Robot Game Visualization Dashboard

:
Overview: strategy
Hey, we are team robopilot. Now I will introduce our guessing strategy. Here is the visualization for our robogame. There are two parts we are concerned about in deciding which robot we tend to guess: productivity and popularity. We have implemented a heatmap for the correlation coefficient to give a better view of the productivity of each robot. Then we combined the heatmap into the popularity network. In the first 50 XTUs, we tend to guess robots with more popularity and earlier deadlines. After we get enough part hints, we can briefly generate a heatmap for different parts of the robot, using color encoding to combine with the network. For the next step, we will have one more rule to choose robots. In the next 50 XTUs we tend to choose the robot with more productivity. We have also implemented a scroll bar for the network graph to see which robots are available at a particular time. Every time we choose the robots we want to guess, we will tell the hacker what our interests are so that we can get more hints about the particular robots and fit the number guessing curve. Finally, we will manually output our guesses for robots before their deadline on the game page.

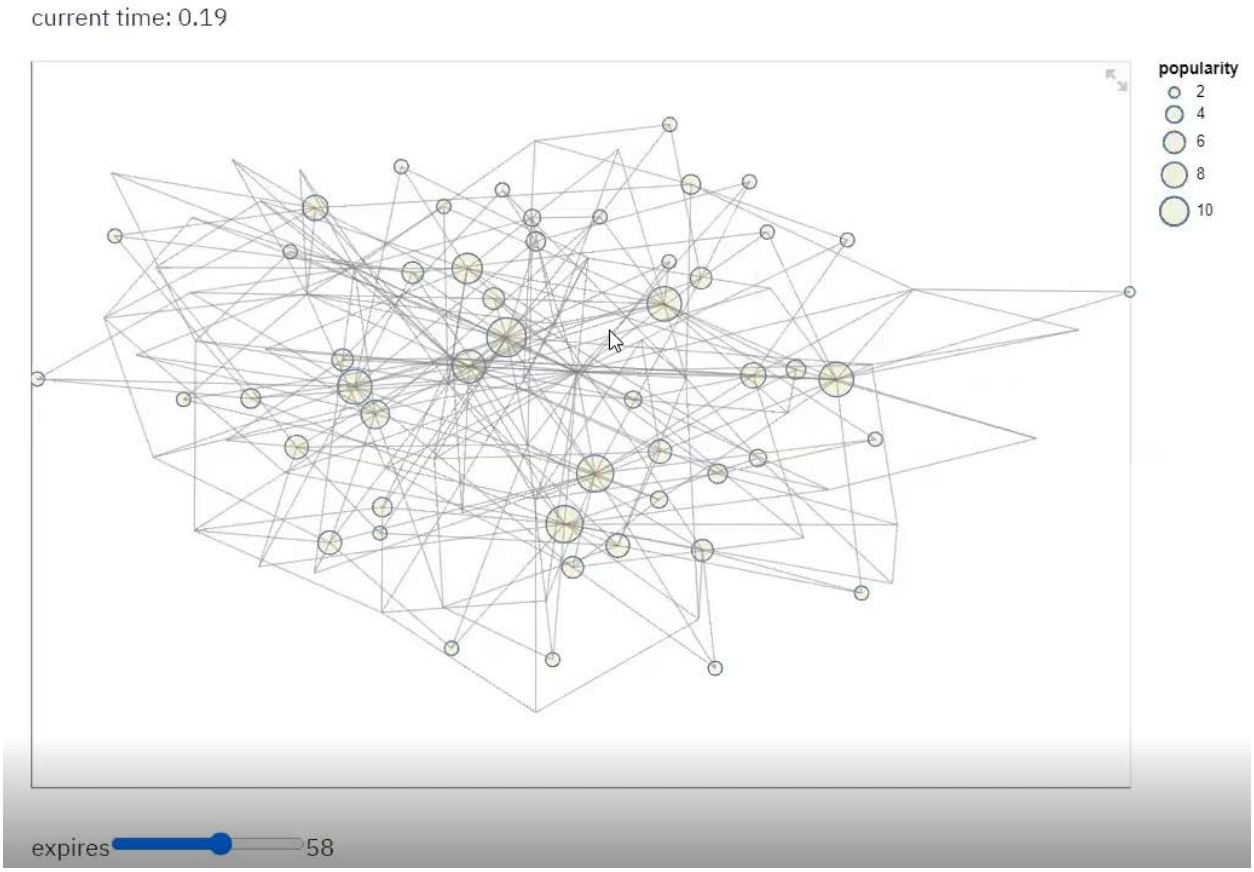
network_before:

For the first 50 seconds, we are looking at the popularity and availability of the robots in the network graph. Each node is represented by a robot and the edge between them represents a direct relationship. A node's popularity is defined as the number of degrees a node has and is encoded by the node size. Therefore, we are paying attention to bigger nodes at the start of the game.

While we are checking a node's popularity, we also take its availability into consideration. A node's availability is defined as whether or not it's already expired at the current moment. Therefore, for every 6 XTU, we check whether each node's expiration time has already passed, and update the availability attribute within the node. The unavailable nodes will be hidden in the network graph so that we will only focus on the available ones. The node color is set as beige, which is the color for average values in the node color we are implementing after 50 seconds when we are considering productive parts. This would reduce the amount of time for users to process color codings throughout the game

In addition, we have implemented a slider that could show that, at any given time from start to finish, what are the nodes that will expire within the time frame between the slider-selected time and the current time so that we could focus on guessing those nodes first before they expire soon. Both the availability and slider function will persist throughout the game.

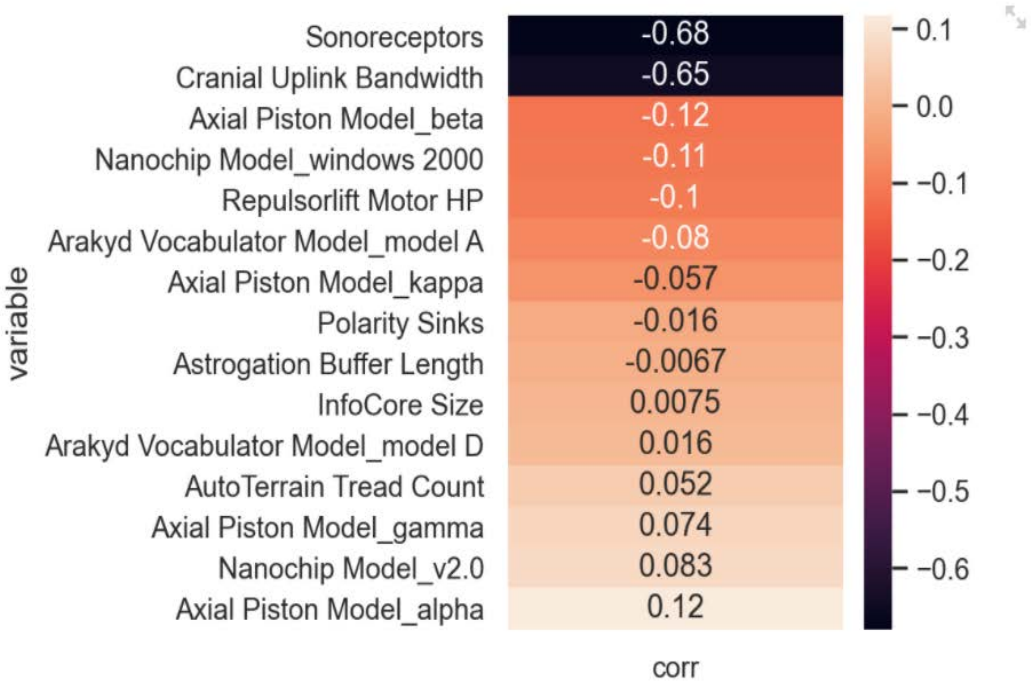
After we combine the above two actions, we select those nodes that are of interest to us and put their ids into the input bar above. The selected nodes will then be computed by the hackers to generate more numbers to fit the number-guessing line.

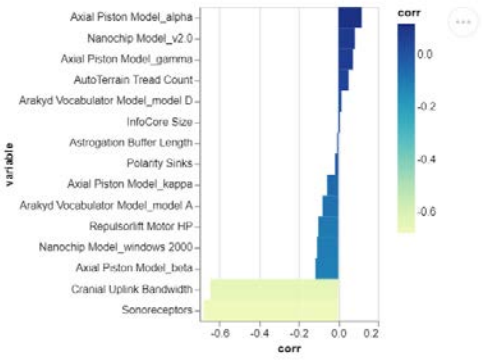


Heatmap & pass top3 variables & Input(Yixin): parts vs productivity, correlation, top 3 related parts, set interest parts

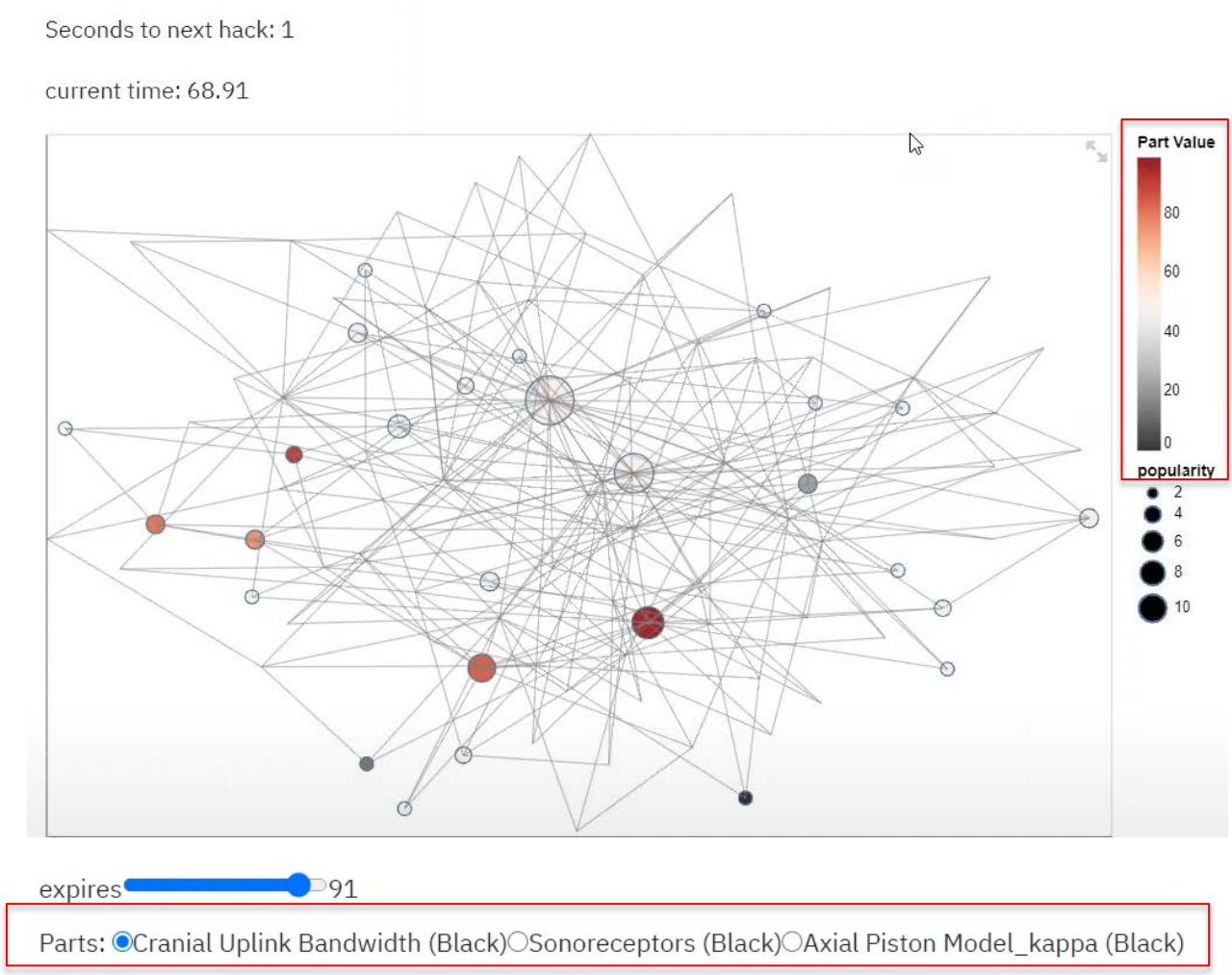
The goal of the heatmap is to identify the most effective parts of robots to predict their productivity. The correlation coefficient between parts and productivity are sorted and visualized in the heatmap when it comes to enough part hints and productivity data. The black parts are negatively related to productivity and the red parts are positively related. It updates every 6 seconds.

Here are the heatmaps of the seaborn version and Altair version. In the seaborn version, the value is encoded in color with text annotation of it while the altair version encoded value in both color and length. The former version is chosen because we can read the correlation coefficient value directly from it. It is more efficient because it fits the user's reading habits.





Heatmap will pass the top 3 three parts to the hacker to set the interest parts. The top 3 parts are parts with the highest absolute correlation coefficient, which are on the top or bottom of the heatmap. We don't set specific interest parts in the first 50 XTU, so hints are distributed in parts evenly for the reasonable correlation coefficient. In the second 50 XTU, the top 3 parts will be passed to the hacker to set the interest part, so we will get more hints of those parts. The ranking of parts is still being updated to make sure that it is not generated by accident; once a part gets the ranking by accident, with enough hints, it will no longer stay in the top 3. The top 3 parts with their values are also passed to the network for users to predict productivity.



Once the node is decided, it will be entered manually passed to time series chart for prediction.

network_after: [color encoding: pos/neg correlated, value of parts] and set interest id

Network after: For the last 50 XTUs, we take productivity into consideration. We implement a radio button where users can select one robot part to guess the productivity. For robot parts that are positively correlated to productivity, users need to focus on the large part values, which are indicated by red color. For parts that are negatively correlated to productivity, users need to focus on the small part values, which are indicated by dark color. To better help users to locate a productive robot in both cases, we label different buttons by

pointing out which color to focus on. For robots which do not have the information of part values, we set their part values as the mean value so that those nodes' colors are light, which will not distract users a lot. The nodes color will change every 6 XTU depending on the robot part values provided by hackers.

For the design choices, we originally used a sequential color scheme to encode the part value. But in the case of negatively related robot parts, a productive robot shows a low part value. Then those robots are not highlighted. In this case, we think the diverging color scheme is more effective (shown in the picture) so that both large values for positive correlation and small values for negative correlation are highlighted.

One thing we could have implemented but not due to the time issue is a heatmap. (shown as the picture), We aim to put robot ids on the rows and the factors we consider such as the popularity, expiration time, and robot parts on the column. The color encodes the values of those variables. The robots with the highest popularity, robots are most likely to expire, and robots of the largest or smallest robot part values are highlighted. Users can order robots by the column values so the vis is efficient. We think this visualization has a similar but better function than our network viz, but due to the limited time, we still use the network viz.

In conclusion, from the network viz, users can quickly figure out which robots are most popular and productive. Then users can make better choices to set interesting robot id to help guess numbers and win the robots.

Time Series for number guessing

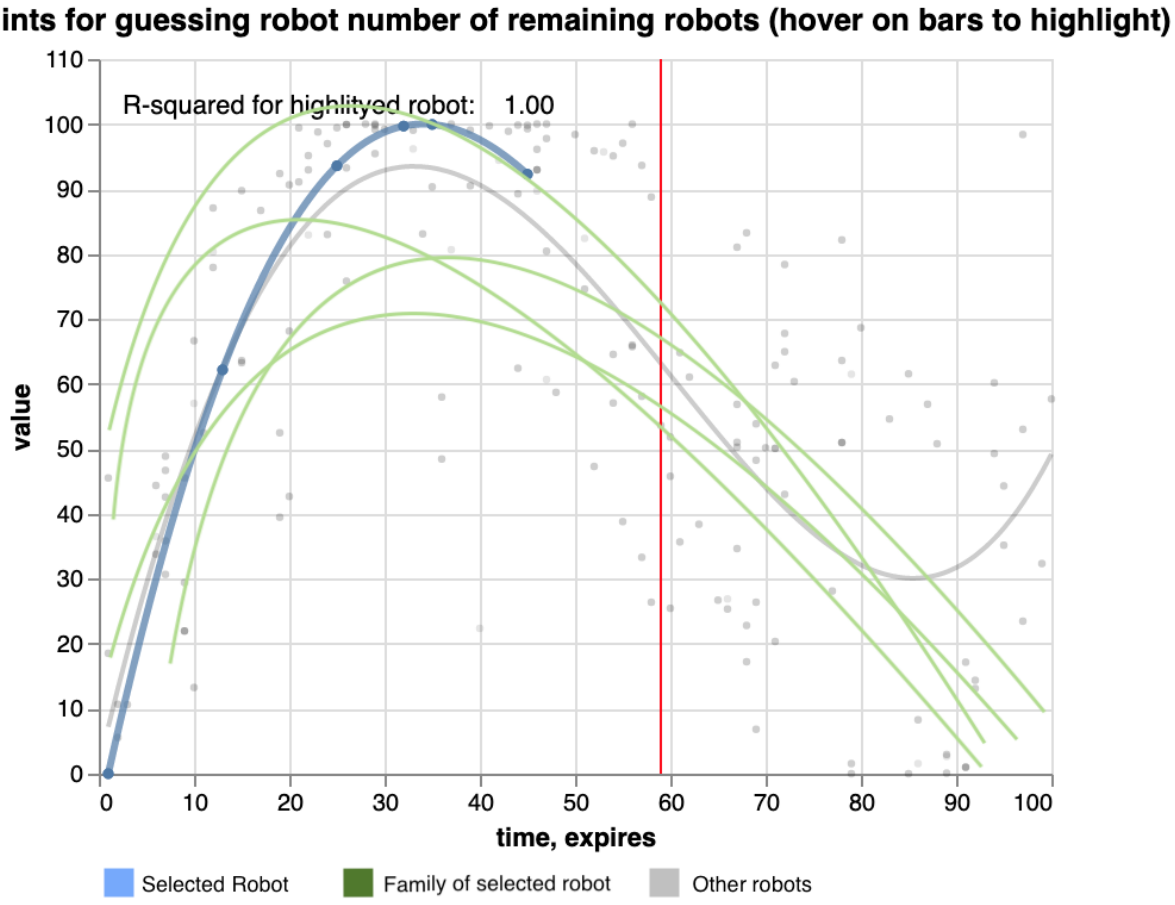
The time series visualization and its linked bar chart is intended to facilitate decision making when guessing the number of a particular robot. The chart displays all hints collected for all winnable robots that haven't expired, their expiration time, and a linear regression model for their random number generator function.

Users can hover on the bars on the right side to select a robot, whose id will be highlighted on the time series, according to how many hints are available for it and other contextual information. The hints for the highlighted robot are represented as larger blue dots with a blue polynomial regression line fitted on top; other points and the regression line for all hints are shown in the background in light grey. The color choice makes the selected robot pop up while providing contextual information of the overall model which makes guessing possible even when we lack data at the beginning of the game. The regression line is based on a polynomial model with an order of 4. We picked the order despite its tendency to overfit because it presents the

data more accurately once we've gathered enough. A red vertical line marks the expiration time of the robot.

When designing the viz, we started with an ambitious idea prototyped here: three colors, each representing the hints for the selected robot, the family of the selected robot, and other robots. We were successful in defining a function to parse a custom level of the family tree (i.e. predecessors and successors) for a given robot id. However, we weren't able to call the function within Altair's transform filter or use the FieldOneOfPredicate. Another thing we would like to implement given more time is adding another vertical line marking the current time, further signaling the urgency of making a guess for the robot.

Additionally, we brainstormed a viz where the user would select the robot by an Altair dropdown menu. Through iterations we realized that a dropdown menu is cumbersome: it refreshes with the viz every 6 seconds, which isn't enough time for the user to make a selection. We improved the usability by having a linked bar-chart on the side, containing the top 20 available robots with the highest number of hints, as a dynamic filter triggered by mouseover and cleared by mouseout.



In summary, during gameplay, to use the chart along with the other visualizations to execute our strategy, we start by looking at the network viz. We filter the network to only show immediately

expiring robots; hover on each of the larger nodes, which are preferred for having more popularity, to identify their ids. Next, through an input text box, we set interest on these ids to gather more information on them, reflected by the time series chart. Last, we make guesses of those desired robots, whose high urgencies are guaranteed by the network viz slider, high popularity guaranteed by network viz size, and high prediction accuracy guaranteed by sufficient data shown on the time series.