3 Analysis

To predict difficulty, we first needed a deeper understanding of how Longcats' is designed. We needed to understand what possible states and actions are actually available to the player and how players parse this information. Therefore, we started with an analysis phase. The method and results of this analysis are described in this chapter. It is split into two sections, one that looks at the levels and what variables can be gained from them, and another that looks at the player data to give a value on how difficult the existing levels are.

3.1 Level Analysis

This section has two parts. The first part describes the graph representation used to analyze Longcat levels. The second part lists and describes the relevant variables we found in analyzing the graphs.

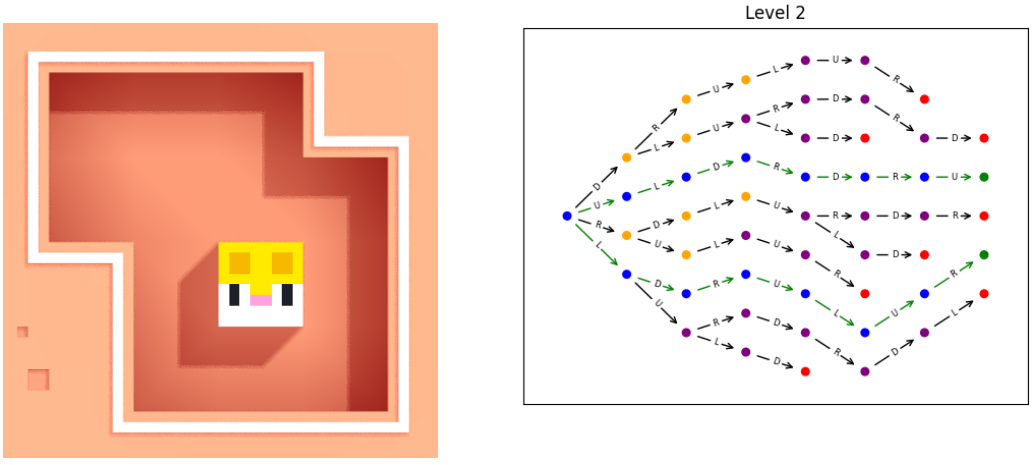

Figure 3.1: Level 2 of Longcat (left) and its solution graph (right)

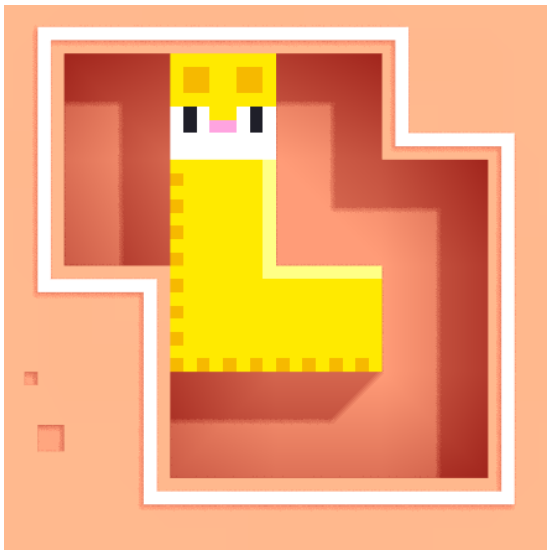

Figure 3.2: Level 2 in a dead state.

Graph Representation

Longcat levels don't always have an apparent difficulty. Unlike games such as the ones analyzed by Kreveld et al. [16], where more puzzle elements and level size correlate with difficulty, Longcat has less clear variables. A Longcat level with a large size can be trivial, and the same level can change substantially in difficulty by adding one or two walls. Because of this, we choose not to look at the levels in the grid representation but instead as a graph.

In this graph representation, each board configuration, or state, is represented by one node, and an edge represents each action. So when the level starts, we are in a start state represented by a node. When the player moves in a direction, it adds cat cells and moves the head so that we are in a new state and, thus, a new node. These two nodes are connected by an edge representing the directional action used. With this, we can use a simple breadth-first search to find all possible states of a specific level and what actions connect them. We can then write this as a graph of nodes and edges. We have chosen to call this the *graph representation*.

As an example, we will look at level 2 in Longcat and its corresponding solution graph, which can be seen in Figure 3.1. The nodes go chronologically from left to right, so the leftmost node is the start state. Each edge is labeled with the action it represents. U for up, D for down, L for left and R for right.

The graph is also color-coded. Each green node represents a success state, which is a state where the level is completed. Each red node represents a fail state, which is a state where the levels has to be restarted. Blue nodes are states from where a solution is reachable, and edges are green if they connect two blue nodes. This makes it easy to see all solutions in the graph. As we can see in Figure 3.1, level 2 has two solutions.

Purple nodes are *dead states*. A dead state is a state where the level is separated into two areas of empty cells. Figure 3.2 shows an example of this. We choose to track this since it is evident for most players that there is no solution from a dead state, and so it is not a state the player will continue to explore from. This means many dead states will likely not lead to a more difficult level.

Yellow nodes are *indeterminate states*. Indeterminate states are defined by not being solutions, success, failure, or dead states. Meaning that these states will not lead to a solution, but this is not obvious, or at least not as obvious as in a dead state. We call them indeterminate states since it is difficult for the player to determine if they have found a solution from them, not be confused with the mathematical concept of determinants.

Graph Variables

With this graph representation, we can now better understand how many possible ways a level can be explored and how many of these paths lead to a solution. The intuition is that a large and more complex graph, with many branches and few solutions, will have a more difficult level. We analyzed a few different attributes of the graphs to attempt to extract this information.

- **Number of states**. This is the number of nodes in the graph. A greater number of states means more states to search before finding the solution.
- **Size**. The number of cells in the grid representation.
- **The number of branches**. If a node has more than one out degree, we say that node has a branch for each additional out degree above one. In the level this equates to the number of choices the player can make. If every state, aside from the success state, has an out-degree of one, then the level is trivial since the player can only make one move in each state and will then eventually reach the end automatically. Intuitively, the inverse should also be true. A great number of branches should lead to many choices having to be evaluated and a more difficult level.
- **Number of solutions**. The number of success states. A great number of success states means more ways of finding a solution and could make the level easier.
- **Number of fails**. The number of fail states. As with a great number of success states a great number of fails states could lead to a more difficult level.
- **Shortest solution length**. Number of nodes in the shortest path from the start node to the nearest solution node. This equates to the smallest number of actions needed to solve a level.
- **Number of solution branches**. Solution branches are the number of branches from a node that is part of a solution (i.e., blue nodes). This equates to the number of actions the player can do incorrectly when following a solution.
- **Number of non-dead states**. As discussed previously, dead states can create parts of the graph that the player won't explore.
- **Number of indeterminate states**. Unlike dead states, these should give a good indication of how many states the player has to search through to find a solution.
- **Number of indeterminate branches**. Indeterminate branches are the number of branches from an indeterminate state. This equates to the number of choices in the level, where it is unclear whether the level is unsolvable.

Not all these variables were defined at the start of the analysis. Some were added and changed during implementation to adapt to observed problems and player behavior. Not all of them proved useful, which was expected, but exploring all of them and later removing irrelevant ones, as done by Kreveld et al. [16] in their work, seemed like a good method.

Data Analysis

From the level analysis, we can now represent the level's complexity through variables. Now we need to analyze how these variables correlate with the actual difficulty of a level. So, we need a set of levels labeled with their difficulty.

Kreveld et al. [16] achieved this by doing a user study where each participant was asked to rank levels after playing them. We decided to use large-scale player analytics to determine

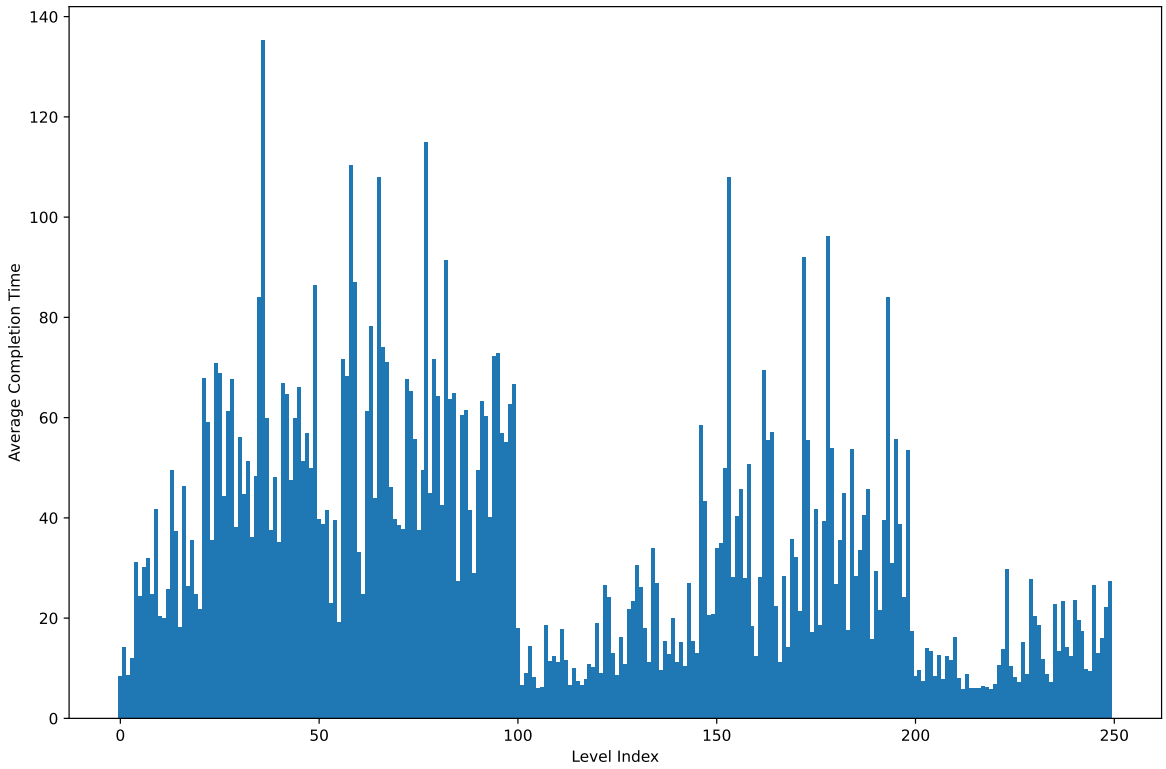

Figure 3.3: Average completion time for all 250 levels in Longcat.

difficulty, so instead, we looked at the player data from the 250 levels that were already part of Longcat. We could extract each player's time to complete a level from these player analytics. With around 900-12000 players per level, we could get a strong average completion time for each level. This number is assumed to correlate with difficulty. This assumption is not completely true. A level can take many time-consuming actions to complete but not be difficult, but we assume that this time is generally outweighed by the time it takes to think about and solve a level for most non-trivial levels.

Figure 3.3 shows the resulting average completion time for all levels. As expected, we can see the completion time rise during the period level 0 to around 20. This is because the first levels are designed to teach the player how to play the game. So, they are designed to rise in difficulty. Interestingly, we can see that levels 100 - 150 and 200 - 250 have a shorter completion time. This is also by design. The first 100 levels of Loncat are designed by hand, with only some generation assistance used for levels 50 - 100. Levels 100 - 250 are essentially completely procedurally generated using a naive approach. This generator was instructed to create more difficult levels for the range 150 - 200 and easier ones for the rest. Another observation of note is level 37, which has a much larger completion time.

We also got a distribution of how many players have played each level from the data. This can be seen in Figure 3.4. As expected, the player count drops off initially as players lose interest. We can also see a player drop-off after level 37, meaning many players started level 37 but never completed it. From these observations, level 37 seems like an outlier, which is

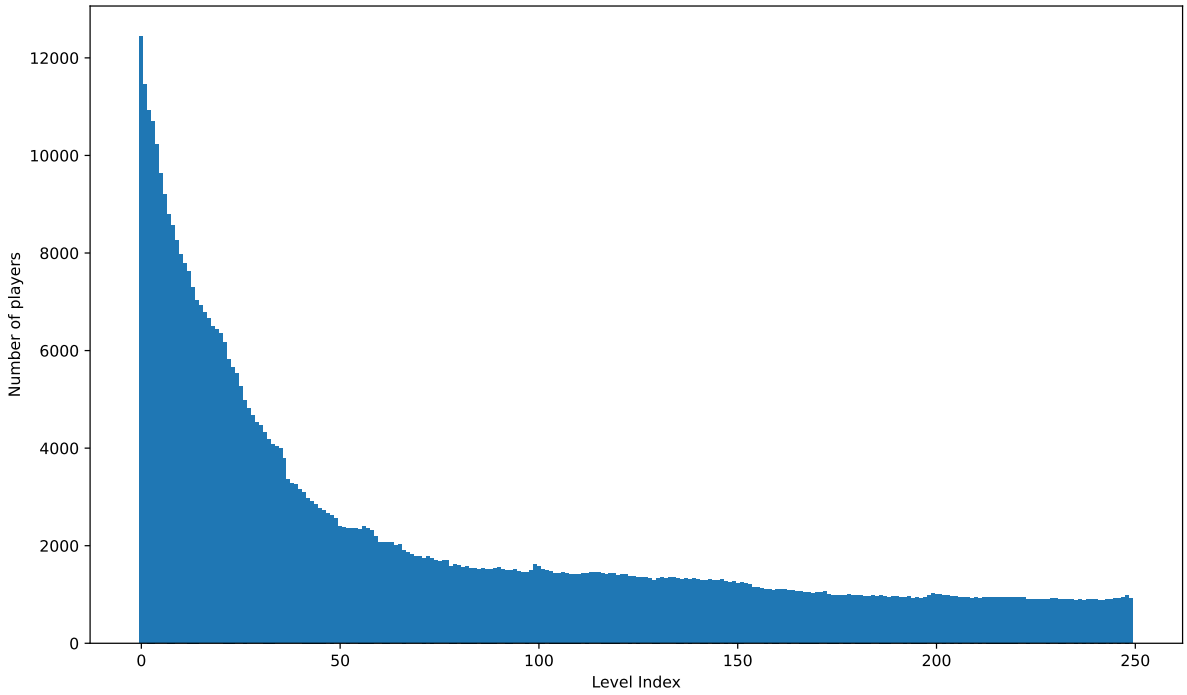

Figure 3.4: Player distribution over all 250 levels in Longcat.

also true in the graph representation seen in Figure 3.5, as it has one of the most significant numbers of nodes in the game.

We then calculated the graphs for all 250 levels, extracted the variables, and plotted them against the average solution time. We also fitted a simple linear model for each variable to observe the corresponding r-values. The results of this can be observed in Figure 3.6 and 3.7.

We can see that most variables and the average solution time are correlated. The strongest correlation is with the number of indeterminate states. This does not give us a definitive answer on what variables should be used, but it indicates that they can, to some extent, be used for prediction.

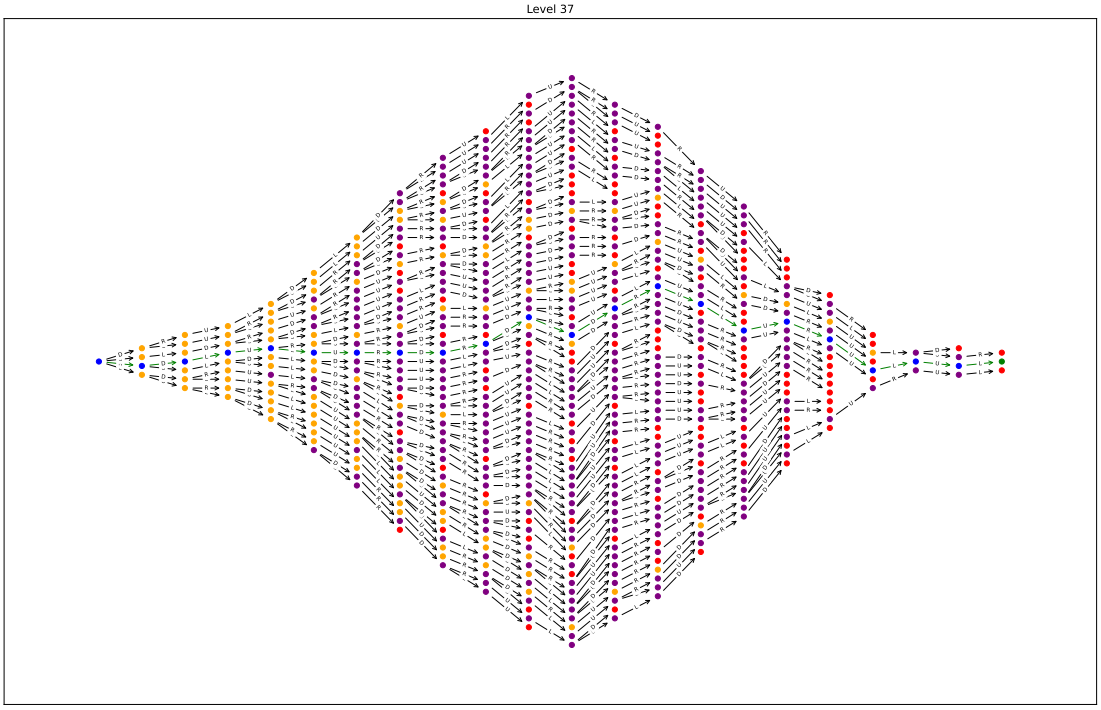

Figure 3.5: Level 37 graph representation.

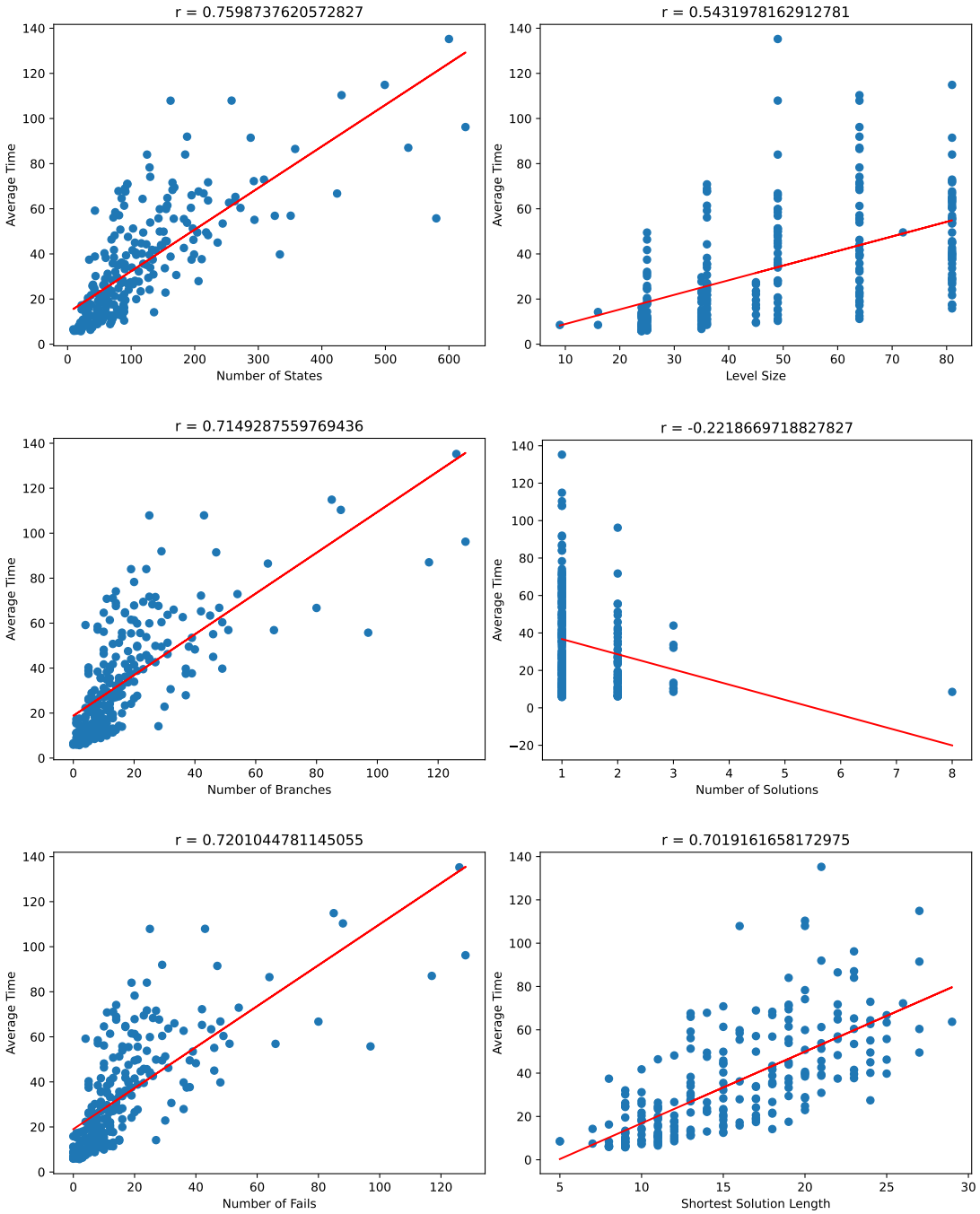

Figure 3.6: Extracted variables plotted to the average solution time for all 250 levels

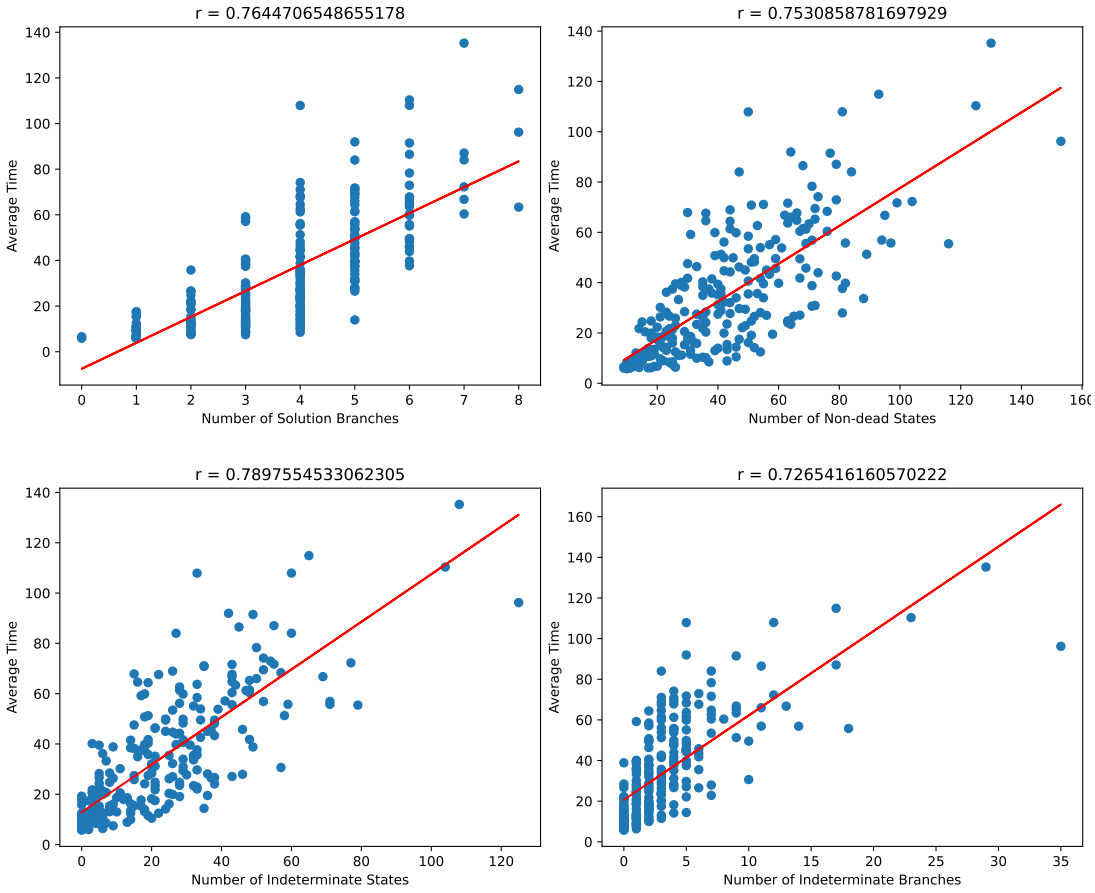

Figure 3.7: Extracted variables plotted to the average solution time for all 250 levels