good enough to make a reasonable estimate of level difficulty. As the variables are meant to give a partial measure of the difficulty of a level, we also refer to them as *difficulty measures*.

Things we consider when choosing our variables are the number of variables, and the ease of measuring these variables. As we want to use the difficulty equation for fast classification of levels, we prefer variables for which the value is easily determined. These are usually simple counts of the number of objects of a certain kind. To improve the process of determining the weights in the difficulty equation, as detailed in Section 3.3, we try to limit the number of variables each game uses. Our games use between four and seven variables, and we found this to be enough to make a good estimate of the difficulty. A smaller number of variables does not contain enough data to accurately estimate the difficulty of a level, while a larger number would require a lot more reference data to properly set the weights corresponding to each of the variables.

3.3 Determining the Weights

To find the weights w_i in the difficulty equation, we attempt to fit the equation to a dataset of levels with known difficulty. We set up a user study to obtain difficulty scores for a set of levels. The way these levels were obtained is detailed in Section 4.3. Details on the way we perform the user study are given in Section 5.

Once we obtain this dataset, we attempt to set the weights in the difficulty equation in such a way, that when we apply the equation to the levels from the dataset, a difficulty score is calculated that is as close as possible to the score assigned to that level by the participants in our user study. This is a well researched problem in mathematics, and there are many methods available to solve this problem, each with its advantages and disadvantages.

We first consider using an exact solution. Using *linear programming*[7] it is possible to find a set of weights that exactly match the dataset. However, as the size of the dataset becomes larger than the number of variables, there is no guarantee that a solution exists. As this is the case in our situation, we discarded the possibility of an exact solution. It is also possible to, instead of an exact match, try to match every level in our dataset to within a certain margin by relaxing the constraints in the linear program. If the margin is made large enough, at some point a solution will exist. It is, however, not trivial to determine how large these margins should be, and therefore there is also no guarantee on the quality of the solution. A solution where each level is matched on the edges of the margin might not be preferable to a solution with a closer match on most levels, but a very poor match on some others.

Instead of matching on the actual scores that users assigned to the levels, we also consider using the relative difficulty between levels. When a user has played two levels, and rated one to be harder than the other, a constraint is added to the difficulty equation that the first level has to be rated higher than the other. As users may not play all levels, and not all users will agree with each other, this may introduce a conflicting set of constraints. Nicky Vendrig [8] solved this problem by removing the constraints where the difference in score were the smallest. He ended up removing such a large number of constraints that the final results were not entirely satisfactory.

We finally consider using *linear regression*[9] to set the weights in such a way that the average

difference between the result of the difficulty equation and the user assigned score is minimal. Linear regression has the advantage that it can always find a solution for such problems. It also finds the best possible solution to a certain metric, which in this case is minimizing the sum of squared differences between the regression and the datapoints. We therefore use linear regression as our method for setting the weights in the difficulty equation.

To perform the linear regression, we make use of the mathematical software package *R* version 3.1.0. Using the command `lm()` the software performs a linear regression on the given dataset. We use the results from this program to set the weights in our difficulty equation.

4 Games for our Experiments

4.1 Games

For our experiments we used three different games. We decided to use multiple games to be able to test how well our method can be adapted to different applications. As our aim was to create a general method, we want it to perform well in more than one case. The rules of these games will be explained next.

4.1.1 Flow

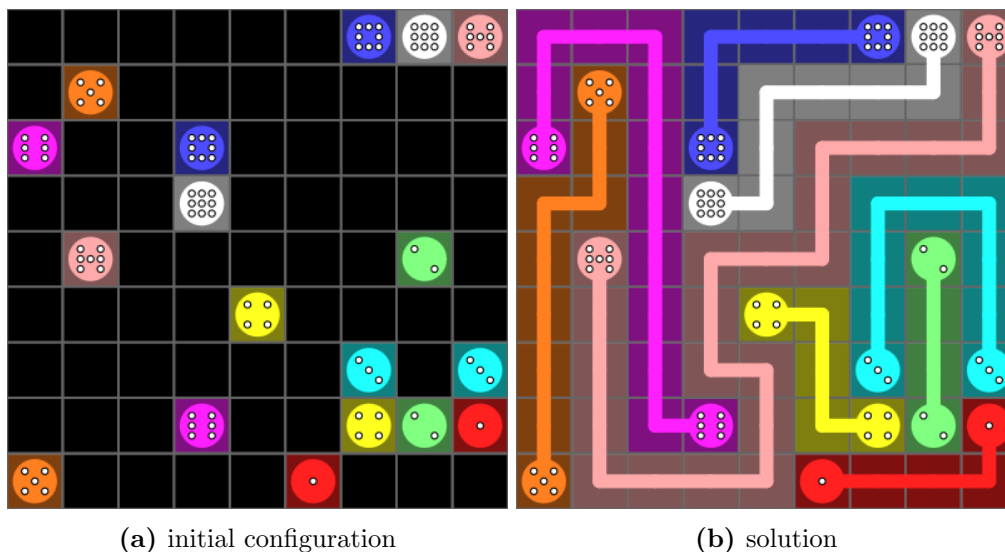


Figure 1: Example of a level in Flow and its solution

The first game we used is Flow by Big Duck Games [10]. Flow is a game laid out on a grid. Placed onto this grid are multiple pairs of balls, where each pair of balls has a different colour. Figure 1a shows a level of the game in its initial configuration. The goal of the game is to draw paths over the grid to connect both balls of each pair. Paths are not allowed to intersect, and the paths should cover the entire grid. The level is solved once all pairs are connected, and the entire grid is covered. An example of a solved level can be seen in Figure 1b.

Although the rules of Flow are fairly straightforward, levels can become difficult when a lot of colours are involved, and paths need to twist around each other to reach a solution. In some levels it is possible to connect all pairs without using the entire grid. This results in holes that need to be filled to reach a solution. Creating a path that fills these holes is also not always a simple task.

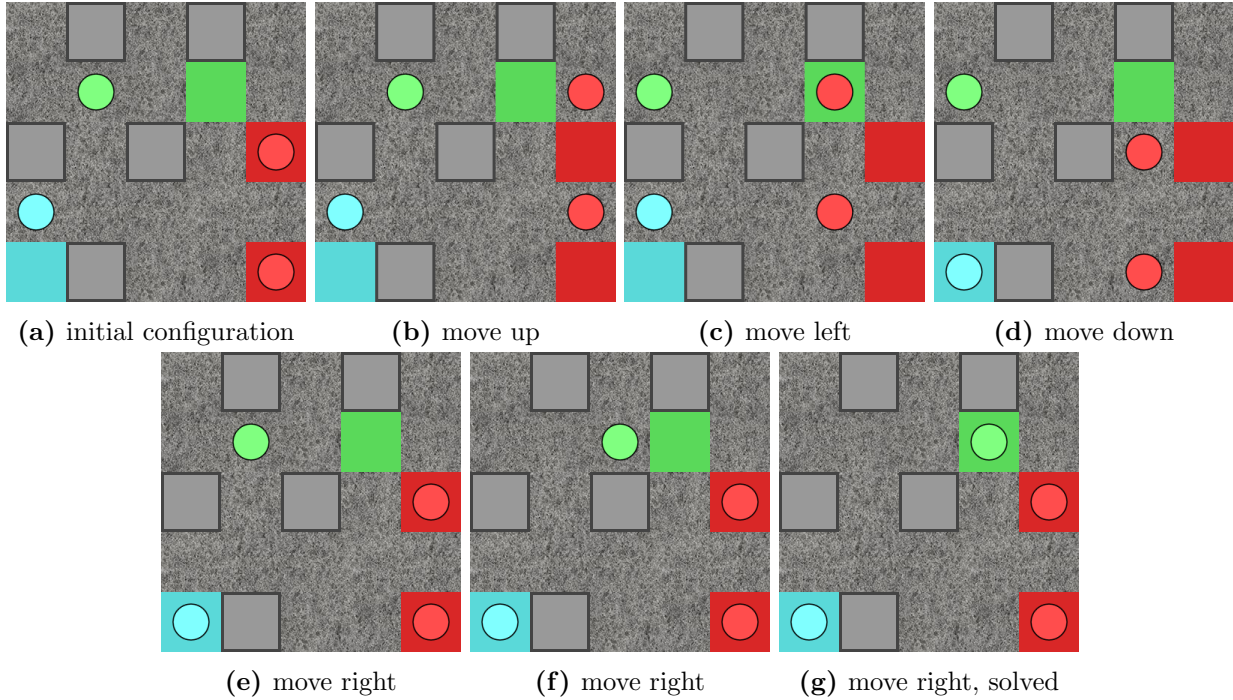


Figure 2: Example of a level in Move and its solution

4.1.2 Move

The second game is Move: A Brain Shifting Puzzle by Nitako Brain Puzzles [11]. The game is played on a small grid onto which three types of elements are placed: balls, goals and rocks. Balls and goals are each assigned a color, with each ball having a corresponding goal of the same colour. Multiple balls can have the same colour, in which case they can be matched to any goal of the same colour. Finally, a number of rocks are placed on the grid. The initial configuration of a level can be seen in Figure 2a.

The goal of the game is to move each ball to a goal of the same colour. Balls can be moved one square left, right, up or down, but the player can only move all balls at the same time. Rocks block the movement of the balls, as do the edges of the grid and other balls. When a ball is blocked, that ball will not move, but the other balls can still be moved in that direction. Figure 2 shows how a level is solved from its initial configuration to the solution shown in Figure 2g.

This game becomes hard as often moving one ball towards its goal results in another ball moving away from its goal. Players must make clever use of the rocks to get each ball in the right position to solve the level.

4.1.3 Lazors

The third game is Lazors by Pyrosphere [12]. In Lazors there are three types of objects. Firstly, there are lasers. These emit a beam of light across the level. Secondly, there are light targets.

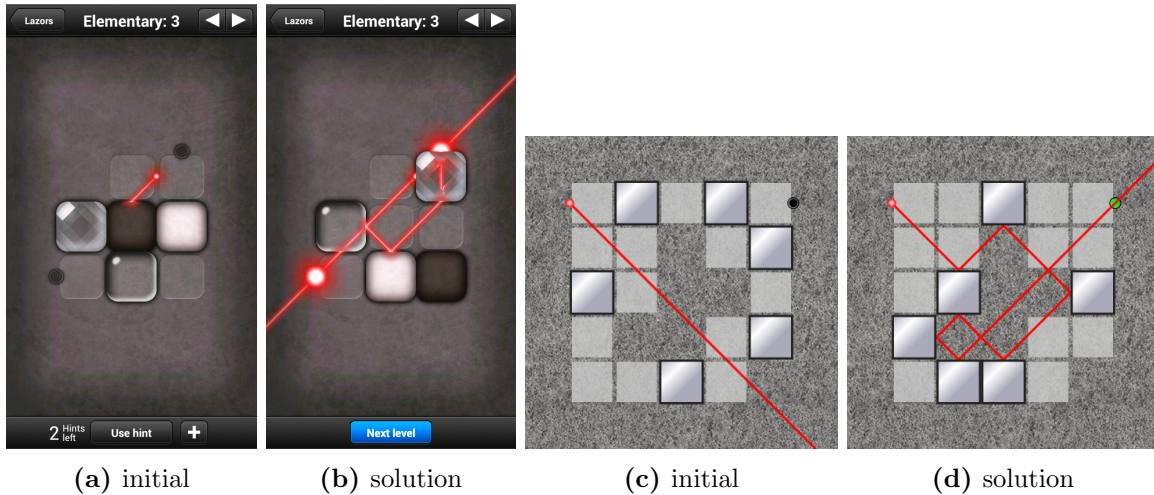


Figure 3: Example of a level and its solution in both the original Lazors game (a, b) and our implementation (c, d)

The targets form the goal of the game. These are activated when there is light shining through it. Once all targets in a level are activated, that level is solved. Thirdly, there are blocks, objects that interact with the light. There are multiple varieties of blocks, including a mirror block to redirect the light, a glass block to split the light into two beams, a diamond block to refract the light, and a black block to absorb the light. The player can move these blocks to a number of open spaces in the level to solve it. Some blocks are locked in place, and cannot be moved by the player. Figure 3a shows a level of the original game containing multiple different blocks, together with its solution in Figure 3b.

Our implementation is a simplified version of the game. In our version only mirror blocks are used, and they can not be locked in place. This is done to reduce the number of variables possibly influencing the difficulty of the level. This means that the difficulty equation can be simpler, and requires less data to find correct weights for the variables. A second effect of this simplification is a reduction of the learning curve for our players. This makes the data produced by our user study more reliable, as players will understand the game faster, thereby reducing the differences in scores given between the first and last played level caused by the improvement of the player’s skill. Figure 3c shows a level in our version of the game, together with its solution, shown in Figure 3d.

4.2 Implementation

All three games were implemented using javascript on an html5 page. The choice for a web-based approach was made to ensure as many people as possible would be able to participate in our user study, and to keep the effort required to participate to a minimum. Javascript also allowed us to have support for both mouse/keyboard control as well as touchscreen devices.

4.3 Levels

The set of levels used in our experiments was chosen in different ways for each of the three games. As each of the games supplied a vast list of levels, those levels were the first to be considered. We also considered the possibility of generating our own levels, which would allow us to specifically test the impact that certain features have on the difficulty, by having multiple similar levels with differences in only one of the features. We decided to base our decision on the characteristics of the original levels. When the original levels have enough variety in the features, we use those levels. We generate our own levels otherwise. The full list of the levels used for our experiments can be found in Appendix A.

In the case of Lazors, we simplified the game to only use one type of block. As most of the levels make use of the other blocks, they are not usable in our implementation. However, as Lazors comes with a large number of levels, there are still enough levels that only use the mirror block. We consider these levels to have enough variation in difficulty and features to be interesting to use. These levels also have the useful property that there is only one unique solution for each level. From this set of usable levels, of which there are 75, we use 65 levels for our experiments. The other 10 levels are used as a tutorial for players to become familiar with the game before starting.

For Flow, in contrast to Lazors, we did not simplify the game. This means that all of the original levels in Flow, with the exception of some of the available expansion packs, are usable in our version of the game. As Flow comes with hundreds of levels, ranging in size from 5 by 5 to 9 by 9, there are enough levels for us to use. For our experiments we chose a set of 40 levels by picking 8 levels for each size at random from the original set of levels.

As with Flow, our implementation of Move contains all elements present in the original game. All levels in the original game are therefore compatible with our version of the game. An inspection of the available levels revealed that the difficulty of the original levels is mostly very low. We consequently decided upon not using these levels, and generating our own levels instead. The algorithm used for the level generation is explained in Section 4.3.1. We used this algorithm to generate a set of levels, from which we picked 80 levels in such a way that for $3 \leq x \leq 10$ there were 10 levels solvable in x moves. The process for selecting levels is detailed in Section 4.3.2.

4.3.1 Level Generation in Move

In this section we explain our method for generating levels for Move. An overview of this method can be found in Figure 4.

First, the size of the level is determined. The size is picked at random without bias between the minimum and maximum size. As with the original game, we use a minimum size of 3 by 3, and a maximum size of 5 by 5, and only square levels are used.

Secondly, the number of balls is chosen at random. We use a minimum of 3 balls, and a maximum of 5 balls, similar to the original game. The number of different colours for the balls is chosen as a random between one colour and each ball a different colour. These colours are distributed randomly over the balls. This means that when 5 balls and 3 colours are chosen, both

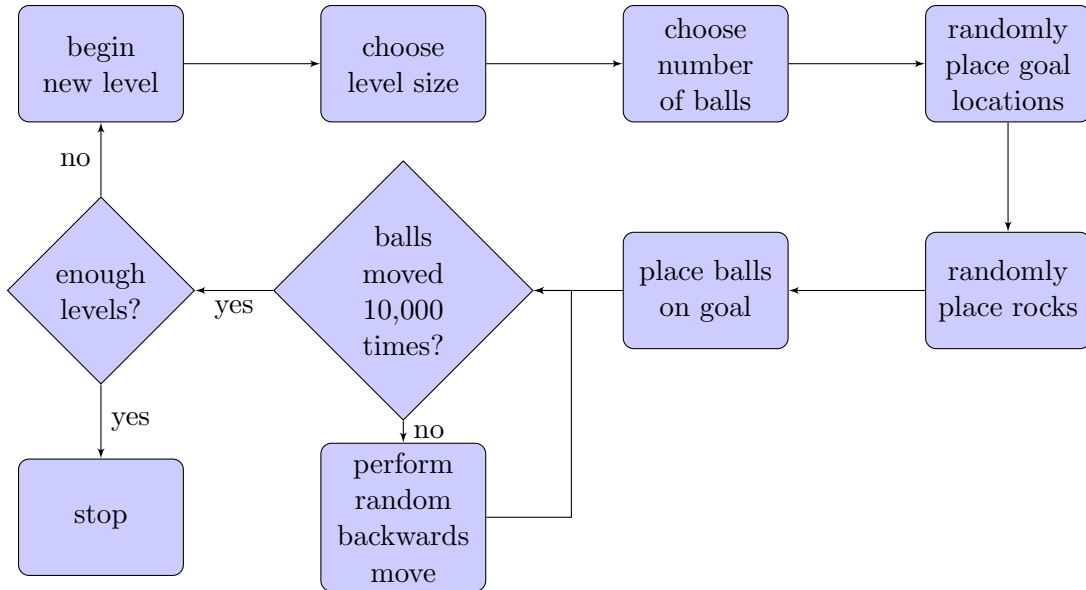


Figure 4: Outline of the level generation process for Move

a 2 : 2 : 1 and a 3 : 1 : 1 ratio between the colours are possible.

Thirdly, goal locations for each ball are randomly placed into the level, in such a way that no two goal locations occupy the same space.

Fourth, rocks are randomly placed in the remaining empty space of the level. We arbitrarily choose a probability of 30% of placing a rock on a tile. As we consider the generated levels to have a reasonable number of rocks, no other probabilities were tested.

Fifth, the balls are placed into the level. A possibility is to place the balls randomly into the level. This would, however, result in the possibility of unsolvable levels, which would require generating many more levels, as most levels will be unusable. We chose instead for a backtracking approach. We start by placing the balls at their respective goal locations. We then perform a backwards move to find a gamestate that can lead to the solved state in one move.

The backwards move is performed by first randomly selecting a direction to do the move in. For each ball it is then checked if it could possibly arrive at its current position by that move. If the move is possible, a decision is made for each ball to move it in the opposite direction of the move. When the ball is blocked in the direction of the move, by a rock, the edge of the level, or another ball, it can also stay in its current position. If both a move and the current position are a possible option for a ball, a biased random choice is made, with a 30% probability of staying in the same place. This bias is introduced to slightly decrease how much the balls are scrambled with respect to each other, which made a lot of the levels too difficult. A total of up to 10,000 backwards moves are performed on the level, to make as many starting configurations as possible reachable, while keeping the computation time low.

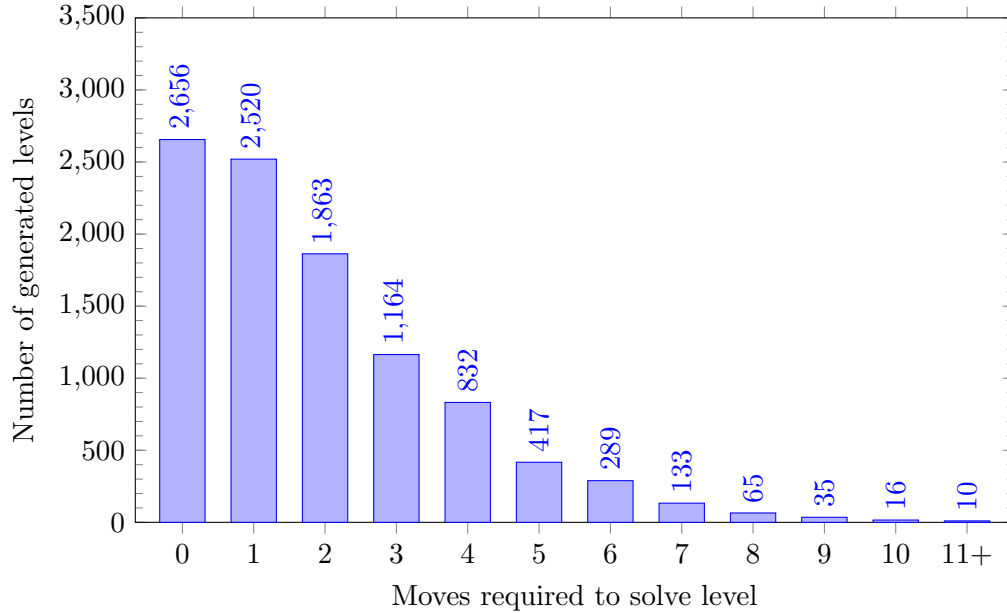


Figure 5: Distribution of the number of moves required to solve a level for Move, from a set of 10,000 generated levels

4.3.2 Level Selection in Move

Our main criteria for the set of levels is that it contains a similar number of easy and hard levels, and levels are not too hard or too easy. As these criteria involve difficulty, and we do not have a clear indicator of difficulty, we use the number of moves required to solve a level instead of actual difficulty. We compute the required number of moves with a level solver, detailed in Section 4.3.3. As can be seen in Figure 5, levels with a low number of required moves are generated significantly more often than levels requiring more moves. This forces us to perform some balancing on the set.

We made the choice to use 10 levels for each number of moves required. Levels that can be solved in two moves or fewer are removed from the set, as we consider these to be too easy. Likewise, levels requiring more than 10 moves are also removed, to prevent levels being too hard, as hard levels can be discouraging to players, and may cause them to quit. Note that this process already removes 70% of the levels. The final choice of levels was made by first generating levels until there were 10 levels for each number of required moves. We then picked 10 levels for each number of moves at random from this set.

4.3.3 Level Solving in Move

To be able to compute the minimum number of moves to solve a level in Move, we make use of an automated level solver. The method we use to solve these levels is explained in this section.

We use a general pathfinding approach to solve our levels. The pathfinding is performed on a directed graph, in which each node represents a possible gamestate, and each edge represents a

move, that, when performed on the gamestate of the first node, results in the gamestate of the second node. Each node has four outgoing edges, one for each possible move, which may connect to the node itself.

The pathfinding algorithm we use is based on the *A* algorithm*[13]. We use the number of moves performed as the distance, therefore giving each edge of the graph a distance of 1, as it represents one move. We try to find a path from the initial gamestate to a solved gamestate. There may be multiple solved gamestates when multiple balls share the same colour.

The A* algorithm requires a *heuristic function* which approximates the distance to the goal. This function should always underestimate the distance to guarantee that the algorithm finds the shortest solution. The heuristic we use to estimate the distance from a certain gamestate to a solved gamestate is shown in equation (2).

$$H(s) = \max_{b \in B} (\min_{g \in G_{\text{Col}(b)}} (\text{Dist}(\text{Pos}(b, s), \text{Pos}(g, s)))) \quad (2)$$

Here s is the current gamestate, B is the set of balls in the level, G_c is the set of goals of colour c , $\text{Col}(b)$ is the colour of ball b , $\text{Dist}(p_1, p_2)$ is the cityblock distance between p_1 and p_2 , and $\text{Pos}(o, s)$ is the position of object o in gamestate s . This heuristic represents the furthest distance for a ball to its closest goal position. This is guaranteed to be smaller than the actual distance, and therefore the A* algorithm is guaranteed to find the shortest path.

4.4 Difficulty Measures

In this section we detail the difficulty measures we use in the three games. We will explain the meaning of the measures and how the values can be interpreted. We also show how these measures are expected to influence the difficulty of a level.

Our goal for the difficulty measures is to find a small set of features for each game, that together are enough to give a good estimate of the difficulty of a level in that game. We want this set to be small, as larger sets of features can lead to overfitting on our dataset, causing our difficulty equation to work very well on our dataset, but not on others. It is also harder to find the proper weights for larger amounts of features. We prefer simple to define features over harder features, as we want our equation to be easy to adapt to different games.

We identify three different types of difficulty measure. The first type is *initial features*, and includes features of the level in its initial configuration. Examples of initial features are the size of the level, and the number of game elements the player can interact with. These values for these measures are usually easy to determine for a given level, and it is usually easy to constrain these values when generating levels, which helps with the possibility of creating levels at a given difficulty once the weights of the difficulty equation are known.

The second type we identify is *solution features*. This type includes features of the level in its solved configuration. An example of a solution feature is the location and state of game elements in the solution. They can in some cases be more useful than initial features, as for certain games, the solved state says more about the difficulty than the initial configuration does. Values for difficulty measures of this type are slightly harder to determine, as they require a level to be solved first.

In some cases, however, levels are generated by starting with the solution, and backtracking to an initial configuration, as we do with our level generator for Move in Section 4.3.1, meaning that level generation to a specified difficulty does not become harder.

The last type we identify is *dynamic features*, including all features related to the process of solving a level. Examples of dynamic features are the number of moves required to solve a level, and the type of moves used. The values for these difficulty measures are usually the hardest to determine, as they often require a solver that is capable of finding a solution in a minimal number of moves. We expect these measures to be the best at determining the difficulty of a level, as levels are often not difficult because of the final state, but because of the process required to get there.

It should be noted that dynamic measures are not relevant for all games. In Flow and Lazors, only the final state of the game is important, while the way the player got to that solution is not. Similarly, some games are not suited for solution features. An example of such a game is Move, where the final state of the game is immediately obvious from the initial configuration. Any solution features could therefore also be modeled as initial features.

4.4.1 Flow

For the game Flow, we use the following list of features as variables in our difficulty equation:

- **Level size (initial):** The size of the level, defined by the number of squares on the board. We expect larger levels to be more difficult, as the player has more space to fill. The creators of the original game use the size of the level as their only indication of level difficulty.
- **Colours (initial):** The number of different colours to connect to each other. With smaller numbers of colours, paths will need to fill more space, and become longer, and possibly more difficult. With larger numbers of colours, we expect the paths to block each other more, which may also lead to higher difficulty. We therefore do not know beforehand how this measure will affect the difficulty of a level.
- **Average distance (initial):** The average distance between the start- and endpoints of the same colour. We measure this using city block distance (L_1 distance). When this value is low, it means that the paths of the solution are more indirect, as the complete level still needs to be filled. We therefore expect lower values to cause higher difficulty.
- **Corners (solution):** The number of corners that are made in the solution. As some levels have multiple solutions, we use the solutions shown in Appendix A.1. More corners would imply that the connection between start- and endpoints is more indirect, which we assume would lead to a more difficult puzzle.

A list with all the values of the difficulty measures for each level can be found in Appendix B.1.

4.4.2 Lazors

For the game Lazors, we initially used the following set of features for our difficulty equation:

- **Level size (initial):** The size of the level, defined by the area of the smallest bounding box around the usable tiles. Our intuition is that larger levels would be more difficult. However, not all of the level is necessarily used, and a larger level may have a lower density of laserbeams, which would lead to less clutter.
- **Usable tiles (initial):** The amount of usable tiles in the level. A higher number of open tiles leads to higher numbers of possible mirror placements, which might make a level more difficult. Lower numbers restrict the number of mirror placements, but may also make some easy solutions impossible. We therefore are unable to predict how the difficulty of a level is affected by this measure.
- **Emitters (initial):** The number of laser emitters in the level. We expect the difficulty of a level to increase as more laserbeams are used. When multiple lasers are used, it becomes unclear which laser will activate what target. A high amount of laserbeams also adds clutter to the level, sometimes making the solution less obvious.
- **Receivers (initial):** The amount of targets that need to be activated to solve the level. A higher amount of targets increases the amount of work that needs to be done for a solution. It might, however, give the player a more clear image of the path the laserbeam needs to take, making the solution more obvious. We were unable to predict how this measure would affect the difficulty of a level.
- **Mirrors (initial):** The amount of mirrors in the level. When the player has more mirrors to move around, the number of possible placements for these mirrors increases (as long as there are less mirrors than half the amount of usable tiles). A larger amount of mirrors is also often related to more interactions between the lasers and the mirrors. Because of this, we expect level difficulty to be higher when there are more mirrors.

In our first experiments, this set proved to not be accurate enough (for further detail on this, please refer to Section 6). We therefore added the following difficulty measures to improve the accuracy of the difficulty equation:

- **Reflections (solution):** The amount of times a laser reflects from a mirror in the solution. This measure indicates how indirect the path of the lasers is. A more indirect path would be less obvious to players, and therefore more difficult.
- **Intersections (solution):** The amount of times two laserbeams cross path in the solution. We use this as a measure of how cluttered a level is. We expect players to lose track of the exact paths of the lasers when they cross each other often. Our expectation is that high values for this measure will lead to more difficult levels.

A list with all the values of the difficulty measures for each level can be found in Appendix B.2.

4.4.3 Move

For the game Move, we started out using the following set of features for our difficulty equation:

- **Moves (dynamic)**: The minimum number of moves required to solve a level. We expect that more required moves results in a more difficult level. The correlation between moves and difficulty may however not be very strong, as some moves can be trivial, and some can be counterintuitive.
- **Level size (initial)**: The size of the level, defined by the number of squares on the board. Larger levels can result in more freedom to move the balls around. This could have either a positive or negative effect on the difficulty, as the player has more possibilities to choose from, but there may also be more solutions.
- **Balls (initial)**: The number of balls controlled by the player. As more balls come into play, it becomes more frequent that moving one ball towards its goal moves another ball away from its goal. We therefore expect a larger number of balls to increase the difficulty of a level.
- **Colours (initial)**: The number of unique ball colours. A higher number of unique colours can be harder, as there are fewer possible goal positions for each ball, and therefore fewer solutions. When multiple balls share a colour, they can use any goal of the same colour, but not all goals are always reachable, making it unclear to the player which ball should go to what goal. This could also lead to increased difficulty when the number of colours is lower than the number of balls.
- **Rocks (initial)**: The number of rocks in the level. Rocks are used to block the balls, and may result in higher difficulty by blocking the path. It is however also possible that the rocks help the player to move the balls relative to each other, which is often a critical step in solving a puzzle.

Our first experiments pointed out that this set of features was not enough to accurately assign a difficulty score to a level, as can be seen in Section 6. To increase the accuracy of our difficulty equation, we added the following difficulty measure to the set:

- **Counterintuitive moves (dynamic)**: The number of moves in the shortest solution that are counterintuitive. We consider a move to be counterintuitive when the average distance between the balls and their closest goal of the same colour increases in that move. As the intuitiveness of a level is closely related to the difficulty, we expect this measure to have a very strong correlation with the difficulty of a level.

A list with all the values of the difficulty measures for each level can be found in Appendix B.3.

5 User Study

To be able to set the weights in our difficulty equation in such a way that it accurately predicts the difficulty of a level, we require data to match the weights to. This data comes in the form of a set of levels and the difficulty rating they should have. Our set of levels does not include difficulty ratings in a consistent or usable form. In Lazors the levels are grouped together, and the group is given a range of difficulties in text form (e.g. Medium-Hard), the levels in this group are roughly ordered by difficulty, but no clear per-level difficulty is given. The levels of Flow also have a difficulty rating in text, but this rating is only based on the size of the level, and not the actual content of the level itself. While level size might be the only important factor in difficulty, we can not be sure about this. Move does not have a difficulty rating for its levels. We therefore require another approach to collect this data for our levels.

We decided on conducting a user study to assign difficulty ratings to our levels. Players were invited to play our levels, and rate the levels on their difficulty. These scores are then used with our levels to match the difficulty equation to. This section describes how this user study was set up.

5.1 User Data

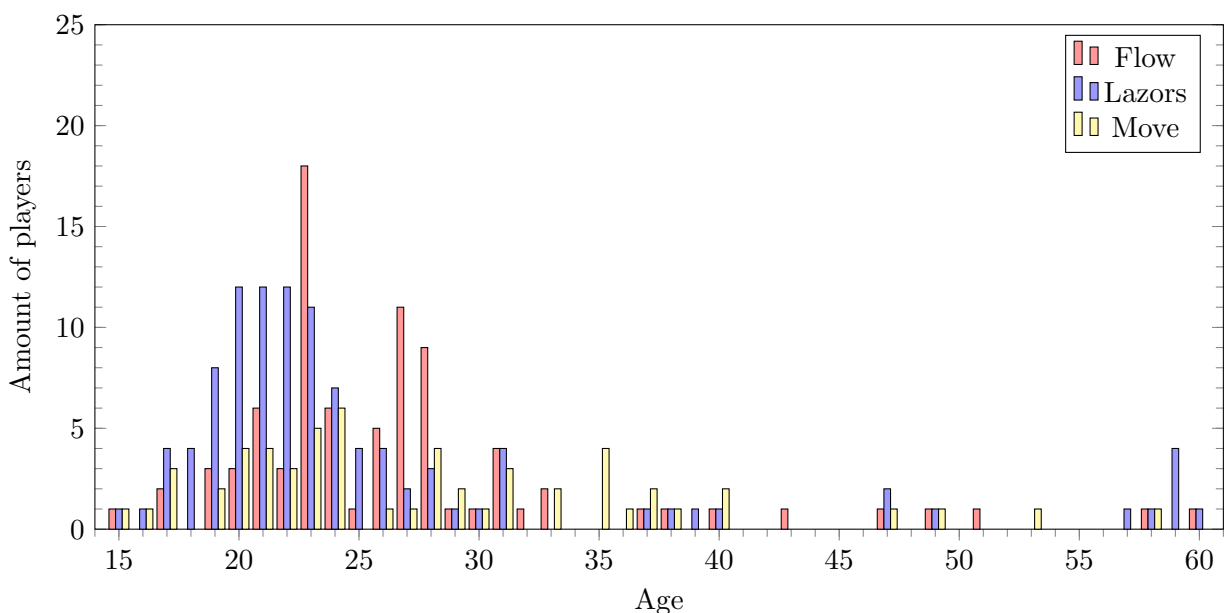


Figure 6: Amount of players of each age participating in the user study

For the players we contacted fellow students, friends and family. We also made a post on a puzzle game website, asking people to join the user study. Not all players participated in all games, therefore the data on our players is separated between the games. In total, 86 participants played Flow, 105 participants played Lazors, and 57 participants played Move. Figure 6 shows the age distribution of the players who participated in the study. Figure 7 shows the highest level

of education players have achieved. It can be noted that most players are students in their early twenties.

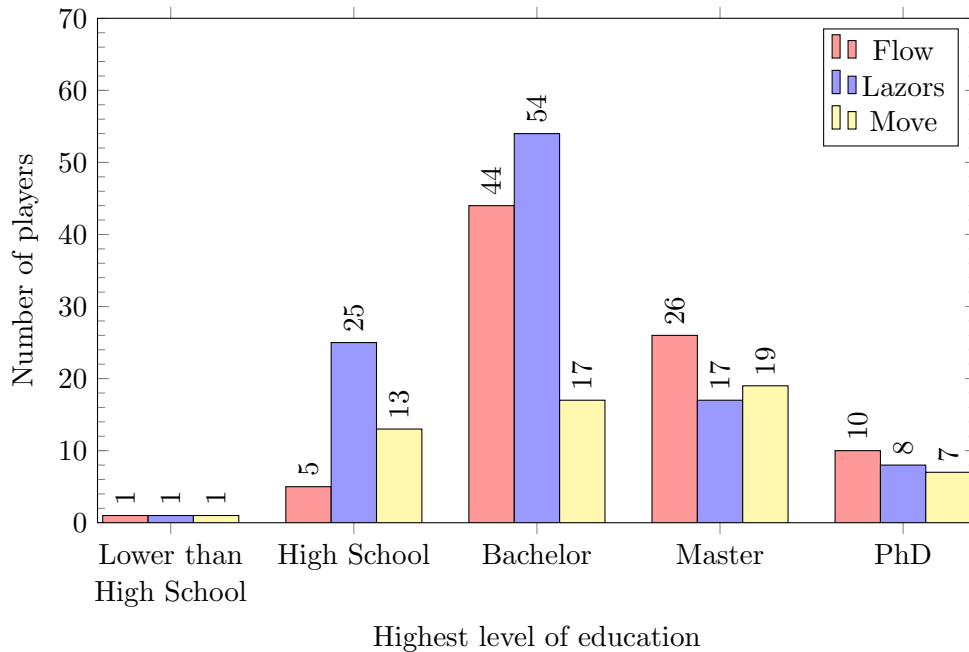


Figure 7: Distribution of education level of players participating in the user study

5.2 Study Setup

One problem we faced when setting up the study was coping with the effects of learning. When players first start with one of the games, they might not be familiar with the games, and experience the first few levels as being more difficult. This may lead to users giving higher difficulty ratings to those levels, due to the fact that they are still learning how the game works. To deal with this, we first present the players with a short tutorial. This tutorial, which is presented at the start of each of the three games, consists of 10 levels, gradually increasing in difficulty. These levels are specifically chosen in such a way that all elements of the game are familiar to the player before starting with the actual rating process. For players that have played the game before, an option to skip the tutorial is also presented.

This tutorial phase cancels out most of the learning effect, but some of it may still remain during the rating phase of the study. To further cope with this problem, the levels are presented to the players in a random order. This is done so that the effect of learning should be cancelled out over multiple plays of a level by different users, as some users are presented with the level early in the rating phase, and others later. We do not require users to complete all of the levels, in order to increase participation by players not willing to spend a long time rating the puzzles. A side effect of this is that the number of times a level is played does not increase at the same rate for all levels, and some levels might end up with hardly any ratings. To correct for this, a bias is given to levels with fewer ratings when selecting the next level.

5.3 Level Rating

Once the player has finished a level, that player is asked to rate the difficulty of that level. This is done on a 1-10 scale, 1 being the easiest level, 10 being the hardest. The player can only proceed to the next level once the current level has been rated. Players are specifically asked to use as much of the scale as possible, and to not be afraid of giving ratings of 1 or 10.

Players are able to revert the level to its initial configuration by clicking a reset button. This helps players to erase mistakes they may have made while solving the puzzle, and can avoid levels becoming unsolvable. Note that only levels in Move can have states where the level is no longer solvable.

If a player is unable to solve a level, an option is presented to skip that level. This option only becomes available after one minute, to encourage players to keep trying if they can not solve the level immediately. We considered multiple options for how we handle skipped levels. Our first option was to give skipped levels a fixed difficulty rating above the highest rating. We found this option to be hard to justify, as an arbitrary value had to be given to skipped levels. It would be very hard to find arguments for why a certain score is better than another. We therefore decided against this. Instead, we chose to ignore skipped levels during the linear regression used to set the weights.

As different users might not be consistent with their assigned scores, all separate scores are used in the linear regression to match the difficulty equation to these scores. We use two different methods for this. The first method uses all scores directly as they were given by the players. We did, however, notice that some players tend to give higher scores, and some only give low scores. To correct for this, we also use a second method. Using this method, the assigned scores of each player are scaled linearly in such a way that their lowest rating becomes 1, and their highest rating becomes 10. We refer to these scores as the *scaled difficulty rating*.

The complete list of average difficulty ratings, average scaled difficulty ratings, average level solving times, and average moves used, for every level in each game, can be found in Appendix C.

6 Results

In this section we show an overview of the results of the user study, and the results of applying the data from the user study to set the weights in our difficulty equation. We evaluate the performance of the difficulty equation, and discuss the major contributing factors on the performance.

6.1 Flow

We start by comparing the results of the user study to the values for the difficulty measures. Besides the difficulty rating provided by the users, we also consider the scaled difficulty rating (as explained in Section 5.3), the time taken to solve a level, and the number of moves performed by the users. We define a move in Flow as the drawing of one (partial) path, from the moment the mouse button is pressed, to the moment the mouse button is released. We enter these values into R to determine the linear correlation coefficients between these values. These coefficients can be found in Figure 8.

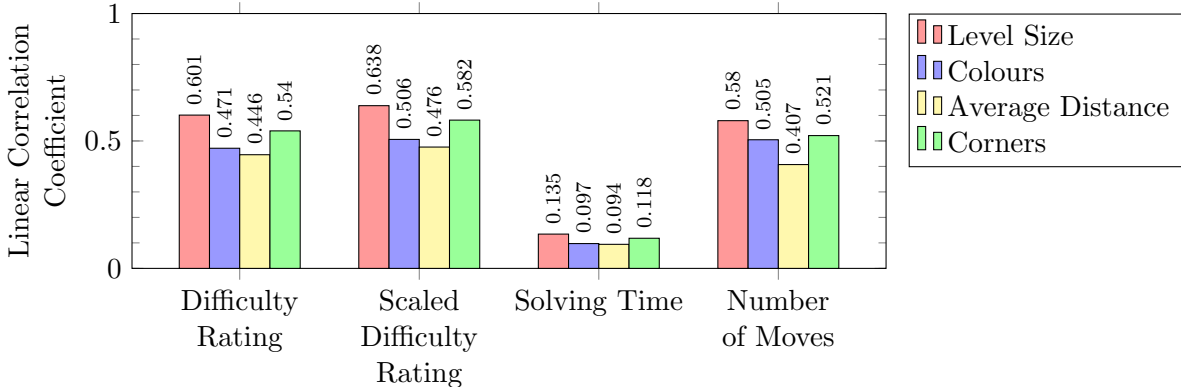


Figure 8: Linear correlation coefficients between user difficulty ratings, solving times and number of moves, and difficulty measures for Flow

It can instantly be noticed that the correlation coefficients with the solving time are a lot lower than the others. This can be explained by a very high variation between users for the time taken to solve each level. While users tend to agree on difficulty ratings, and solve the level in a similar number of moves, some players are significantly faster than others. This variation leads to lower correlations.

A look at the difficulty measures shows that the size of the level has the highest correlation with all four aspects, followed by the number of corners, the number of colours, and lastly the average distance between endpoints. Even so, the average distance measure still shows a fairly high correlation with all aspects, except the solving time.

Using R again, we are able to perform multiple linear regressions on the data from the user study. This allows us to find the weights for the difficulty measures in our difficulty equation. We also apply the same method to find an equation for the other three aspects. While our difficulty equation is not intended for this use, we are also interested in its performance for aspects other

than difficulty.

We split our data into 5 sets, each containing the data for 8 different levels in order, such that set 1 contains the data for levels 1–8, set 2 contains the data for levels 9–16 and so on. We train the difficulty equation for each set using the data from the other four sets, so that the difficulty equation for each set is not influenced by the data in this set. This way we can perform a proper analysis on how the difficulty equation would perform on new levels outside of our dataset. This process is referred to as *cross-validation*. Table 16 in Appendix D.1 shows the weights for the difficulty measures for each of the four aspects in each set.

It can be seen that some difficulty measures are assigned significantly higher weights than others. It should be noted that this does not imply a higher importance for the corresponding aspect, as the values for the difficulty measures are not normalised.

We use the difficulty equation to calculate the difficulty for all levels, so that we can compare this to the original ratings by the users. We also do this for the other three aspects. This comparison can be seen in detail in Appendix C.1, and an overview of the average error is presented in Table 1.

	Set	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
Weighted	1	0.3850	0.5979	61.96%	18.59%
	2	0.4713	0.6195	30.53%	8.71%
	3	0.3220	0.5767	66.28%	10.37%
	4	0.3628	0.6043	33.58%	14.82%
	5	0.4757	0.8111	24.52%	7.47%
	Average	0.4040	0.6423	43.21%	11.86%
Unweighted	1	0.3970	0.6117	57.80%	18.81%
	2	0.4874	0.6423	30.14%	8.99%
	3	0.3189	0.5752	67.10%	10.30%
	4	0.3803	0.6269	34.16%	15.40%
	5	0.4681	0.7949	25.00%	7.45%
	Average	0.4103	0.6502	42.84%	12.19%

Table 1: Errors in each of the level sets of Flow produced by the difficulty equation when compared to user study data. Shown for difficulty, scaled difficulty, time, and number of moves, both unweighted and weighted by number of level ratings.

The difficulty equation gives us an average error of 0.4103 points, or 0.4040 points when we weigh the error for each level by the amount of times that level has been played. For the scaled difficulty rating this error is 0.6502 points, or 0.6423 when the error is weighted. This is noteworthy, since we introduced the corrected score to decrease the error, but it actually increases. This can partly be explained by the fact that the scaled difficulty makes use of a larger range of scores. The range is, however, only approximately 1.3 times as large on the average, and this accounts for only half of the difference.

For the solving time aspect, we measure the error in a percentage of the average time users took to solve the level, as the time is unbounded, and a 5 second deviation on a total of 10 seconds is more significant than the same deviation on a 200 second total. This results in an average error of

42.84%, and when we introduce weights for the levels, this error increases to 43.21%. As expected from the correlation coefficients in Figure 8, the results for predicting the average level solving time are quite poor.

Lastly, for the number of moves our equation gives an average error of 12.19%, or 11.86% when the error is weighted. We consider this to be a surprisingly good result, as our difficulty equation is not originally designed to calculate the number of moves, but the difficulty of a level. It should be noted, however, that the number of moves performed in Flow is fairly meaningless, as different players play in different ways. Some players were observed to create the paths segment by segment, others drew larger partial paths with each move. A third group of players only drew a path in its entirety once they were certain about it. The aspect of number of moves was mainly added for the other two games, where it has a more obvious meaning.

6.2 Lazors

As with Flow, we also begin by analysing the user study data. We find the correlation between the difficulty measures, and the difficulty rating, scaled difficulty rating, time taken to solve the level, and the amount of moves used to solve the level. One move in Lazors is defined as moving one of the mirror blocks to another free tile. Invalid moves, such as moving a mirror to an already occupied tile or out of the level, are not counted as moves. When we enter the data into R we are able to calculate the linear correlation coefficients between these values. The results of this can be found in Figure 9.

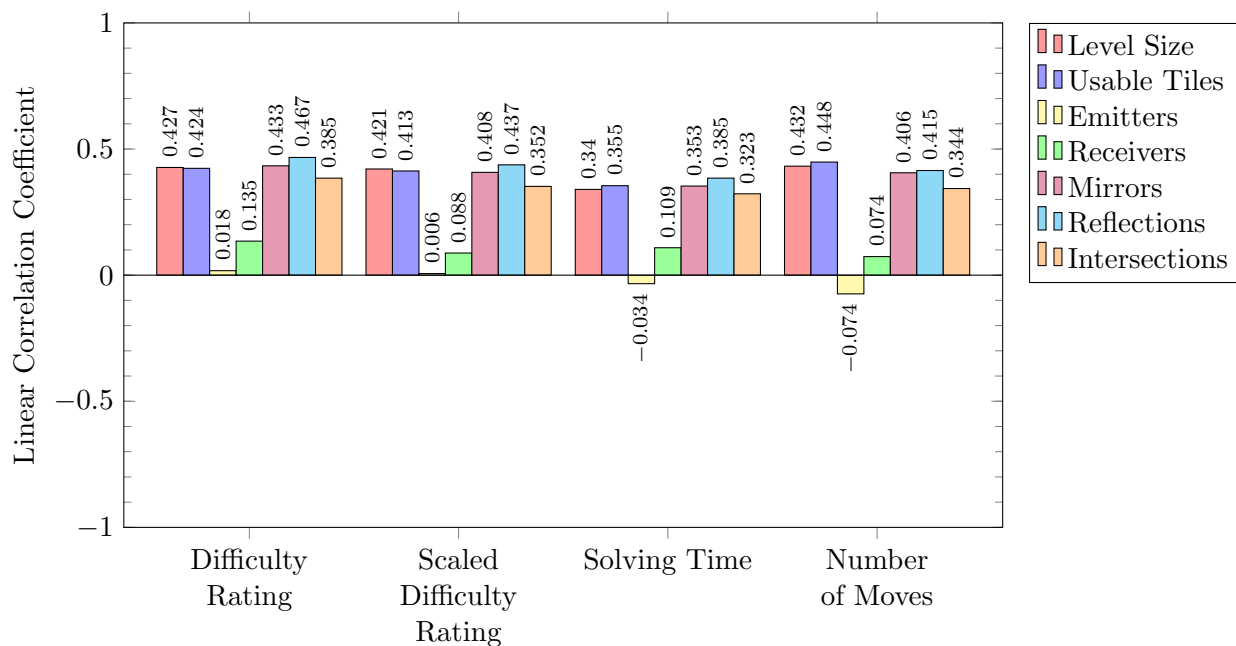


Figure 9: Linear correlation coefficients between user difficulty ratings, solving times and number of moves, and difficulty measures for Lazors

In comparison to Flow, there is a much smaller difference between the four aspects. The

correlations with the level solving time are significantly higher than in Flow. A look at our raw data shows that the variation in solving time between players are a lot smaller in Lazors, which explains this difference.

When looking at the separate difficulty measures, it is immediately obvious that two of the measures have significantly lower correlation coefficients than the other measures. The number of laser receivers seems to only have a small effect on the difficulty of a level. The number of laser emitters has hardly any influence on the level difficulty at all. The correlation coefficient for the emitters on both difficulty aspects is on the same scale as noise data. For the other two aspects, there is a very small negative correlation. This can be explained in two possible ways. Firstly, it is possible that there just is not any correlation between the amount of lasers and the difficulty. Secondly, there is not enough variation in our level set for any correlation to become obvious. Our level set mostly contains levels that have only one or two lasers, with only one level having four lasers. A more varied level set is needed to make a judgement on this.

All other difficulty measures have similar correlation coefficients, which are all moderately high. Note that this set of difficulty measures already includes the two measures that we introduce later, namely the reflections and the intersections measures.

Using R , we perform a linear regression on the data for difficulty, scaled difficulty, level solving time, and number of moves. This way we are able to find the weights for our difficulty equation with the original set of difficulty measures (not including the reflections and intersections measures yet). We again make use of cross-validation with 5 sets of levels, this time using 13 levels per set. The resulting weights can be found in Table 17 in Appendix D.2.

	Set	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
Weighted	1	1.1179	1.4191	90.42%	67.16%
	2	1.2814	1.8489	175.31%	134.34%
	3	1.4255	1.7493	116.60%	138.26%
	4	0.8875	1.3183	77.17%	57.17%
	5	0.7242	1.0129	72.54%	50.91%
	Average		1.0645	1.4453	105.03%
Unweighted	1	1.1013	1.3878	84.75%	62.69%
	2	1.2556	1.8744	152.43%	116.02%
	3	1.6076	2.0027	87.40%	96.37%
	4	1.0090	1.5334	68.25%	49.46%
	5	0.7281	1.0207	71.04%	50.20%
	Average		1.1403	1.5638	92.77%

Table 2: Errors in each of the level sets of Lazors produced by the difficulty equation when compared to user study data. Shown for difficulty, scaled difficulty, time, and number of moves, both unweighted and weighted by number of level ratings.

As is the case with Flow, the values for the difficulty measures are not normalised, and therefore no conclusions should be drawn from the relative values of the weights directly. It is, however, notable that the weight assigned to the level size is positive for both difficulty aspects, and negative in the other two aspects, with the exception of set 5.

We compare the outcome of the difficulty equation to the ratings provided by the users. Details of this comparison can be found in Appendix C.2, and an overview of the average errors is presented in Table 2. On average, our calculated difficulty has a deviation of 1.1403 with the difficulty rating from the user study. When the errors are weighted by the amount of times each level has been played, this average error becomes 1.0645. Using the scaled difficulty ratings instead, an average error of 1.5638 is achieved, when this is weighted the error becomes 1.4453. As with Flow, this increase can partially be explained by the use of a larger range of scores. The corrected scores use a range that is on the average approximately 1.1 times as wide as the original difficulty scores, which is responsible for less than a third of the difference.

For the time taken to solve a level, despite the higher correlation coefficients than Flow, we achieve an average error of 92.77%, or 105.03% when weighted. The equation for the number of moves produces an average deviation of 74.95%, when weighted this becomes 86.09%. Although this is significantly higher than Flow, the average amount of used moves is also a lot higher, which accounts for most of the difference.

As we do not consider these results to be completely satisfactory, we introduce two new difficulty measures. The reflections difficulty measure and the intersections difficulty measure are added to the set. After this, we again perform linear regressions on the data. The resulting weights are shown in Table 18 in Appendix D.2, and the resulting errors are presented in Table 3.

	Set	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
Weighted	1	0.9492	1.4091	84.22%	59.35%
	2	1.3387	1.9147	260.21%	183.96%
	3	1.2285	1.5492	87.52%	99.28%
	4	0.7506	1.2036	53.05%	33.01%
	5	0.8657	1.1462	57.88%	45.64%
	Average	1.0138	1.4337	108.39%	82.54%
Unweighted	1	0.9284	1.3772	78.73%	56.34%
	2	1.3164	1.9066	224.31%	158.30%
	3	1.3751	1.7592	70.25%	70.55%
	4	0.9950	1.5228	53.48%	33.81%
	5	0.8636	1.1391	57.20%	45.17%
	Average	1.0957	1.5410	96.79%	72.84%

Table 3: Errors in each of the level sets of Lazors produced by the improved difficulty equation when compared to user study data. Shown for difficulty, scaled difficulty, time, and number of moves, both unweighted and weighted by number of level ratings.

After adding the two new difficulty measures, the average difficulty error becomes 1.0957 (weighted: 1.0138). The scaled difficulty score gets an error of 1.5410 (weighted: 1.4337). The deviation for the time taken to solve a level increases to 96.79% (weighted: 108.39%). Lastly, the error for the amount of moves is improved to 72.84% (weighted: 82.54%).

It can be noted that most of the error comes from level set 2. Looking at the individual levels it becomes clear that this error is mostly caused by three levels (level 17, 19 and 24). These levels all have two things in common, which are a low rating by the users, and a large number of laser

receivers. The weights for the number of receivers are significantly higher in set 2 than in the other four sets, which causes this deviation.

We can see that the two new difficulty measures slightly improved the performance of the difficulty equation. It does, however, not perform as well as it does in Flow. We assume this is mostly due to the fact that Lazors is a more complex game than Flow is. We expect that more complex games will generally have poorer results for predicting difficulty than simpler games.

6.3 Move

For Move, we also begin by analysing the data from the user study. We use R to find the linear correlation coefficients for each of the difficulty measures compared to the user-rated difficulty, the scaled difficulty rating, the time taken to solve a level, and the number of moves used to solve a level. The resulting coefficients are presented in Figure 10.

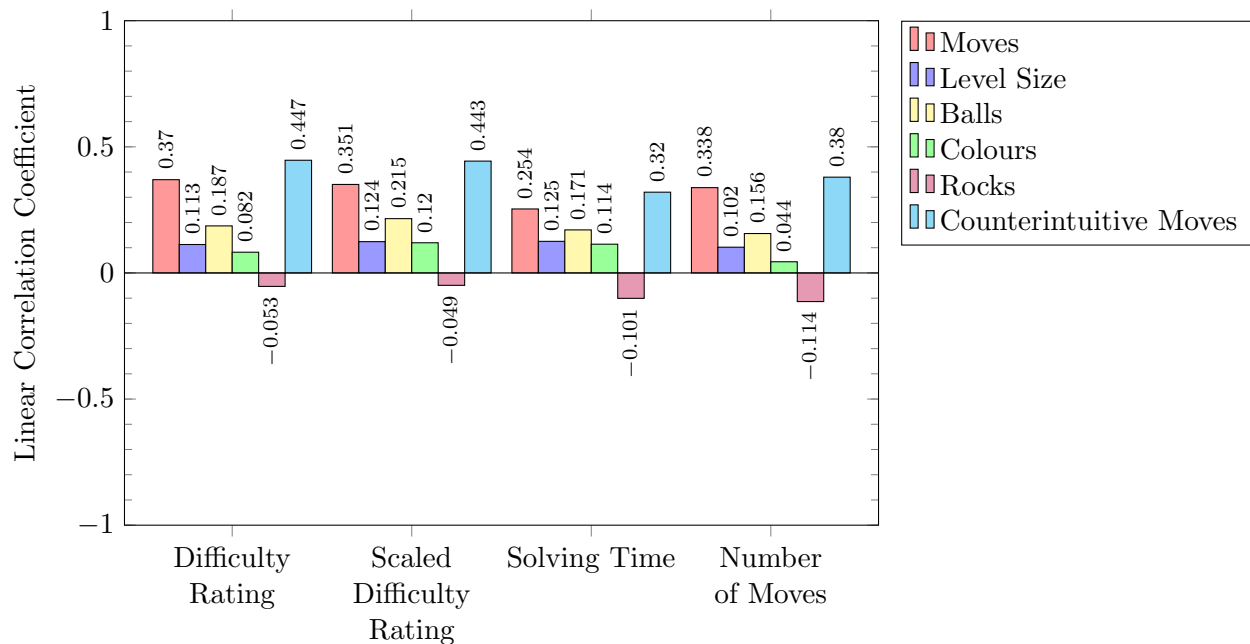


Figure 10: Linear correlation coefficients between user difficulty ratings, solving times and number of moves, and difficulty measures for Move

Note that this figure already includes the counterintuitive moves difficulty measure. This measure is not used initially, and is only added later to improve the results.

The correlation coefficients follow a similar pattern across all four aspects. The counterintuitive moves shows the highest correlation, closely followed by the number of moves required to solve the level. The number of balls has a smaller correlation, followed by the size of the level and the number of unique ball colours. The number of rocks has a small negative correlation.

As with the other two games, the correlations are lower for the solving time aspect. One curious

detail is the fairly low correlation between the number of moves required to solve the level, and the number of moves actually used to solve the level. One would expect these values to have a strong correlation, as they are so closely related. The correlation is, however, not stronger than with the difficulty aspects.

We use R again to perform the linear regression required to set the weights in our equations. At this point, the difficulty measure for the number of counterintuitive moves is not yet used. The resulting weights can be found in Table 19 in Appendix D.3.

We use the resulting weights in our difficulty equation to calculate a difficulty for each level. We then compare these to the ratings given to the levels by the participants in the user study. We do the same for the scaled difficulty, the time to solve the level, and the number of moves. The complete list of these comparisons can be found in Appendix C.3, while an overview of the average errors is presented in Table 4.

	Set	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
Weighted	1	1.0081	1.3938	100.17%	74.56%
	2	1.2078	1.4345	92.24%	63.07%
	3	0.9743	1.3426	87.44%	78.24%
	4	0.7530	1.3549	110.79%	62.51%
	5	1.1318	1.6278	147.98%	112.30%
	Average	1.0212	1.4289	106.72%	77.72%
Unweighted	1	1.0886	1.5517	96.80%	71.86%
	2	1.2334	1.4616	90.79%	62.59%
	3	0.9523	1.3103	83.17%	77.89%
	4	0.7449	1.3708	101.57%	58.66%
	5	1.1067	1.5912	143.86%	109.41%
	Average	1.0252	1.4571	103.13%	76.08%

Table 4: Errors in each of the level sets of Move produced by the difficulty equation when compared to user study data. Shown for difficulty, scaled difficulty, time, and number of moves, both unweighted and weighted by number of level ratings.

The average error between the user-rated difficulty and the calculated difficulty is 1.0252, when weighted this becomes 1.0212. For the scaled difficulty score this error is 1.4571, and with the weights it is 1.4289. The prediction of the time to solve each level is again poor with an error of 103.13%, which increases to 106.72% when the weights are applied. The number of moves has an error of 76.08%, with weights this is 77.72%. The performance of the prediction of the number of moves is quite low. This is related to the fact that the average number of moves is shifted by a few users that need a significantly higher amount of moves to solve a level than other users.

We attempt to increase the accuracy of the difficulty equation by adding the difficulty measure for the amount of counterintuitive moves. The newly calculated weights are presented in Table 20 in Appendix D.3.

We find a slight improvement, as the error in the difficulty is decreased to 0.9456 (weighted: 0.9276). The scaled difficulty error decreases to 1.3399 (weighted: 1.2930). The error for the

	Set	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
Weighted	1	0.8470	1.2006	79.84%	59.95%
	2	1.0413	1.1691	79.28%	54.29%
	3	0.8292	1.1840	82.40%	78.76%
	4	0.8253	1.3959	85.54%	53.12%
	5	1.0939	1.5594	106.84%	75.93%
	Average	0.9276	1.2930	86.34%	64.35%
Unweighted	1	0.9329	1.3613	78.46%	58.68%
	2	1.0632	1.1858	77.37%	54.19%
	3	0.8147	1.1682	78.17%	77.58%
	4	0.8243	1.4304	80.99%	51.76%
	5	1.0930	1.5541	105.24%	75.02%
	Average	0.9456	1.3399	84.05%	63.45%

Table 5: Errors in each of the level sets of Move produced by the improved difficulty equation when compared to user study data. Shown for difficulty, scaled difficulty, time, and number of moves, both unweighted and weighted by number of level ratings.

amount of time is improved to 84.05% (weighted: 86.34%). With the amount of moves the error is decreased to 63.45% (weighted: 64.35%).

The results of Move are fairly similar to those of Lazors. The difference with Flow is quite large, which we assume is again due to the difference in complexity between both games.

6.4 Discussion

An overview of the errors in the difficulty for all three games is given in Figure 11. It can clearly be seen that our difficulty equation produces the best results in Flow. The results for Lazors and Move are similar, with Move producing slightly better results. Note that while Flow produces the best results, it does use only 4 difficulty measures, whereas Lazors and Move use 7 and 6 respectively.

As mentioned before, we assume the differences are mostly caused by the lower complexity of Flow compared to the other two games. Another contributing factor may be the fact that more people participated in the user study for Flow, which may mean that the average difficulties for Flow are more reliable than the other two games. Nevertheless, even the larger errors of Lazors and Move are still good results. An error of one point in the difficulty is in most cases good enough to give players a clear indication of the level of skill required for a level.

We can conclude that our idea of the scaled difficulty scores did not result in the improvement we aimed for, and decreased the accuracy instead. A possible cause for this may be the amount of levels some players solved. If a player only rates a few levels, that player might only be playing harder levels, while our scaling method would still spread the ratings over the entire 1–10 scale. Other methods for normalising the scores may be more effective in improving accuracy. Our difficulty equation does not seem to work as well for predicting the amount of moves players will need to solve a level, and works very poorly to predict the time it will take players on average to solve the

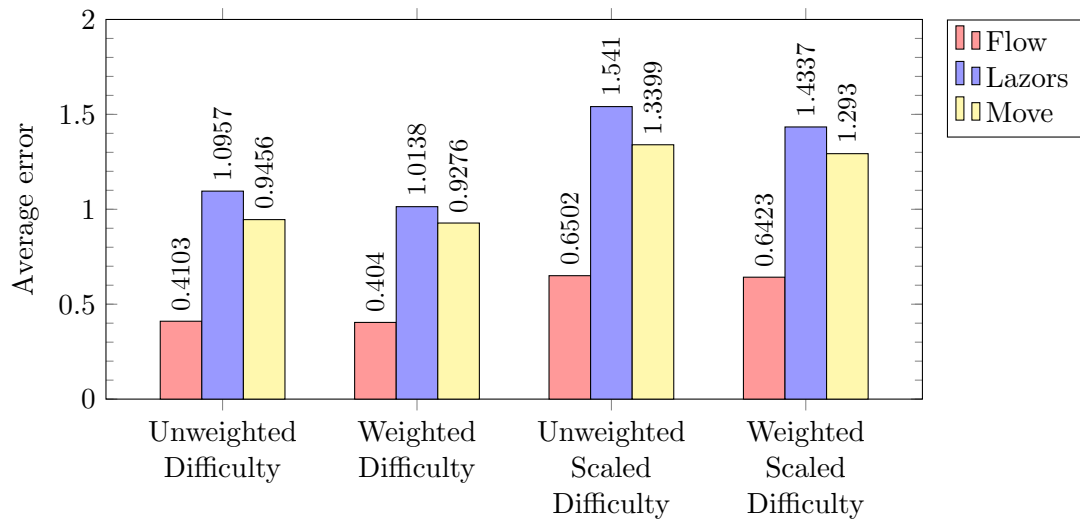


Figure 11: Overview of average errors in calculating the difficulty compared to user-rated difficulty

level. These were, however, not our goals, and we performed these tests only because we had the data anyway.

7 Conclusion and Future Work

In this section we draw a conclusion. We restate our original goals and evaluate to what extent we achieved them. We also suggest possibilities for future work.

7.1 Conclusion

We created a difficulty equation capable of automatically rating the difficulty of puzzle game levels. We performed a user study to gather data on the difficulty of levels in three different games: Flow, Lazors, and Move. We used this data to set the weights in our difficulty equation in order to match the user-provided difficulty ratings. We then compared the results from the difficulty equation to the ratings given to the levels by the users, in order to evaluate the accuracy of the equation.

Our first goal was to create an effective equation, capable of accurately predicting the difficulty of levels. With an average error of approximately one point for Lazors and Move, and less than half of a point for Flow, we consider that we achieved this goal.

Our second goal was to have this difficulty equation be easily applicable to most puzzle games, and still provide good results. We have shown the process for choosing the variables in the equation to make it work on any puzzle game. We consider this to be an easy process, as long as there is some familiarity with the game. We tested the difficulty equation on three very different games, and in all three cases our method was capable of accurately predicting the difficulty rating of levels, showing that our method adapts well to different games.

We also applied the method of the difficulty equation to calculate other aspects of the levels. These aspects include the time taken to solve a level, and the amount of moves used to solve a level. The equation had a lower performance in these cases, which is to be expected, as the used measures were chosen for their relation with difficulty, and not these aspects. As these aspects were not part of our goals, we do not consider these results in our overall judgement.

Considering our original goals, we are satisfied with our results. We have achieved our goals, and have done so with great accuracy.

7.2 Future Work

While our results are satisfactory, improvements are still possible. In this section we will highlight some possible future extensions and improvements to our work.

We only tested a linear model for our difficulty equation. This may not be the best model for determining the difficulty. Other possible models include quadratic models, polynomial models, or even models combining the different difficulty measures. The use of a different model can cause great differences, both positively and negatively. Further research is necessary to make a judgement on this.

The results for the difficulty equation may be used to enhance level creation. With the weights

of the difficulty measures, and the correlation factors, it is possible to find the requirements for creating levels at a specified difficulty. This can be used to create an automated level generator, that creates levels at the player's skill level. This can greatly increase the enjoyment of the players, as they are constantly challenged.

Our results show that the selection of proper difficulty measures is very important for the performance of the difficulty equation. We select these difficulty measures by hand, after playing the game for a while. We may not have picked the right measures, and this could possibly be improved by an automated selection of the difficulty measures. The user study data could be used directly to find patterns between the levels and their assigned difficulty ratings. This is obviously not a trivial task, but future work might show possibilities to do this.

References

- [1] Petr Jarušek and Radek Pelánek. Difficulty rating of sokoban puzzle. In *Proceedings of the 2010 Conference on STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*, pages 140–150, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.
- [2] D. Ashlock and J. Schonfeld. Evolution for automatic assessment of the difficulty of sokoban boards. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, July 2010.
- [3] Cameron Browne. *Evolutionary game design*. Springer, 2011.
- [4] T. Mantere and J. Koljonen. Solving, rating and generating sudoku puzzles with ga. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1382–1389, Sept 2007.
- [5] Maria-Virginia Aponte, Guillaume Levieux, and Stephane Natkin. Measuring the level of difficulty in single player video games. *Entertainment Computing*, 2(4):205–213, 2011.
- [6] Szilárd András, Kinga Sipos, and Anna Sóos. Which is harder?-classification of happy cube puzzles. 2013.
- [7] Leonid Vitalievich Kantorovich. A new method of solving of some classes of extremal problems. In *Dokl. Akad. Nauk SSSR*, volume 28, pages 211–214, 1940.
- [8] Nicky Vendrig. Automated level generation and difficulty rating for Trainyard. 2013.
- [9] N.R. Draper and H. Smith. *Applied Regression Analysis*. Number dl. 766 in Applied Regression Analysis. Wiley, 1981.
- [10] Big Duck Games. Flow. <http://blog.bigduckgames.com/>.
- [11] Nitako Brain Puzzles. Move. <http://www.nitako.com/>.
- [12] Pyrosphere. Lazors. <http://pyrosphere.net/lazors/>.
- [13] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

List of Figures

1	Level example, Flow	8
2	Level example, Move	9
3	Level example, Lazors	10
4	Move level generation flowchart	12
5	Complexity distribution of generated Move levels	13
6	Age distribution of user study participants	18
7	Education distribution of user study participants	19
8	Correlation coefficients, Flow	21
9	Correlation coefficients, Lazors	23
10	Correlation coefficients, Move	26
11	Average difficulty equation errors overview	29
12	List of Flow levels (1-20)	35
13	List of Flow levels (21-40)	36
14	List of Lazors levels (1-20), initial configuration	37
15	List of Lazors levels (21-45), initial configuration	38
16	List of Lazors levels (46-65), initial configuration	39
17	List of Lazors levels (1-20), solved	40
18	List of Lazors levels (21-45), solved	41
19	List of Lazors levels (46-65), solved	42
20	List of Move levels (1-20), initial configuration	43
21	List of Move levels (21-40), initial configuration	44
22	List of Move levels (41-60), initial configuration	45
23	List of Move levels (61-80), initial configuration	46

List of Tables

1	Difficulty equation errors, Flow	22
2	Difficulty equation errors, Lazors	24
3	Improved difficulty equation errors, Lazors	25
4	Difficulty equation errors, Move	27
5	Improved difficulty equation errors, Move	28
6	Difficulty measure values, Flow	47
7	Difficulty measure values, Lazors (1-40)	48
8	Difficulty measure values, Lazors (41-65)	49
9	Difficulty measure values, Move (1-40)	50
10	Difficulty measure values, Move (41-80)	51
11	User study data compared to difficulty equation results, Flow	52
12	User study data compared to difficulty equation results, Lazors (1-40)	53
13	User study data compared to difficulty equation results, Lazors (41-65)	54
14	User study data compared to difficulty equation results, Move (1-40)	55
15	User study data compared to difficulty equation results, Move (41-80)	56
16	Difficulty equation weights, Flow	57
17	Difficulty equation weights, Lazors	58
18	Adjusted difficulty equation weights, Lazors	59
19	Difficulty equation weights, Move	60
20	Adjusted difficulty equation weights, Move	61

A List of All Levels

A.1 Flow

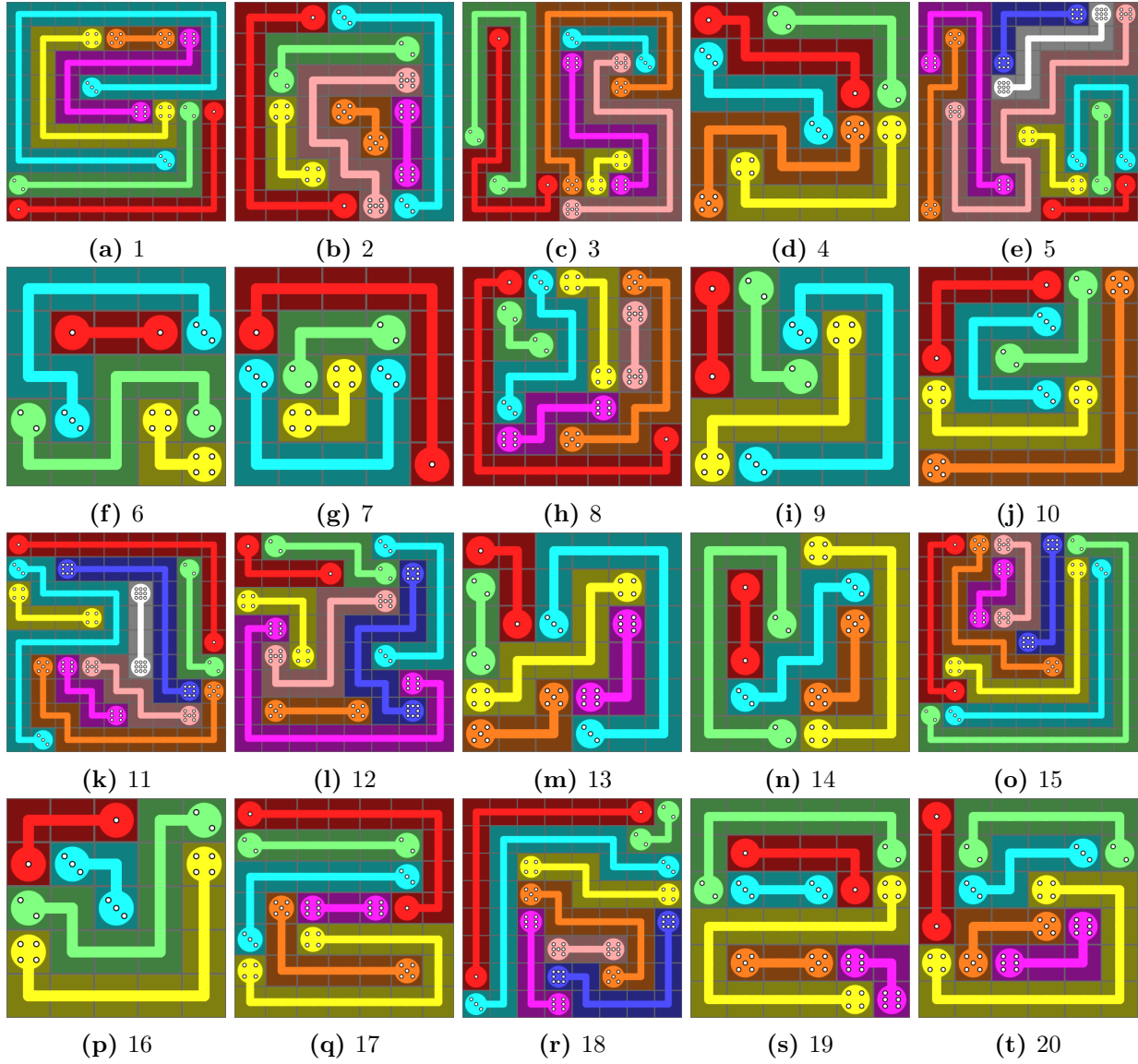


Figure 12: List of levels used for Flow, levels 1-20, shown solved

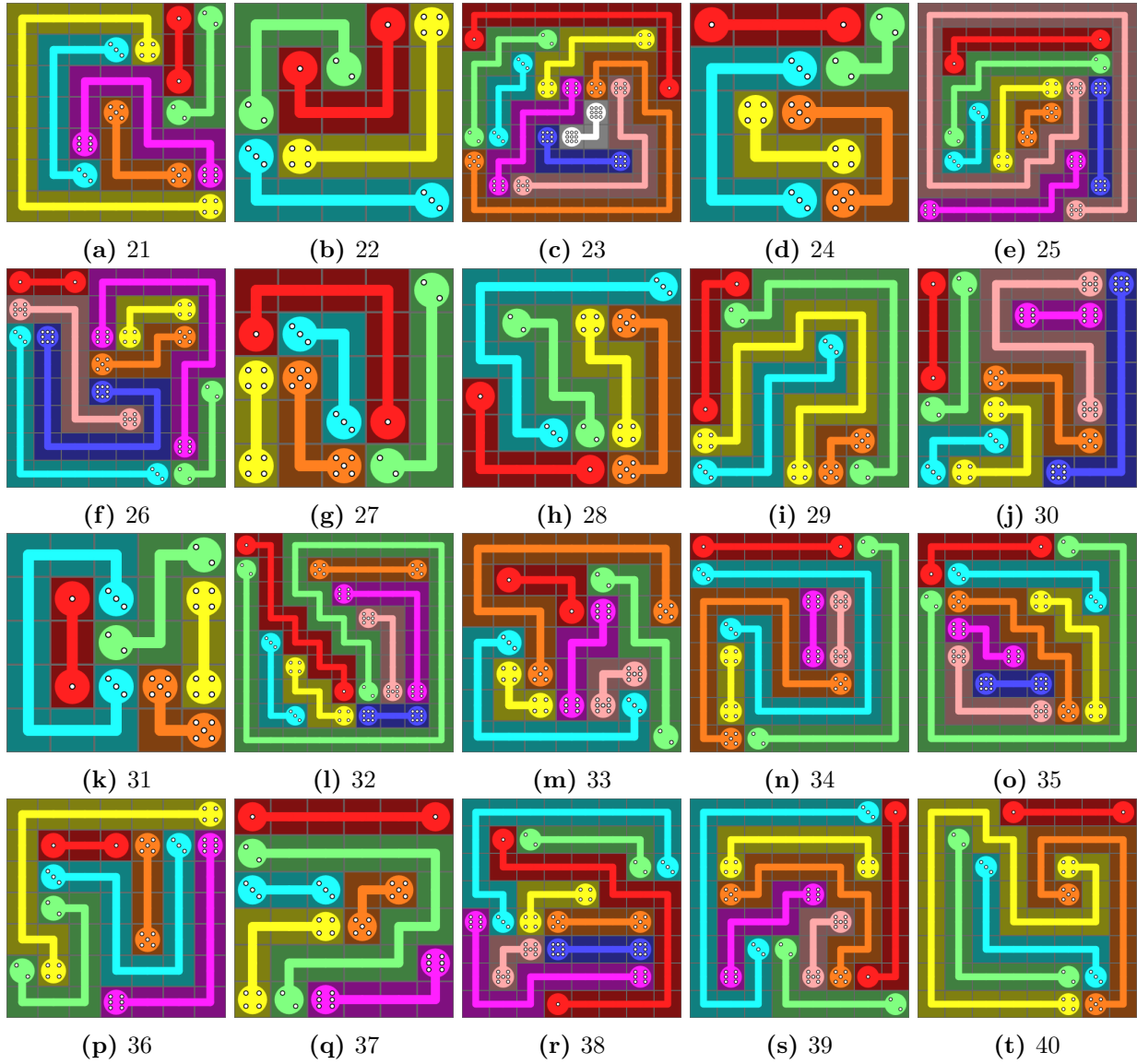


Figure 13: List of levels used for Flow, levels 21-40, shown solved

A.2 Lazors

A.2.1 Initial Configuration

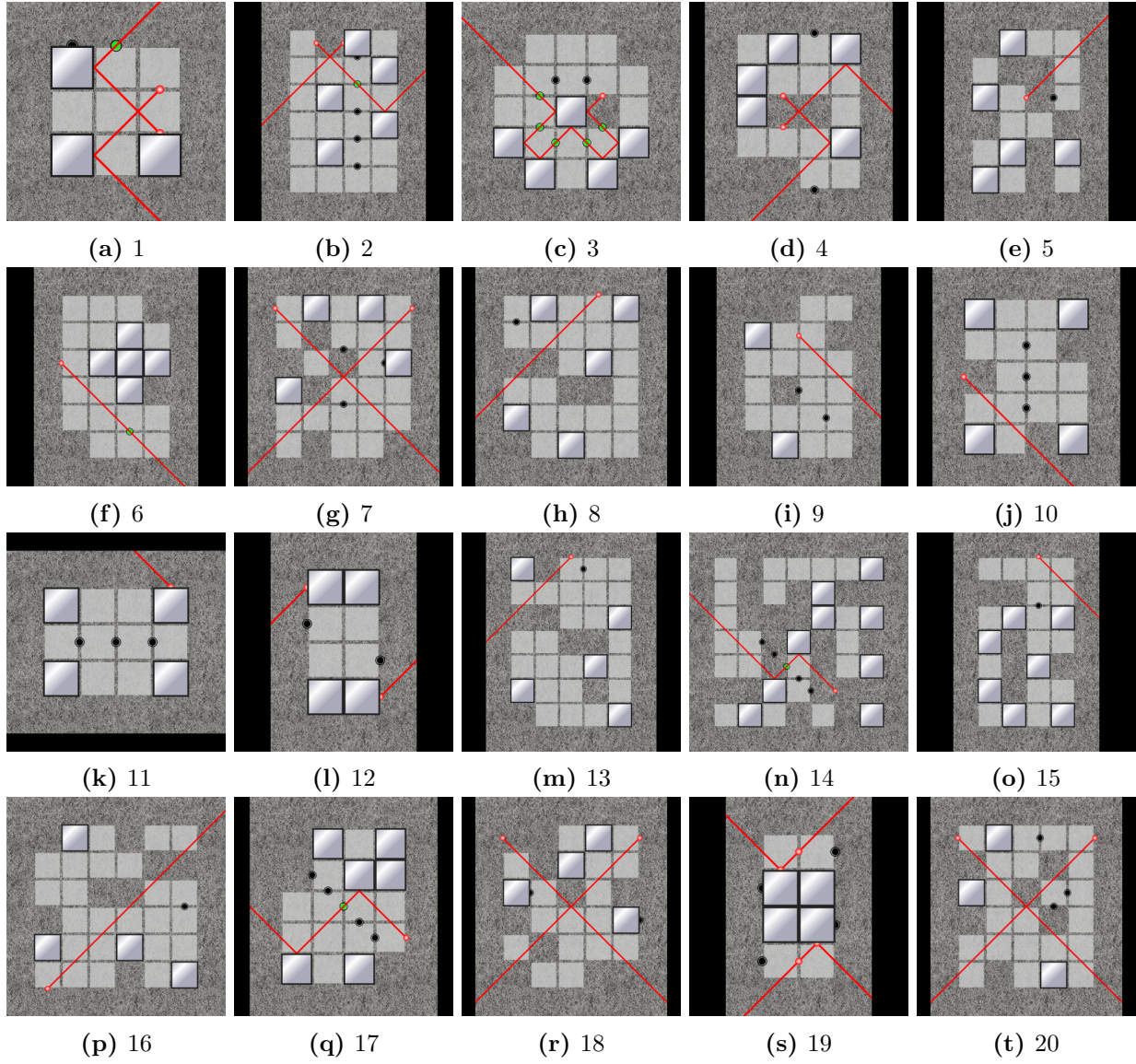


Figure 14: List of levels used for Lazors, levels 1-20, shown in initial configuration

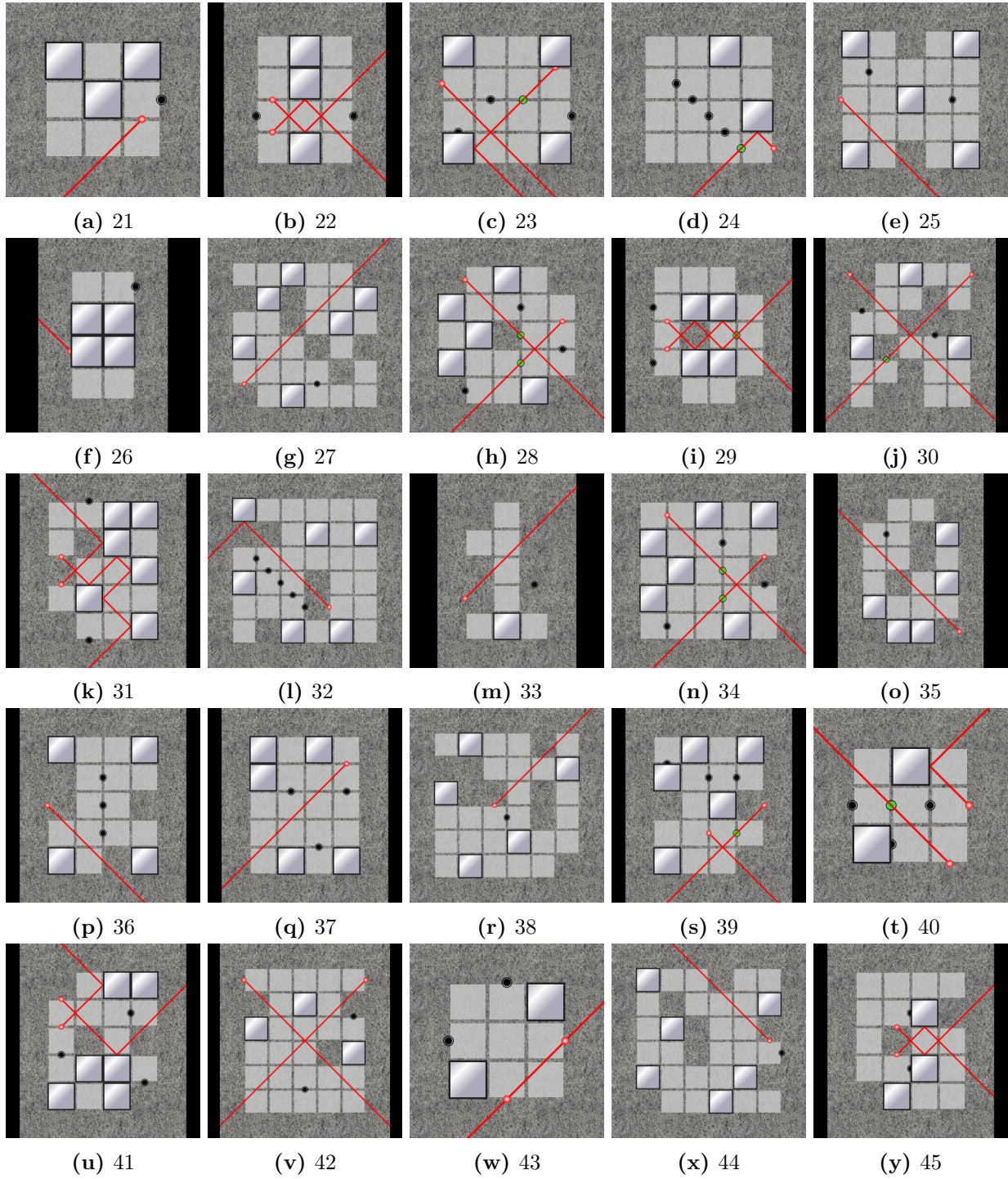


Figure 15: List of levels used for Lazors, levels 21-45, shown in initial configuration

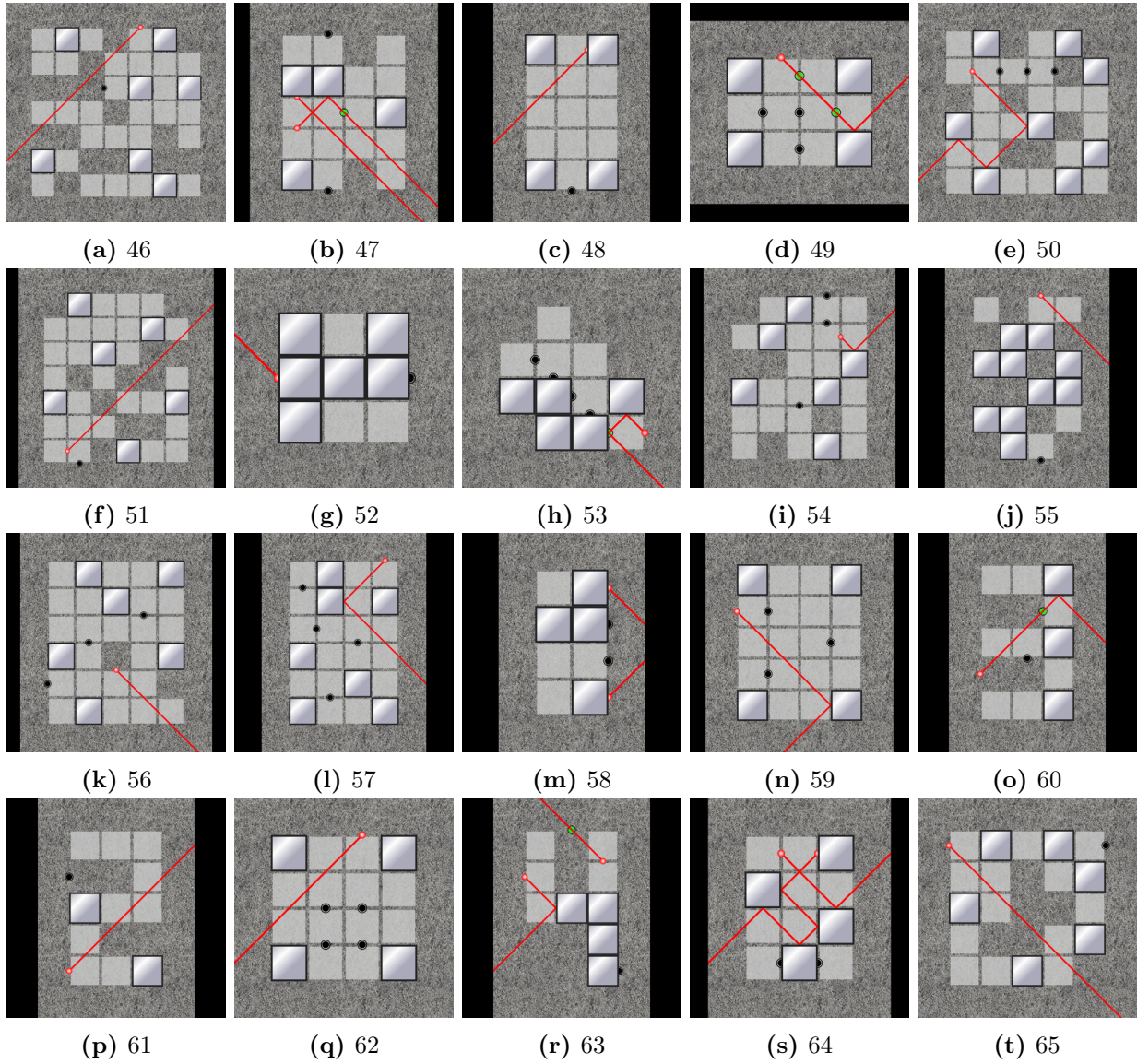


Figure 16: List of levels used for Lazors, levels 46-65, shown in initial configuration

A.2.2 Solved

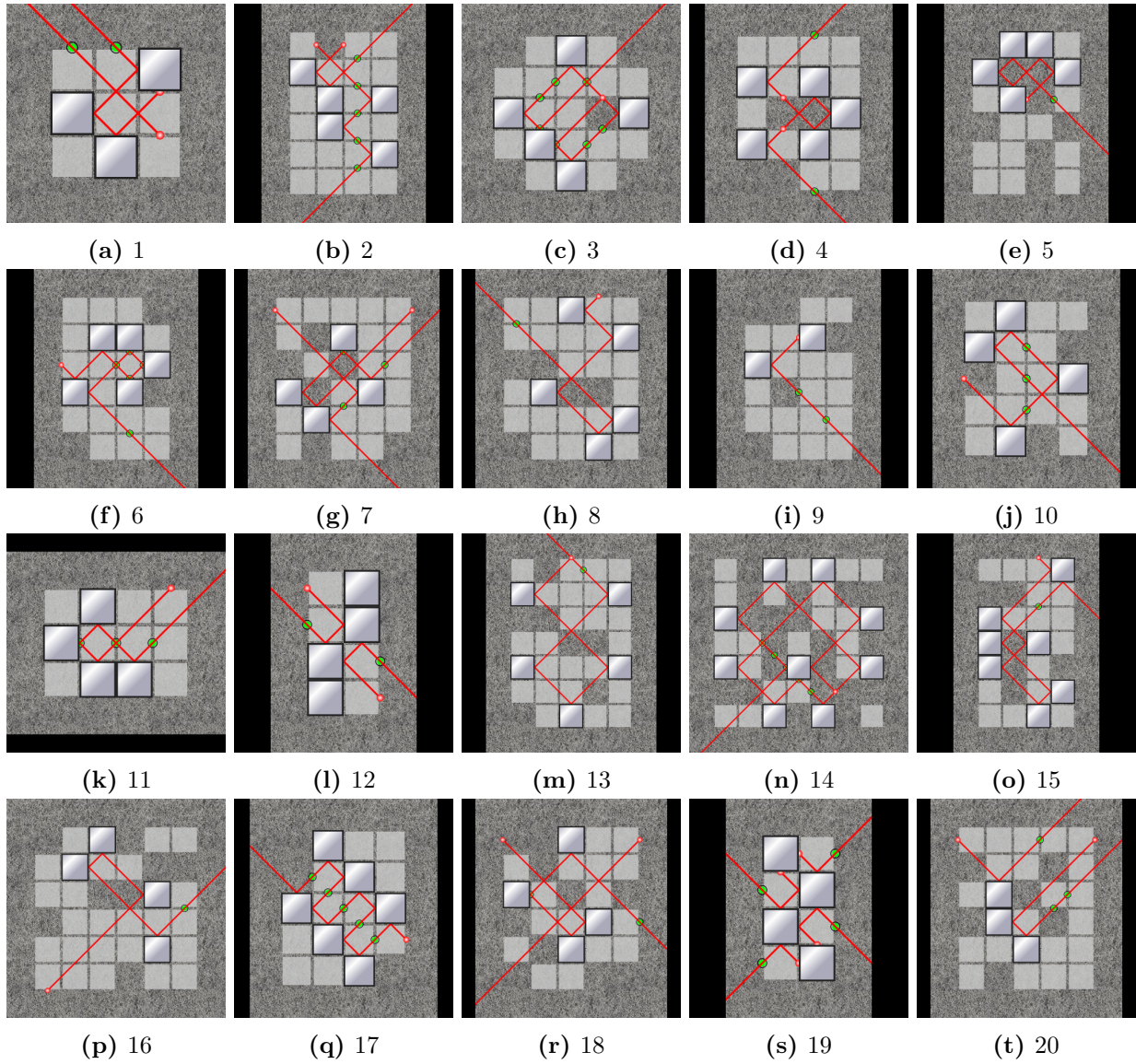


Figure 17: List of levels used for Lazors, levels 1-20, shown solved

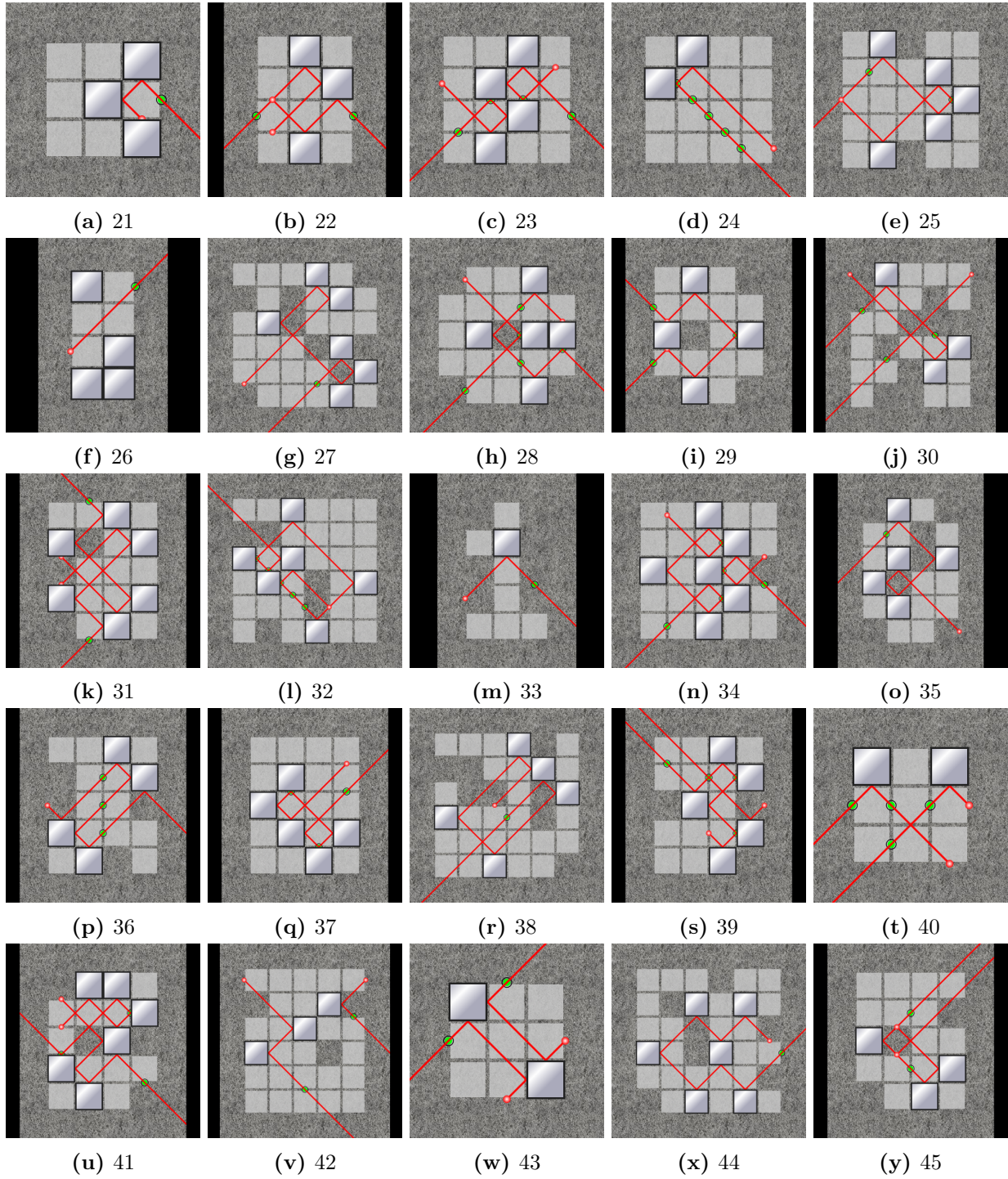


Figure 18: List of levels used for Lazors, levels 21-45, shown solved

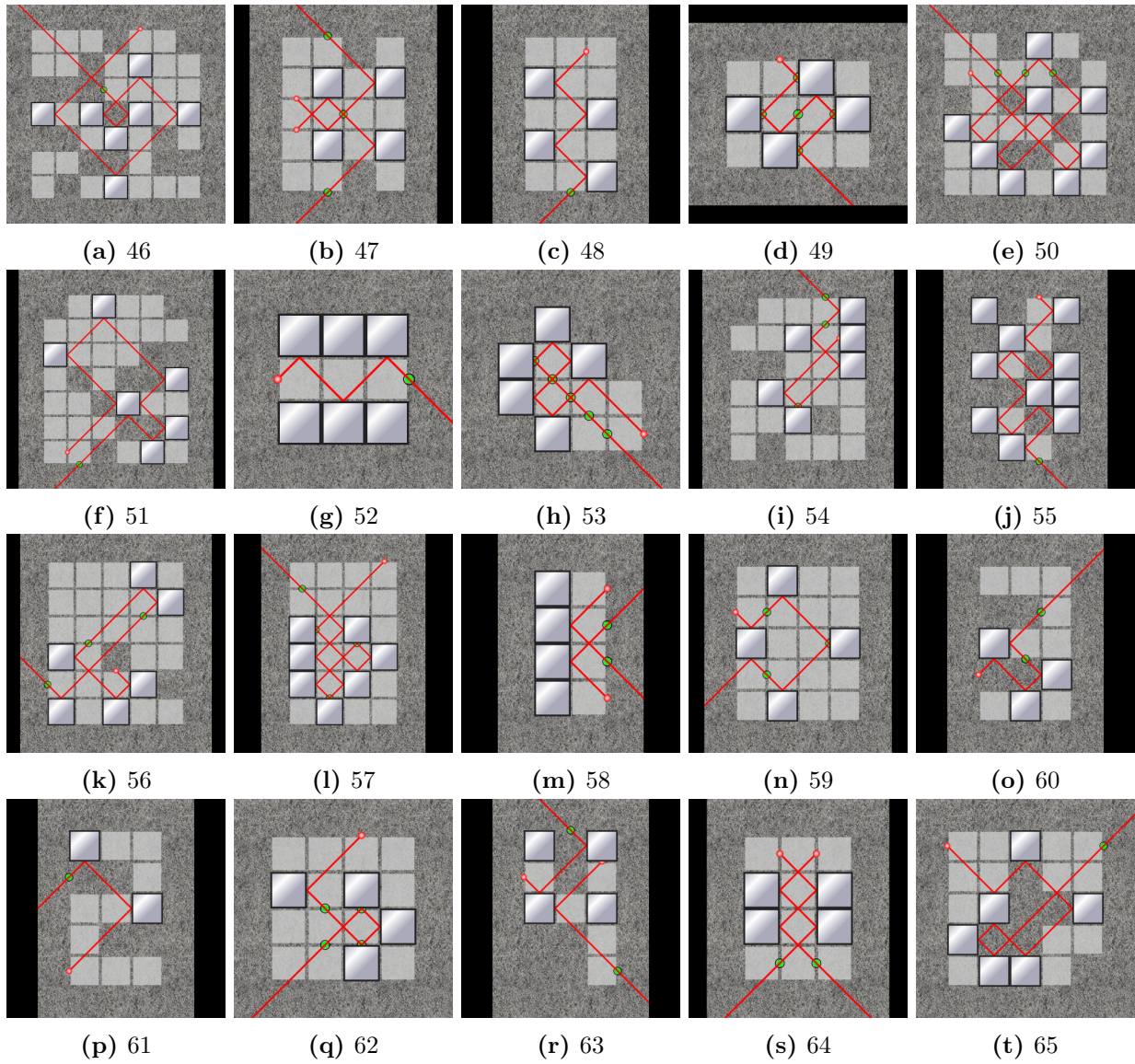


Figure 19: List of levels used for Lazors, levels 46-65, shown solved

A.3 Move

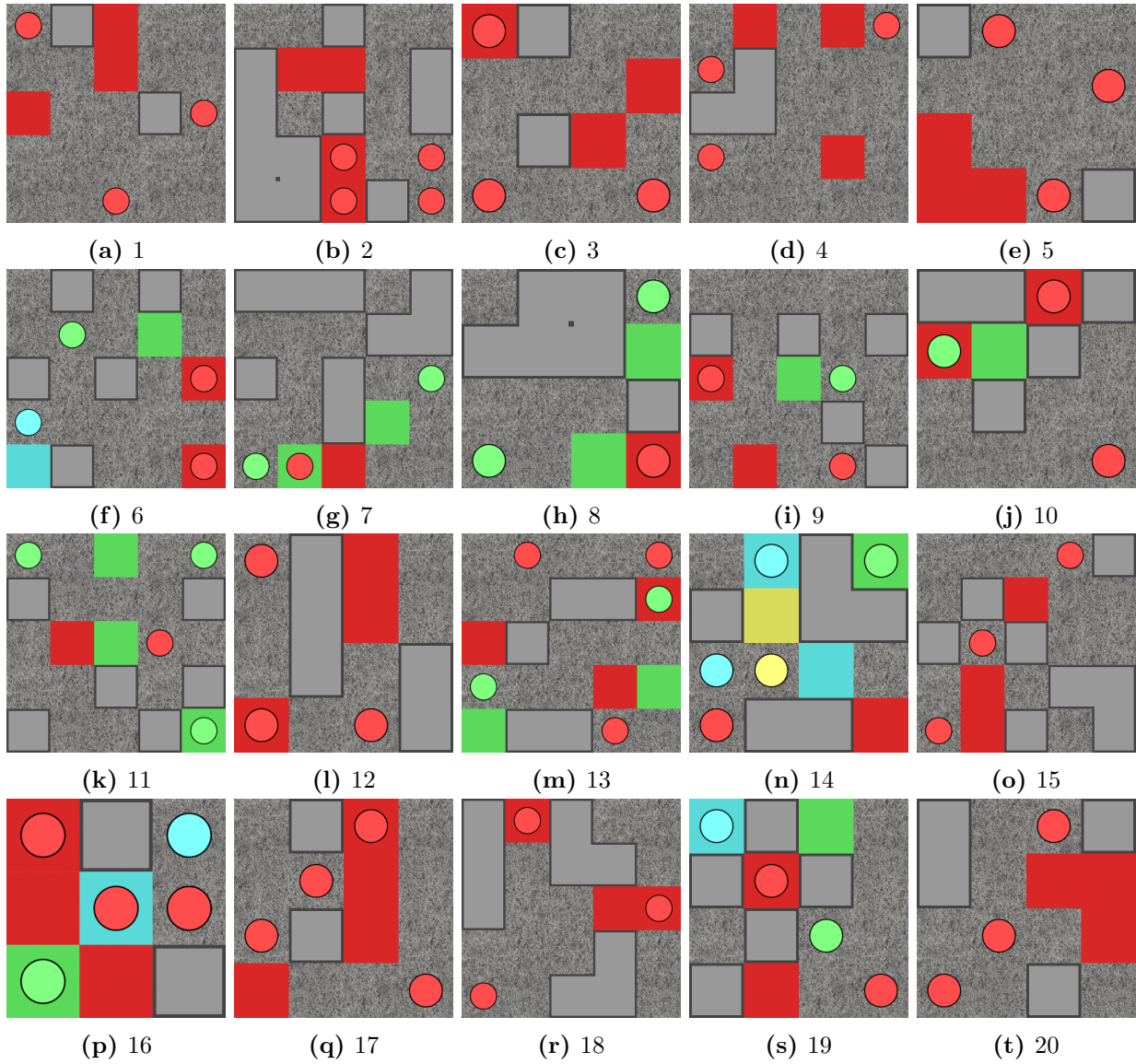


Figure 20: List of levels used for Move, levels 1-20, shown in initial configuration

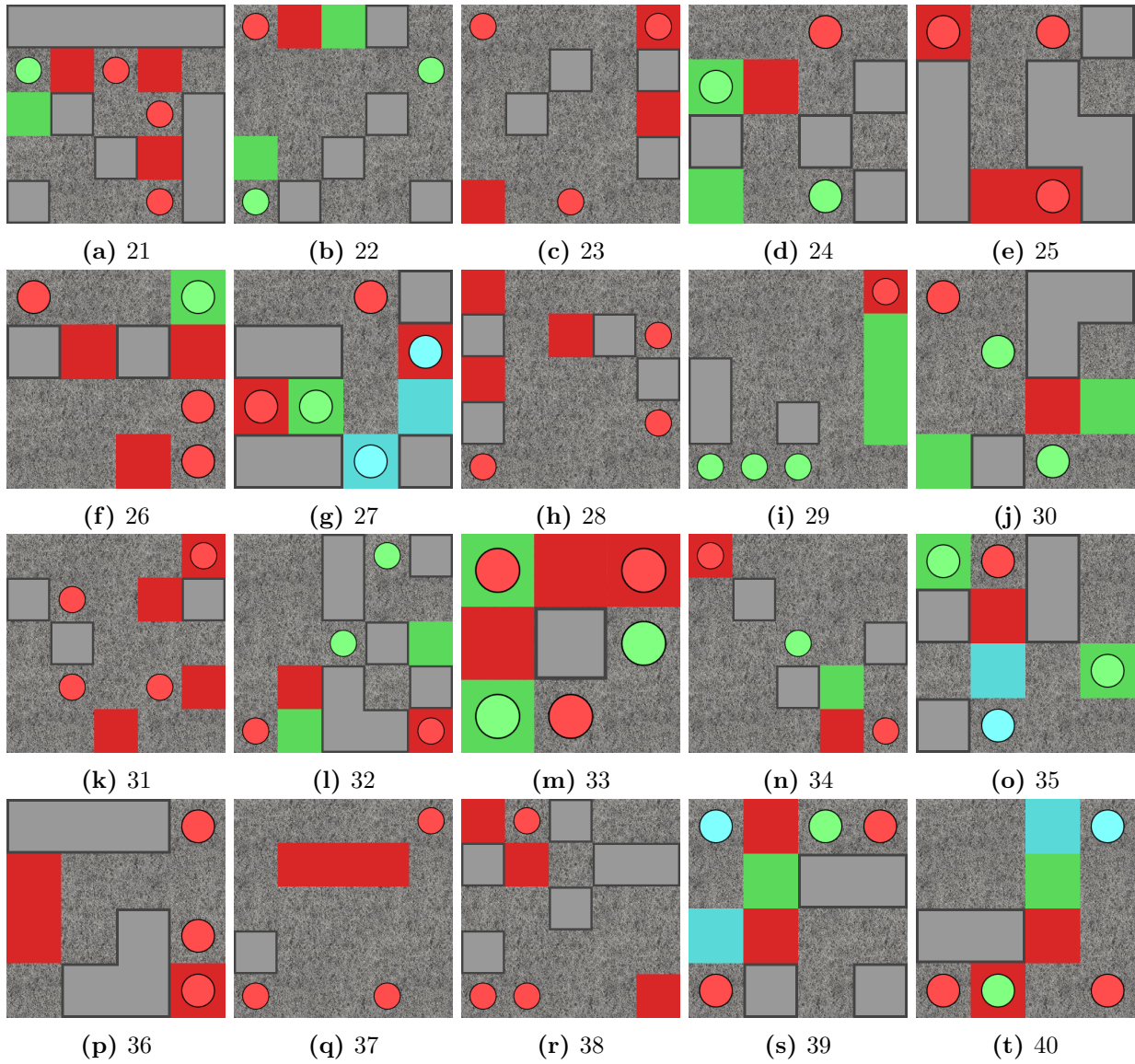


Figure 21: List of levels used for Move, levels 21-40, shown in initial configuration

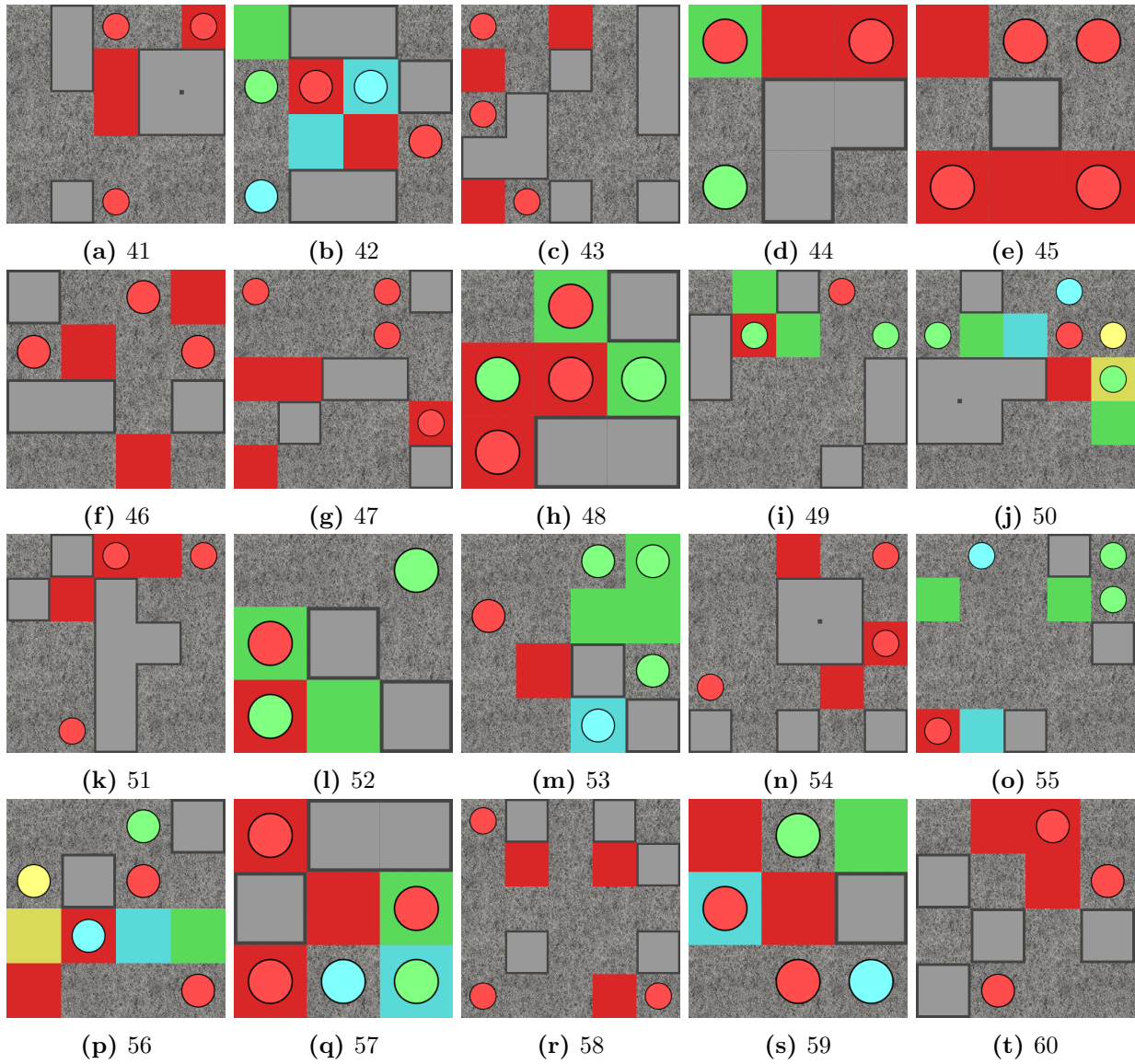


Figure 22: List of levels used for Move, levels 41-60, shown in initial configuration

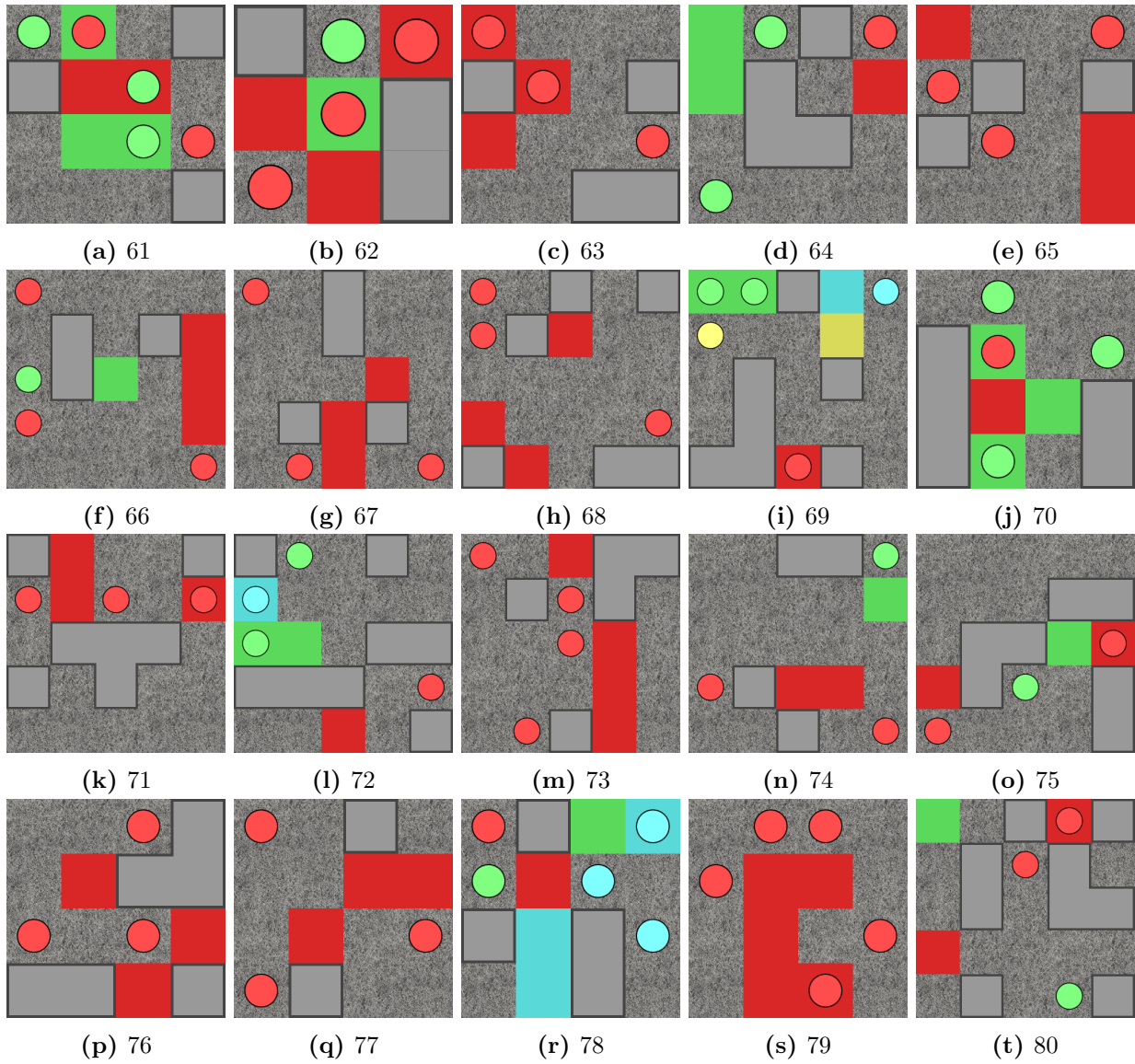


Figure 23: List of levels used for Move, levels 61-80, shown in initial configuration

B Level Properties

B.1 Flow

Level	Difficulty Measures			
	Size	Corners	Colours	Distance
1	9 × 9	12	6	6.833
2	7 × 7	12	7	4.571
3	9 × 9	21	7	5.286
4	6 × 6	12	5	5.400
5	9 × 9	22	9	5.889
6	5 × 5	9	4	2.750
7	5 × 5	6	4	3.750
8	7 × 7	15	7	4.857
9	5 × 5	6	4	3.750
10	6 × 6	7	5	5.000
11	9 × 9	22	9	6.444
12	8 × 8	18	8	4.750
13	6 × 6	10	6	3.667
14	6 × 6	10	5	3.800
15	9 × 9	19	8	7.375
16	5 × 5	8	4	4.250
17	7 × 7	7	6	5.167
18	8 × 8	16	8	6.250
19	6 × 6	7	6	3.333
20	6 × 6	9	6	3.667
21	7 × 7	11	6	4.500
22	5 × 5	6	4	4.250
23	9 × 9	25	9	6.222
24	5 × 5	6	5	2.800
25	9 × 9	19	8	5.375
26	8 × 8	14	8	5.250
27	5 × 5	5	5	3.600
28	6 × 6	12	5	5.000
29	7 × 7	16	5	5.600
30	7 × 7	10	8	4.125
31	5 × 5	7	5	2.400
32	9 × 9	23	8	5.625
33	7 × 7	15	7	4.286
34	8 × 8	10	7	4.429
35	8 × 8	16	8	5.250
36	7 × 7	10	6	5.167
37	6 × 6	8	6	3.667
38	8 × 8	15	8	5.000
39	8 × 8	18	7	6.143
40	8 × 8	15	5	6.200

Table 6: Values of difficulty measures for Flow levels

Appendix

B.2 Lazors

Level	Difficulty Measures						
	Size	Usable Tiles	Emitters	Receivers	Mirrors	Reflections	Intersections
1	3 × 3	9	2	2	3	3	2
2	4 × 6	23	2	5	5	5	2
3	5 × 5	20	1	7	5	6	2
4	4 × 5	16	2	2	5	8	1
5	4 × 6	16	1	1	5	5	2
6	4 × 6	20	1	5	5	6	1
7	5 × 6	24	2	3	4	6	3
8	5 × 6	25	1	1	5	5	1
9	4 × 6	19	1	2	2	2	0
10	4 × 5	17	1	3	4	4	1
11	4 × 3	12	1	3	4	4	1
12	2 × 4	8	2	2	4	4	0
13	5 × 7	28	1	1	5	5	2
14	7 × 7	29	1	5	9	11	5
15	4 × 7	22	1	1	7	7	2
16	6 × 6	28	1	1	4	4	1
17	4 × 5	17	1	5	6	10	0
18	5 × 6	20	2	2	4	4	4
19	2 × 4	8	4	4	4	10	0
20	5 × 6	24	2	3	3	3	0
21	3 × 3	9	1	1	3	3	0
22	3 × 4	12	2	2	3	4	2
23	4 × 4	16	2	4	4	6	2
24	4 × 4	16	1	5	2	2	0
25	5 × 5	23	1	2	5	5	2
26	2 × 4	8	1	1	4	0	0
27	6 × 6	28	1	1	6	6	2
28	5 × 5	20	2	5	5	6	2
29	4 × 5	16	2	3	4	10	0
30	5 × 6	20	2	3	3	3	3
31	4 × 5	17	2	2	6	8	4
32	6 × 6	31	1	5	6	8	2
33	3 × 5	8	1	1	1	1	0
34	5 × 5	24	2	5	6	10	3
35	4 × 6	17	1	1	5	5	1
36	4 × 5	16	1	3	4	6	0
37	4 × 5	20	1	3	5	6	2
38	6 × 6	28	1	1	5	6	1
39	4 × 5	17	2	4	5	7	2
40	3 × 3	9	2	4	2	2	1

Table 7: Values of difficulty measures for Lazors levels (levels 1-40)

Appendix

Level	Difficulty Measures						
	Size	Usable Tiles	Emitters	Receivers	Mirrors	Reflections	Intersections
41	4×5	16	2	3	6	9	3
42	5×6	27	2	2	3	3	0
43	3×3	9	2	2	2	4	0
44	6×6	29	1	1	6	7	0
45	4×5	17	2	2	3	3	3
46	7×7	34	1	1	7	7	2
47	4×5	18	2	3	4	4	2
48	3×5	15	1	1	4	4	0
49	4×3	12	1	5	4	6	0
50	6×6	28	1	3	8	10	6
51	6×7	31	1	1	6	8	0
52	3×3	9	1	1	6	3	0
53	4×4	11	1	5	5	6	2
54	5×6	25	1	3	6	6	1
55	4×6	15	1	1	10	12	0
56	5×6	28	1	3	6	6	1
57	4×6	24	1	4	7	9	4
58	2×4	8	2	2	4	2	1
59	4×5	20	1	3	4	5	0
60	3×5	11	1	2	3	4	0
61	3×5	11	1	1	2	2	0
62	4×4	16	1	4	4	4	1
63	3×5	9	2	2	4	4	0
64	3×4	12	2	2	4	4	3
65	5×5	19	1	1	6	7	2

Table 8: Values of difficulty measures for Lazors levels (levels 41-64)

B.3 Move

Level	Difficulty Measures						Solution
	Size	Balls	Colours	Rocks	Moves	Counterintuitive Moves	
1	5 × 5	3	1	2	9	2	↑↑↑→→←↓↓←
2	5 × 5	4	1	11	7	2	←↑↑←↑↑←
3	4 × 4	3	1	2	4	0	→↑→↑
4	5 × 5	3	1	3	7	2	→→↓↑→→←
5	4 × 4	3	1	2	4	0	←↓←↓
6	5 × 5	4	3	5	6	2	↑←↓→→→
7	5 × 5	3	2	9	4	1	↓→→←
8	4 × 4	3	2	6	3	0	→→↓
9	5 × 5	3	2	5	3	1	→←←
10	4 × 4	3	2	5	6	0	←←←↑→↑
11	5 × 5	4	2	6	9	3	←←←↓↓↓→↑→
12	4 × 4	3	1	5	10	3	→↑→↑↑→↓↓↓←
13	5 × 5	5	2	5	7	2	↑←↓↓→↑↓
14	4 × 4	5	4	6	10	3	←↑↑→↑→→→↓→
15	5 × 5	3	1	8	4	1	←←→↓
16	3 × 3	5	3	2	3	0	←↓←
17	4 × 4	4	1	2	5	1	→↓→↑←
18	5 × 5	3	1	9	7	2	→↓↑→↓↑→
19	4 × 4	4	3	5	7	1	→↑←↑←↓←
20	4 × 4	3	1	4	6	0	→↑→↓→↑
21	5 × 5	4	2	11	8	3	↑←↓←↑→↓↑
22	5 × 5	3	2	5	5	1	←←←→↑
23	5 × 5	3	1	4	10	1	↓↓↓↓↑↑→→↓↓
24	4 × 4	3	2	4	3	0	←↓←
25	4 × 4	3	1	8	6	2	→↓↓↓←←
26	4 × 4	4	2	2	8	4	↑←←←↑→↓→
27	4 × 4	5	3	6	5	2	↑↑↓→↓
28	5 × 5	3	1	4	10	1	←←↑←←→↑↑←↓
29	5 × 5	4	2	3	8	1	→↑↑→↓→→↑
30	4 × 4	3	2	4	7	2	↓↓↑←↓→→
31	5 × 5	4	1	3	5	2	←→↑→↓
32	5 × 5	4	2	8	7	2	↑↑↑←↓→↓
33	3 × 3	5	2	1	9	2	↑←→↓→↑←↑←
34	5 × 5	3	2	3	3	0	→↓←
35	4 × 4	4	3	4	3	1	↓↓↑
36	4 × 4	3	1	6	7	1	↓←↑←↓←←
37	5 × 5	3	1	1	8	1	←→↑←↑↑↑↓
38	5 × 5	3	1	6	9	1	→↑←↑↑↑→→↓
39	4 × 4	4	3	4	8	2	←↑↑↓←←→↓
40	4 × 4	4	3	2	6	2	→↑→↓←↑

Table 9: Values of difficulty measures for Move levels (levels 1-40)

Appendix

Level	Difficulty Measures						Solution
	Size	Balls	Colours	Rocks	Moves	Counterintuitive Moves	
41	5 × 5	3	1	7	4	0	↑↑↑↓
42	4 × 4	5	3	5	4	2	↑←←→
43	5 × 5	3	1	9	5	1	→→→←↑
44	3 × 3	3	2	3	3	0	↑→↑
45	3 × 3	4	1	1	5	1	↓→←↓←
46	4 × 4	3	1	4	6	1	↓↓↓←↑→
47	5 × 5	4	1	5	9	1	→↓←←↑↓←←↓
48	3 × 3	5	2	3	8	1	←↑→↓←↑→↓
49	5 × 5	3	2	6	7	1	↑↓←↑↓←↑
50	5 × 5	5	4	6	7	2	↓←↑↑↓→↓
51	5 × 5	3	1	7	4	0	↑↑↑←
52	3 × 3	3	2	2	6	0	←←→↓←↓
53	4 × 4	5	3	2	5	2	↓↑←↓→
54	5 × 5	3	1	7	8	1	→→→→↑←↓←
55	5 × 5	4	3	3	10	2	↓↓↓↓←←←↑↓←
56	4 × 4	5	4	2	10	2	↓→↑↑←←↓↑↑←
57	3 × 3	5	3	3	9	3	↑←↓→↑←↓→↑
58	5 × 5	3	1	5	10	3	→→↑↑↑↑←←↓→
59	3 × 3	4	3	1	9	3	→←↑→↑←↓↓↑
60	4 × 4	3	1	4	5	0	→↑↑←↑
61	4 × 4	5	2	3	8	2	←↑↑↓→→↑←
62	3 × 3	4	2	3	3	0	←↓→
63	4 × 4	3	1	4	3	0	←←←
64	4 × 4	3	2	4	6	2	↓↓↓↑←↑
65	4 × 4	3	1	3	6	0	←↓→↓→↑
66	5 × 5	4	2	3	9	2	→↓→↑→↑↑→↓
67	5 × 5	3	1	4	9	1	→↓↓→↓←←↑→
68	5 × 5	3	1	6	10	2	↓↓→↑↑→←↑↓←
69	5 × 5	5	4	7	5	1	→→→→→←
70	4 × 4	4	2	5	3	1	→←↓
71	5 × 5	3	1	7	8	1	→↓→↑←←←↑
72	5 × 5	4	3	8	4	0	←↓←↓
73	5 × 5	4	1	5	9	4	↑→↓→↓→→←↑
74	5 × 5	3	2	4	8	1	↓↑↑→→→←↓
75	5 × 5	3	2	7	4	0	→↑←↑
76	4 × 4	3	1	6	6	1	←←→↓→→
77	4 × 4	3	1	2	5	1	→↑↑↓→
78	4 × 4	5	3	4	10	0	←↓←↓→↓→↑↑↑
79	4 × 4	5	1	0	4	1	↓←←→
80	5 × 5	3	2	9	10	1	←↑←↓↓←↑↑↑←

Table 10: Values of difficulty measures for Move levels (levels 41-80)

C User Study Data

C.1 Flow

Level	Plays		Difficulty		Scaled Difficulty		Time (s)		Moves	
	Solved	Skipped	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.
1	29	1	5.00	5.19	6.31	7.26	53.01	56.04	15.31	11.91
2	30	0	2.37	2.91	2.75	3.73	24.50	23.82	7.17	8.82
3	27	2	5.96	5.77	8.48	8.12	74.09	60.70	18.52	15.04
4	30	0	2.57	1.99	3.04	2.38	14.78	15.03	5.97	5.17
5	26	1	5.77	4.89	7.66	6.84	58.05	42.10	17.69	14.75
6	32	0	2.06	2.39	2.45	2.82	10.15	27.11	4.06	5.71
7	34	0	1.47	1.88	1.33	2.08	7.01	19.63	3.65	4.15
8	35	0	2.91	2.86	3.68	3.69	54.07	21.86	9.06	8.88
9	36	0	1.56	1.77	1.56	1.96	83.10	9.26	4.08	3.88
10	29	0	2.52	2.13	2.83	2.48	12.22	13.07	5.31	5.20
11	36	0	4.44	4.91	6.44	6.75	40.08	50.58	14.64	14.79
12	32	1	4.88	4.06	6.68	5.47	39.63	41.14	13.53	12.30
13	37	0	2.57	2.28	3.07	2.76	16.96	17.90	7.08	6.51
14	35	0	2.54	2.53	3.22	3.13	16.23	19.94	6.74	6.54
15	31	0	3.35	4.86	4.64	6.63	28.06	47.65	10.35	13.88
16	32	0	1.50	1.70	1.43	1.90	9.29	7.71	3.88	3.50
17	34	0	2.94	2.68	3.93	3.18	20.83	29.20	7.59	7.84
18	33	0	3.39	3.43	4.42	4.57	25.14	28.51	9.30	10.47
19	34	0	2.03	2.38	2.31	2.81	11.56	26.41	5.76	7.09
20	30	0	2.27	2.35	2.30	2.82	13.15	23.70	6.30	6.80
21	35	1	3.66	3.19	4.67	4.07	26.38	33.22	9.46	8.97
22	35	0	1.91	1.53	2.14	1.52	9.55	16.39	4.03	3.39
23	32	0	4.47	5.12	6.10	7.29	50.46	42.93	14.94	15.09
24	31	0	1.58	1.90	1.82	2.09	8.33	21.84	4.55	5.17
25	28	3	5.64	5.45	7.75	7.55	50.43	56.82	12.93	16.62
26	30	2	4.47	3.70	6.35	4.90	44.53	32.76	13.60	11.27
27	34	0	1.29	1.52	1.22	1.53	13.30	11.43	4.50	3.53
28	35	0	2.23	2.21	2.64	2.64	14.71	19.26	5.31	5.25
29	35	0	3.46	3.31	4.81	4.31	29.77	33.63	8.31	8.48
30	30	0	2.63	2.68	3.22	3.36	21.21	20.29	9.07	8.41
31	30	0	1.57	2.23	1.59	2.62	9.45	23.07	4.70	5.98
32	28	0	4.57	5.55	6.38	7.76	45.50	58.14	14.21	16.82
33	33	1	4.21	3.10	5.99	4.04	36.30	28.71	12.06	9.15
34	28	1	4.64	4.17	6.10	5.45	63.63	39.64	13.21	12.30
35	35	0	2.83	3.79	3.53	5.06	22.58	34.31	10.23	11.52
36	33	0	2.85	2.89	4.08	3.59	26.96	26.19	7.88	8.04
37	31	0	2.19	2.26	2.70	2.65	12.94	19.13	6.35	6.43
38	36	0	4.22	3.85	5.78	5.12	34.30	34.86	11.69	11.74
39	32	1	4.31	3.80	5.69	5.11	37.59	36.71	10.56	10.91
40	31	1	4.42	4.22	6.07	5.63	33.55	45.22	10.23	11.16

Table 11: User study data compared to difficulty equation results for Flow levels

Appendix

C.2 Lazors

Level	Plays		Difficulty		Scaled Difficulty		Time (s)		Moves	
	Solved	Skipped	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.
1	10	0	3.50	3.26	2.68	3.77	17.50	32.33	7.30	11.29
2	10	0	3.70	5.67	4.00	6.22	45.46	66.44	17.30	25.74
3	9	1	5.56	5.42	6.65	5.69	84.21	71.59	35.78	25.32
4	8	2	6.00	4.90	7.23	5.75	94.79	68.97	40.38	25.80
5	7	3	5.29	4.47	6.21	5.33	54.86	60.90	18.57	24.34
6	8	2	6.13	4.82	6.81	5.21	104.88	62.48	30.13	23.54
7	8	2	6.13	6.11	6.54	7.20	97.60	98.91	34.50	37.65
8	9	1	5.11	4.82	6.06	5.76	57.82	73.46	29.78	31.36
9	10	0	2.40	2.89	2.39	3.41	17.34	23.12	8.00	11.93
10	10	0	2.40	3.82	2.09	4.29	9.92	45.21	5.90	18.03
11	10	0	1.60	3.26	1.33	3.59	11.09	36.02	6.20	13.26
12	10	0	1.90	2.64	1.98	2.89	9.85	7.57	3.40	2.99
13	10	0	3.70	5.59	4.41	6.74	39.79	92.95	18.00	38.80
14	3	7	7.67	9.20	8.00	11.12	124.23	151.25	50.00	56.28
15	10	0	5.20	5.30	5.70	6.21	65.85	82.50	26.00	33.47
16	8	2	3.63	4.31	4.35	5.22	41.53	57.11	22.38	26.55
17	10	0	2.70	6.53	2.52	7.51	12.35	114.60	5.40	40.10
18	4	6	7.00	4.96	8.31	5.81	170.40	58.34	58.75	23.00
19	10	0	2.50	7.28	2.53	8.42	16.03	105.52	5.20	32.99
20	10	0	4.10	4.18	3.78	5.00	28.40	35.01	12.40	16.55
21	10	0	1.80	1.91	2.14	1.93	13.69	6.47	5.40	3.65
22	10	0	4.70	3.75	6.12	4.03	32.22	46.88	15.20	16.51
23	4	6	5.00	5.33	6.28	5.89	55.68	84.32	23.25	29.20
24	10	0	1.30	2.71	1.13	2.79	6.45	20.40	3.20	8.13
25	9	1	4.33	4.58	5.06	5.16	33.37	75.75	18.11	30.46
26	10	0	1.40	0.38	1.31	0.08	6.11	-46.64	4.00	-13.51
27	1	9	7.00	5.70	10.00	6.75	121.50	89.43	42.00	37.77
28	7	3	8.71	5.05	9.09	5.61	180.04	64.01	56.43	24.63
29	9	1	6.44	4.56	7.75	5.31	137.88	57.27	48.33	21.58
30	7	3	4.86	4.87	6.62	5.43	47.70	52.69	19.29	21.71
31	7	3	6.29	6.07	6.73	7.07	140.36	96.15	36.14	37.14
32	4	6	5.25	6.22	5.81	7.20	141.58	106.62	46.50	42.20
33	10	0	1.00	1.75	1.00	1.73	5.58	-16.49	1.10	-5.10
34	1	9	10.00	6.49	10.00	7.54	71.00	112.06	41.00	42.51
35	5	5	4.40	4.11	6.55	4.70	80.82	45.75	35.20	19.52
36	9	1	4.22	3.57	4.91	3.99	28.42	36.83	10.56	14.66
37	7	3	8.00	4.60	9.11	5.29	194.59	72.04	69.29	28.51
38	8	2	5.75	5.17	6.33	6.15	40.85	76.88	23.63	33.03
39	7	3	5.57	4.92	6.38	5.55	54.26	64.98	20.43	24.72
40	10	0	3.10	2.26	2.93	2.17	8.41	11.22	3.20	3.05

Table 12: User study data compared to difficulty equation results for Lazors levels (levels 1-40)

Appendix

Level	Plays		Difficulty		Scaled Difficulty		Time (s)		Moves	
	Solved	Skipped	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.
41	7	3	6.86	6.34	8.13	7.30	96.10	109.27	34.14	38.59
42	8	2	4.25	3.82	6.15	4.05	49.71	32.60	24.38	16.42
43	10	0	2.10	2.32	1.83	2.51	9.50	14.17	4.00	5.13
44	5	5	6.00	5.36	6.36	6.26	72.62	67.44	32.80	30.24
45	8	2	4.25	4.50	6.19	4.96	21.19	65.87	11.00	24.32
46	5	5	4.60	7.22	5.25	8.52	86.26	98.46	37.20	41.79
47	10	0	4.60	4.39	5.57	4.72	57.04	57.19	22.90	21.64
48	10	0	2.10	2.89	2.23	3.26	15.36	27.45	7.10	12.83
49	9	1	4.67	3.06	5.12	3.33	44.20	37.20	14.11	12.98
50	2	8	5.50	9.34	5.31	10.95	95.50	190.09	42.50	69.16
51	9	1	6.44	5.94	8.18	7.07	131.30	75.36	54.11	33.28
52	10	0	1.90	2.36	2.08	2.48	8.54	0.34	4.00	3.03
53	10	0	4.10	4.21	4.97	4.36	16.92	61.17	7.40	19.28
54	9	1	5.89	4.86	6.48	5.47	78.33	69.43	36.56	28.47
55	9	1	5.89	4.74	7.36	5.14	74.66	83.42	27.67	31.16
56	10	0	6.30	5.21	7.11	5.92	81.98	73.41	33.80	30.99
57	8	2	7.00	7.54	8.36	8.53	206.38	125.19	72.13	45.33
58	10	0	2.00	2.25	1.90	2.32	9.41	6.31	4.50	2.25
59	9	1	3.44	3.72	4.74	4.15	52.94	42.42	14.33	18.32
60	9	0	3.33	2.50	3.71	2.74	18.17	21.36	6.78	8.54
61	9	0	1.33	1.93	1.38	2.21	9.86	5.84	3.89	3.92
62	10	0	3.80	3.63	3.85	3.87	37.20	40.46	18.80	15.28
63	10	0	4.90	2.28	5.75	2.40	22.39	14.11	8.40	5.31
64	9	0	3.22	4.47	3.14	5.08	16.21	52.61	7.44	18.39
65	9	1	6.44	5.12	7.41	5.92	108.00	78.03	49.44	30.06

Table 13: User study data compared to difficulty equation results for Lazors levels (levels 41-65)

Appendix

C.3 Move

Level	Plays		Difficulty		Scaled Difficulty		Time (s)		Moves	
	Solved	Skipped	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.
1	7	0	3.00	4.82	3.48	5.73	30.51	79.18	16.86	40.72
2	7	0	3.00	3.63	3.45	4.14	24.37	22.66	14.29	19.90
3	5	2	2.60	2.24	2.49	2.37	21.00	17.43	22.40	10.55
4	7	0	4.43	4.33	5.34	5.08	81.10	66.25	28.43	34.26
5	7	0	1.43	2.24	1.21	2.37	9.09	17.43	6.14	10.55
6	6	1	6.67	3.97	9.38	4.77	171.10	61.91	65.83	27.58
7	7	0	2.43	2.45	2.29	2.59	17.81	15.03	6.71	7.74
8	7	0	1.43	1.64	1.43	1.54	14.89	-0.90	3.14	-0.24
9	7	0	3.29	2.63	5.02	2.93	34.69	31.71	15.29	13.18
10	7	0	1.71	2.34	1.85	2.43	8.31	15.92	6.57	8.50
11	6	1	5.17	5.18	6.00	6.23	52.20	77.12	34.00	40.60
12	7	0	3.43	5.24	4.15	6.01	16.11	69.79	19.43	40.40
13	2	5	7.00	4.24	10.00	5.25	350.85	57.23	55.50	31.72
14	7	0	4.00	5.22	6.06	6.33	35.94	78.20	17.14	38.17
15	7	0	3.57	2.56	5.05	2.74	35.07	14.04	19.43	10.47
16	7	0	2.14	1.97	2.10	2.29	8.29	11.11	3.71	5.57
17	6	1	3.83	3.03	5.09	3.44	38.93	38.54	29.50	19.72
18	7	0	3.00	3.20	3.58	3.74	41.83	26.49	22.00	16.21
19	7	0	2.00	3.23	2.11	3.89	11.23	38.42	8.43	18.27
20	7	0	2.71	2.30	2.90	2.30	12.29	14.67	7.86	11.77
21	6	1	6.50	3.82	7.38	4.77	90.77	36.19	50.17	19.26
22	6	1	2.50	3.16	2.94	3.85	21.50	47.90	10.00	19.54
23	7	0	4.71	4.03	5.61	4.75	38.81	61.66	39.71	31.04
24	7	0	2.57	1.93	2.58	2.02	5.89	13.39	3.29	6.75
25	7	0	3.86	2.60	4.21	2.80	10.54	2.11	14.71	8.96
26	6	1	6.67	5.08	7.65	6.46	133.60	76.20	66.67	37.32
27	6	1	5.33	3.33	6.38	4.18	55.90	39.10	44.50	16.80
28	6	1	5.00	4.03	5.07	4.75	64.43	61.66	37.83	31.04
29	7	0	3.00	4.13	3.19	5.13	20.26	75.88	19.00	31.96
30	7	0	4.86	3.55	4.96	4.21	58.77	39.35	32.57	21.83
31	7	0	5.29	3.91	6.30	4.86	69.56	67.36	40.43	28.51
32	7	0	3.71	3.65	4.24	4.53	23.06	46.40	13.00	21.13
33	7	0	5.57	4.14	6.58	5.02	82.61	48.67	42.29	29.19
34	5	2	1.80	2.72	1.90	3.26	35.74	46.03	5.20	16.49
35	7	0	2.86	2.83	3.33	3.52	20.17	36.28	4.86	15.04
36	7	0	2.86	2.77	2.88	2.86	16.81	9.96	13.86	12.62
37	7	0	2.14	4.10	2.08	4.87	16.51	69.25	15.43	32.10
38	5	1	3.80	3.62	4.51	4.07	55.30	39.58	25.8	22.61
39	3	4	3.33	4.00	4.16	4.91	106.57	53.53	37.00	26.66
40	7	0	6.00	4.03	6.98	5.07	98.33	64.36	54.29	28.90

Table 14: User study data compared to difficulty equation results for Move levels (levels 1-40)

Appendix

Level	Plays		Difficulty		Scaled Difficulty		Time (s)		Moves	
	Solved	Skipped	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.	Users	Diff. Eq.
41	7	0	1.86	2.33	1.96	2.51	10.63	16.27	7.57	8.90
42	7	0	2.14	3.68	1.72	4.75	19.74	49.36	7.14	22.41
43	7	0	1.86	2.85	1.93	3.15	9.10	18.79	5.71	12.45
44	7	0	1.43	1.57	1.43	1.45	6.63	-4.32	3.43	2.19
45	7	0	2.86	2.86	2.97	3.18	32.54	21.44	23.86	16.61
46	6	1	5.00	2.90	6.39	3.12	59.17	21.44	32.50	15.83
47	6	0	4.17	3.94	5.16	4.67	72.23	50.13	49.67	16.43
48	6	0	2.83	3.15	2.78	3.66	17.53	21.90	10.33	17.27
49	7	0	3.57	3.39	5.14	3.91	29.27	41.23	18.29	19.91
50	4	3	4.75	4.34	7.58	5.28	59.05	72.12	42.50	27.96
51	7	0	1.14	2.06	1.00	2.18	4.90	12.93	4.71	6.87
52	7	0	2.00	2.39	2.20	2.48	16.89	9.44	9.86	10.36
53	6	1	3.67	4.18	6.35	4.99	35.98	67.75	15.00	29.38
54	6	0	3.67	3.33	4.30	3.77	50.60	31.15	22.67	19.29
55	3	3	4.33	4.96	5.36	5.97	157.97	81.63	50.33	37.37
56	3	3	6.00	4.85	8.37	5.81	188.57	73.89	62.67	34.59
57	6	0	2.67	5.08	2.80	6.06	20.60	65.42	16.00	35.98
58	5	1	4.60	5.35	5.85	6.43	70.02	76.73	38.20	41.56
59	5	1	5.00	5.29	5.27	6.32	57.24	74.16	35.40	39.72
60	6	0	1.50	2.22	1.25	2.28	8.68	9.06	9.17	9.25
61	6	0	5.67	4.36	8.02	5.13	109.23	58.10	54.00	30.97
62	6	0	2.67	1.94	2.90	1.95	13.50	5.28	8.83	5.34
63	6	0	1.17	1.98	1.00	2.01	2.30	8.64	3.00	7.04
64	6	0	2.33	3.96	2.58	4.63	20.07	49.08	9.00	26.03
65	6	0	1.83	2.17	2.00	2.13	19.50	11.82	11.33	11.10
66	6	0	3.33	4.82	3.62	6.12	31.03	85.49	13.83	40.84
67	6	0	5.00	3.58	5.78	4.19	45.75	49.71	28.50	27.91
68	4	2	5.00	4.22	6.00	4.95	101.93	54.15	45.25	31.89
69	6	0	1.67	3.54	1.25	5.06	9.58	65.77	5.00	22.38
70	6	0	3.00	2.89	3.21	3.41	24.93	28.23	8.67	13.77
71	5	1	4.00	3.17	4.63	3.65	37.62	32.50	19.80	20.19
72	6	0	1.83	2.24	1.50	3.07	8.72	28.95	4.50	8.17
73	5	1	4.40	6.30	5.11	7.77	37.12	99.00	20.40	52.11
74	6	0	5.67	3.44	6.40	4.23	140.98	52.65	47.67	25.07
75	6	0	2.83	2.04	2.79	2.49	30.57	18.11	10.17	6.54
76	6	0	2.00	2.64	2.01	2.67	13.23	9.79	8.00	11.66
77	6	0	4.33	3.04	5.35	3.26	39.05	30.28	29.67	18.98
78	6	0	4.33	2.91	5.59	3.66	23.75	43.14	14.00	21.62
79	6	0	2.33	3.99	2.45	4.84	42.90	64.49	18.00	32.69
80	6	0	2.67	2.98	2.68	3.52	12.75	27.83	13.50	16.88

Table 15: User study data compared to difficulty equation results for Move levels (levels 41-80)

D Difficulty Equation Weights

D.1 Flow

Set	Difficulty Measure	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
1	Level Size	0.0924	0.1385	1.3030	0.1837
	Colours	-0.3223	-0.4767	-7.1158	0.1458
	Average Distance	-0.4406	-0.6110	-7.3338	-1.1696
	Corners	0.0244	0.0433	0.0479	0.1304
	w_0 (Intercept)	2.3606	2.5557	42.7309	2.5798
2	Level Size	0.0850	0.1232	1.0881	0.2408
	Colours	-0.2859	-0.4219	-2.6671	-0.1594
	Average Distance	-0.2540	-0.3806	-4.7177	-0.9792
	Corners	0.0306	0.0643	0.4017	0.0535
	w_0 (Intercept)	1.5579	1.6072	8.0120	1.8512
3	Level Size	0.0848	0.1203	1.2131	0.2393
	Colours	-0.2695	-0.3732	-4.8174	-0.0884
	Average Distance	-0.4400	-0.6493	-7.0815	-1.2869
	Corners	0.0556	0.1148	-0.1760	0.0671
	w_0 (Intercept)	2.0231	2.0766	36.4837	2.8285
4	Level Size	0.0905	0.1320	1.2266	0.2809
	Colours	-0.3426	-0.4842	-6.8249	-0.5748
	Average Distance	-0.4975	-0.7414	-8.2883	-1.7747
	Corners	0.0554	0.0997	0.8492	0.1601
	w_0 (Intercept)	2.4841	2.8225	40.4792	4.9661
5	Level Size	0.0854	0.1235	1.0135	0.2456
	Colours	-0.2704	-0.3861	-5.4640	-0.2690
	Average Distance	-0.3649	-0.5369	-5.0256	-1.1554
	Corners	0.0318	0.0719	0.7112	0.0727
	w_0 (Intercept)	1.8928	1.9091	28.1642	2.8586

Table 16: Weights assigned to the difficulty measures for difficulty, scaled difficulty, solving time and number of moves for Flow

D.2 Lazors

D.2.1 Original

Set	Difficulty Measure	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
1	Level Size	0.0498	0.0652	-0.3151	-0.0744
	Usable Tiles	0.0630	0.0772	3.4577	1.4353
	Emitters	0.7062	0.8775	10.4125	3.0829
	Receivers	0.2320	0.1632	4.0768	0.6853
	Mirrors	0.4704	0.5233	11.1186	3.7394
	w_0 (Intercept)	-1.5207	-1.7611	-73.3714	-24.6679
2	Level Size	0.0510	0.1001	-1.0097	-0.3111
	Usable Tiles	0.0497	0.0268	4.1865	1.6610
	Emitters	1.0041	1.1962	15.6460	4.5526
	Receivers	0.3080	0.3160	6.3974	1.5269
	Mirrors	0.5434	0.6404	13.0041	4.4664
	w_0 (Intercept)	-2.1782	-2.9320	-91.0790	-30.5123
3	Level Size	0.0714	0.0734	-0.1088	-0.0505
	Usable Tiles	0.0210	0.0512	2.9697	1.3027
	Emitters	0.5775	0.7221	7.9978	2.8595
	Receivers	0.1388	0.0982	2.6652	0.3731
	Mirrors	0.4484	0.5510	11.0072	3.9764
	w_0 (Intercept)	-1.0045	-1.6110	-70.0758	-25.3205
4	Level Size	0.0533	0.0843	-0.7945	-0.2828
	Usable Tiles	0.0619	0.0442	3.7854	1.5317
	Emitters	0.7129	0.7397	11.4068	3.2034
	Receivers	0.0818	0.0452	3.0043	0.4297
	Mirrors	0.5362	0.6550	12.0247	4.2387
	w_0 (Intercept)	-1.5456	-1.8299	-72.2776	-24.0322
5	Level Size	0.0436	0.0579	1.6555	0.4714
	Usable Tiles	0.0632	0.0848	0.1824	0.4862
	Emitters	0.8150	1.0290	13.3161	4.2724
	Receivers	0.1771	0.1137	4.4798	0.8734
	Mirrors	0.4642	0.4940	11.5984	3.7987
	w_0 (Intercept)	-1.5475	-1.9787	-67.5938	-23.1951

Table 17: Weights assigned to the difficulty measures for difficulty, scaled difficulty, solving time and number of moves for Lazors

D.2.2 Adjusted

Set	Difficulty Measure	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
1	Level Size	0.0261	0.0331	-1.2192	-0.3649
	Usable Tiles	0.0695	0.0872	3.7885	1.5389
	Emitters	0.2634	0.2871	-5.8929	-2.1771
	Receivers	0.0727	-0.0515	-1.9436	-1.2515
	Mirrors	0.1244	0.0540	-2.1604	-0.5258
	Reflections	0.2083	0.2888	8.4145	2.6888
	Intersections	0.4326	0.5512	14.2205	4.6460
	w_0 (Intercept)	-0.1318	0.0892	-22.3278	-8.1984
2	Level Size	0.0174	0.0594	-2.0634	-0.6516
	Usable Tiles	0.0680	0.0524	4.6547	1.8193
	Emitters	0.5614	0.7300	-0.5712	-0.5362
	Receivers	0.1617	0.1536	1.3107	-0.0847
	Mirrors	-0.0265	-0.0310	-5.5393	-1.4834
	Reflections	0.4729	0.5691	14.9937	4.8358
	Intersections	0.1914	0.1574	8.4515	2.5711
	w_0 (Intercept)	-0.9182	-1.5742	-45.9499	-16.2937
3	Level Size	0.0538	0.0510	-0.5912	-0.2185
	Usable Tiles	0.0190	0.0500	2.9352	1.2855
	Emitters	0.1643	0.1945	-3.3292	-1.0726
	Receivers	0.0098	-0.0668	-0.8752	-0.8552
	Mirrors	0.1118	0.1096	1.6093	0.7615
	Reflections	0.2084	0.2817	5.9423	1.9990
	Intersections	0.4224	0.4954	10.9339	3.9748
	w_0 (Intercept)	0.2964	0.0477	-34.4494	-12.9434
4	Level Size	0.0377	0.0635	-1.3460	-0.4614
	Usable Tiles	0.0470	0.0282	3.2863	1.3711
	Emitters	0.1974	0.0492	-6.8661	-2.7169
	Receivers	-0.0252	-0.1024	-0.8218	-0.8110
	Mirrors	0.1159	0.0720	-3.0207	-0.6413
	Reflections	0.2604	0.3756	9.4282	3.0619
	Intersections	0.5109	0.6090	17.5557	5.6677
	w_0 (Intercept)	-0.0552	0.1468	-19.5922	-6.9678
5	Level Size	-0.0199	-0.0196	0.3234	0.0577
	Usable Tiles	0.1181	0.1519	1.3274	0.8407
	Emitters	0.0940	0.1438	-3.2429	-1.0995
	Receivers	0.0221	-0.0768	0.8333	-0.3223
	Mirrors	-0.0847	-0.1811	-1.3542	-0.4532
	Reflections	0.3203	0.3954	8.0188	2.6977
	Intersections	0.5932	0.7225	11.8304	3.5769
	w_0 (Intercept)	0.3372	0.3345	-24.5300	-9.2574

Table 18: Weights assigned to the extended set of difficulty measures for difficulty, scaled difficulty, solving time and number of moves for Lazors

D.3 Move

D.3.1 Original

Set	Difficulty Measure	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
1	Moves	0.3376	0.4201	6.8295	3.7237
	Level Size	0.0173	0.0377	1.3209	0.4181
	Balls	0.3897	0.6420	4.6226	4.5600
	Colours	-0.0347	-0.0224	5.7384	-1.0082
	Rocks	-0.0641	-0.1083	-4.4894	-1.6029
	w_0 (Intercept)	-0.1398	-1.2855	-36.2740	-17.8358
2	Moves	0.2819	0.3672	4.5608	2.8333
	Level Size	0.0730	0.1195	3.2948	0.9305
	Balls	0.3079	0.4839	8.5212	2.7535
	Colours	0.1692	0.3588	7.4496	1.0752
	Rocks	-0.1559	-0.2203	-6.8244	-2.4926
	w_0 (Intercept)	-0.6628	-2.2014	-65.2961	-16.6493
3	Moves	0.2689	0.3269	4.2191	2.6947
	Level Size	0.0760	0.1193	3.2362	0.9372
	Balls	0.5716	0.9158	12.2607	5.4858
	Colours	-0.1096	-0.0348	3.4750	-1.0951
	Rocks	-0.0827	-0.1309	-5.3067	-1.7269
	w_0 (Intercept)	-1.2405	-2.9904	-75.4553	-24.1578
4	Moves	0.2707	0.3389	3.5647	2.6474
	Level Size	0.0451	0.0671	2.4267	0.6475
	Balls	0.3896	0.5294	9.9488	4.1447
	Colours	-0.0177	0.0136	3.1572	-1.1969
	Rocks	-0.0709	-0.0899	-4.2446	-1.5968
	w_0 (Intercept)	-0.1872	-1.0074	-51.6057	-13.6686
5	Moves	0.2690	0.3188	4.7053	3.0779
	Level Size	0.0690	0.1273	3.0519	0.9855
	Balls	0.6964	1.0359	17.5241	7.9601
	Colours	0.0550	0.2602	6.3349	-0.3996
	Rocks	-0.0608	-0.0939	-4.4436	-1.6800
	w_0 (Intercept)	-1.9096	-4.1466	-100.3506	-36.3881

Table 19: Weights assigned to the difficulty measures for difficulty, scaled difficulty, solving time and number of moves for Move

D.3.2 Adjusted

Set	Difficulty Measure	Difficulty Rating	Scaled Rating	Time to Solve	Number of Moves
1	Moves	0.1991	0.2494	3.8929	2.2686
	Level Size	0.0213	0.0427	1.4064	0.4605
	Balls	0.0571	0.2321	-2.4320	1.0644
	Colours	-0.0160	0.0008	6.1376	-0.8104
	Rocks	-0.0950	-0.1464	-5.1444	-1.9274
	Counterintuitive Moves	0.6984	0.8606	14.8119	7.3394
	w_0 (Intercept)	1.1372	0.2880	-9.1931	-4.4170
2	Moves	0.1743	0.2061	2.8480	2.0012
	Level Size	0.0636	0.1055	3.1457	0.8580
	Balls	0.1414	0.2345	5.8705	1.4657
	Colours	0.1575	0.3413	7.2631	0.9846
	Rocks	-0.1532	-0.2163	-6.7819	-2.4720
	Counterintuitive Moves	0.4576	0.6852	7.2836	3.5385
	w_0 (Intercept)	0.2672	-0.8089	-50.4938	-9.4580
3	Moves	0.1022	0.0968	0.6491	0.9718
	Level Size	0.0718	0.1135	3.1467	0.8940
	Balls	0.2084	0.4145	4.4844	1.7329
	Colours	0.0155	0.1379	6.1536	0.1976
	Rocks	-0.1180	-0.1796	-6.0634	-2.0921
	Counterintuitive Moves	0.6539	0.9027	14.0021	6.7575
	w_0 (Intercept)	0.3144	-0.8438	-42.1585	-8.0886
4	Moves	0.1206	0.1382	0.2142	1.1056
	Level Size	0.0345	0.0530	2.1907	0.5389
	Balls	0.0195	0.0345	1.6876	0.3429
	Colours	0.0684	0.1287	5.0773	-0.3133
	Rocks	-0.1142	-0.1478	-5.2111	-2.0416
	Counterintuitive Moves	0.7780	1.0404	17.3647	7.9910
	w_0 (Intercept)	1.3923	1.1048	-16.3541	2.5538
5	Moves	0.0588	0.0535	1.0607	1.2734
	Level Size	0.0595	0.1152	2.8866	0.9036
	Balls	0.3921	0.6519	12.2472	5.3473
	Colours	-0.0806	0.0891	3.9836	-1.5637
	Rocks	-0.1153	-0.1626	-5.3881	-2.1476
	Counterintuitive Moves	0.8153	1.0289	14.1368	6.9995
	w_0 (Intercept)	0.1125	-1.5949	-65.2915	-19.0293

Table 20: Weights assigned to the extended set of difficulty measures for difficulty, scaled difficulty, solving time and number of moves for Move