Using Interaction Networks to Identify Unproductive Solution Steps in Multistep Problems

By: Drew Hicks, North Carolina State University
Michael Eagle, Carnegie-Mellon University
Elizabeth Rowe, EdGE at TERC

January 2017
About Analytics for Learning (A4L)

The availability of data coming from digital learning environments is creating the possibility to measure learning like never before. The Analytics for Learning (A4L) Network is made up of researchers exploring the measurement of student learning behaviors and strategies in digital learning environments. Learn more: http://analytics4learning.org.

The Improvement Analytics group engages in collaborative data intensive research to help educational organizations improve learning opportunities for all students. Based within SRI Education, the Improvement Analytics group leverages broad domain expertise and diverse methodological approaches to support those working with learners of all types turn promising ideas into improvements. Across multiple partnerships, we have helped educational organizations find actionable insights within complex data sets and develop as well as productively adapt instructional innovations. Learn more: http://improvement-analytics.org.

SRI Education, a division of SRI International, is tackling the most complex issues in education to identify trends, understand outcomes, and guide policy and practice. We work with federal and state agencies, school districts, foundations, nonprofit organizations, and businesses to provide research-based solutions to challenges posed by rapid social, technological and economic change. SRI International is a nonprofit research institute whose innovations have created new industries, extraordinary marketplace value, and lasting benefits to society. Learn more: http://www.sri.com/education.

This material is based upon work supported by the National Science Foundation through grant SMA-1338487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Using Interaction Networks to Identify Unproductive Solution Steps in Multistep Problems

By: Drew Hicks, North Carolina State University
Michael Eagle, Carnegie-Mellon University
Elizabeth Rowe. EdGE at TERC

January 2017
Contents

Abstract ................................................................................................................................................... 1
The Interaction Networks Approach to Trace Data .................................................................................. 1
Interaction Networks and Wheel Spinning .............................................................................................. 2
Interaction Networks and Evidence-Centered Design ............................................................................ 2
   Difficulties of This Approach for Some Domains .............................................................................. 4
Results ................................................................................................................................................... 8
   Results for BOTS .................................................................................................................................... 8
   Results for Quantum Spectre ................................................................................................................. 11
Discussion and Future Work .................................................................................................................. 12
References .............................................................................................................................................. 13
Appendix: Design Pattern ...................................................................................................................... 15
Abstract

Interaction networks are a complex graph structure for transactional student data. They enable developers to use common graph analysis algorithms to uncover insights into both student behavior and system behavior. The data contained in these networks can be used to supplement expert understanding of those domains, allowing student, task, and evidence models to be derived from observed behavior in the systems. However, even for simple problems, these networks can be large and dense, making it difficult for domain experts and educators to gain insights from them and hard to draw conclusions about groups of student paths. This is especially true for systems that are not simple rule-using systems, for example, systems where code is written and then run. To address this issue, we use an abstract state representation to create clusters. This state representation is built for each domain by generalizing multiple similar solutions based on their results. With these representations, we are able to use interaction networks to differentiate problem-solving strategies between groups and identify specific regions of the network which represent unproductive solution steps. We can then present the data in an intuitive, accessible way for educators and domain experts. In this work, we applied the evidence-centered design (ECD) concept of student, task, and evidence models to describe how these improved interaction networks could provide evidence of wheel-spinning behavior in rule-using systems.

The Interaction Networks Approach to Trace Data

Interaction networks, as outlined by Michael Eagle based on previous work with Hint Factory by Tiffany Barnes and John Stamper, are complex network representations of student data in rule-using systems such as intelligent tutoring systems (ITS) and turn- or move-based educational games (Barnes & Stamper, 2008; Eagle, Johnson, & Barnes, 2012). Interaction networks have proven useful for visualization (Johnson, Eagle, Joseph, & Barnes, 2011), analyzing problem solving strategies (Eagle, 2013), and providing automated feedback (Hicks & Peddycord, 2013; Mostafavi, Eagle, & Barnes, 2015).

Interaction networks resemble simple game trees; each vertex of the network represents a unique state configuration in the environment’s state space. Edges between vertices are directed and represent an in-game action or move that would result in a state change, from the first state to the second state. As students progress through an exercise, their actions are tracked and recorded using this structure. After enough data are collected, the result is a detailed representation of successful and unsuccessful problem solving strategies. Unlike in typical game trees, this representation has additional information tracked on each vertex and edge of the structure, including individual student IDs and chance of error or success.
from each state. Using this representation, measurements of “distance” from a goal state can take into account the probability of failure by treating the recorded game data as a Markov decision process (MDP) with transition probabilities equal to the observed transition probabilities in the previously collected data. This property of the interaction network can be used to provide hints (suggesting an action with high transition probability that leads to a goal state) and can be observed by experts to identify areas of difficulty in the problem, where high-probability transitions lead to an irrelevant or erroneous outcome.

**Interaction Networks and Wheel Spinning**

Interaction networks are used to collect and analyze trace data in graphical format, helping identify patterns of successful or unsuccessful behavior within multistep problems with varied solutions. This data representation lends itself to examination of many different constructs, but for this work we focused on how the technique could be applied to detect unproductive solution steps in multistep problems, which may then be used as evidence to identify wheel-spinning behavior.

Wheel-spinning has been examined to a great extent in single-step problems and in the context of the outer loop of an intelligent tutoring system. In their work defining the construct, Gong and Beck (2015) proposed a definition based on mastery learning, as measured using success or failure at consecutive problems on the same topic. Most simply, the student who does not approach mastery despite repeated attempts is said to be “wheel-spinning.” In the context of a large multistep problem, a student could be said to be wheel-spinning if he/she makes continued unproductive solution steps without making measurable progress towards a goal. Interaction networks are well suited to identify this behavior because their data-driven construction allows them to accommodate solutions to multistep problems that experts did not predict. This is important to determining whether or not an action is productive because even steps that do not cause immediate progress toward an expert-known solution may still lie on a solution path discovered by other students.

**Interaction Networks and Evidence-Centered Design**

Evidence-centered design is useful for approaching this construct. The ECD framework has three main components or models. The student model describes the attributes of the student that the researcher intends to measure. The most common application of interaction networks is producing hints for problem states automatically based on student behavior, essentially using student behavior and analysis of the resulting interaction network to produce a data-driven task model for new examples in a family of related
tasks. New students’ behavior within this structure—the trace data of the paths traversed within the interaction network—constitutes an evidence model where students’ actions take them into or out of regions that exhibit different characteristics. In particular, elements of the graph structure are used to identify difficult or dangerous areas of the problem where a student may become stuck, thus exhibiting wheel-spinning behavior. Using interaction networks, we can identify these regions for simple rule-using systems and characterize when students enter or exit them.

Our student model is composed precisely of the attribute of the student we want to measure. In this case, we would like to use interaction networks to measure wheel-spinning behavior, where a student continues to act within the system but does not make progress toward the goal.

In our case, the development of a task model for an individual problem first requires collection of student data. The interaction network approach we use was detailed by Eagle, Rowe, et al. (2015) and Mostafavi et al. (2015). In general, the task model for a particular problem is generated from historical data. This model outlines all the moves that students have made as well as their success in reaching a correct goal for that problem. Every student (both students in the historical data and those currently under investigation) is thus characterized by a unique sequence of actions or path through the graph. In both systems described in this paper, BOTS and Quantum Spectre, an individual student within a single problem walks a path from the start state of the problem through various parts of the graph, which may or may not terminate in a goal state. The properties of each student’s path are logged, including the actions taken and number of actions taken.

For our evidence model, we examine the properties of the paths through the graph that students create. Because our construct of interest for this paper is wheel-spinning, we must be able to directly compare individual students’ paths with known successful paths to identify which steps in these paths are productive, that is, which advance toward a solution. Wheel-spinning is characterized not by single unproductive actions but by a pattern of such actions, representing repeated but unsuccessful attempts to progress. To show evidence of wheel-spinning, we must be able to identify these patterns from the evidence model. With graph clustering methods, we can identify regions where further actions do not progress toward any observed solution, and specific bridge actions must be taken to escape the region. Actions within these regions therefore fit the description of wheel-spinning for multistep problems because students may be making technically correct moves within the system but are not making progress toward the goal until the bridge action to move them out of the region is taken.

In a simple interaction network where each action makes an immediate change to the state, the evidence model can be very simple. Actions can be assigned scores on the basis of how much closer they move the user to the goal, and unproductive actions can be easily identified as individual actions that do not lie on a correct solution path or that lie on a correct solution path but have a low observed likelihood of
leading to that path. This is sufficient for a simple definition of wheel-spinning based on sequences of unproductive actions without reaching a goal (or closer) state, and indeed such paths are commonly found in interaction networks and are easy to visually identify.

Working with the simple rule-using Deep Thought tutoring system, we used the Girvan-Newman betweenness clustering algorithm on the interaction network to produce regions of moves that are highly connected to one another. Regions with no goal states are identified as “bad regions,” and a student making moves within them must take a bridge to another region to reach a goal. We used a two-tailed chi-squared test to measure whether students in different groups visited such regions with differing frequencies and were able to find such differences between students who received automatically generated hints and students who did not (Eagle & Barnes, 2014).

**Difficulties of This Approach for Some Domains**

For more complex systems, the interaction network has structural properties that make existing techniques more difficult. Although we would like to be able to identify such constructs as wheel-spinning, which have definitions in other systems, these definitions sometimes fail when applied to multistep problems, particularly in rule-using systems whose actions do not always directly affect the state. This problem stems from the domains these interaction networks were originally developed with. Systems like Deep Thought can be characterized as rule using and can be compared with turn-based single-player games. The only thing that changes the state is a user action, and that user action immediately changes the state. Our work extends the existing work with interaction networks to two kinds of systems that do not perfectly adhere to this assumption.

First, in both systems examined in this paper, BOTS (a game designed to teach programming fundamentals) and Quantum Spectre (a game designed to teach the basics of optics as well as underlying mathematical concepts), individual actions may not immediately affect the state even if not erroneous. This frustrates the formulation of an *evidence model* as used in earlier systems because individual actions may appear on their own to be unproductive but not actually be so, and a sequence of many seemingly unproductive actions may be useful or even required for a solution. For example, in BOTS, adding an additional line of code to the robot’s program will not immediately affect the state until the program is run, and extrapolating the result (as if the robot’s program were executed immediately) will give an inaccurate picture of problem solving. Consider a player who is halfway through writing a function, for example. Similarly, in Quantum Spectre, mirrors may be placed in a location or orientation that does not immediately affect the state representation but will later. Separating these behaviors, which may be in preparation for something later or undirected “pottering” behaviors between strategic actions, is difficult in such a system but easier in a system where actions immediately affect the state. Since this is not the
case for BOTS or Quantum Spectre, a direct measurement of constructs such as wheel-spinning behavior using simple unproductive actions would not be appropriate.

Second, in BOTS, users provide an additional level of information relative to a system such as Deep Thought. In BOTS, users choose when to execute their program, confirming that they wish to attempt to solve the problem, or verify that their program correctly accomplishes some self-selected subgoal. Users are telling us that these states are more important to their problem-solving than surrounding states, but the representation as is does not capture this; even if “Run” were treated as an action, additional work would be needed to confirm that states immediately preceding Run steps are different is a meaningful way from those that do not, as this information is not built into the interaction network itself.

**Figure 1: Equivalent states in Quantum Spectre. These two solutions (and those similar solutions in between) demonstrate the same understanding.**

Finally, in both systems, there are many functionally equivalent states that are reached by fundamentally different sequences of actions. Unlike in other systems where this claim may be made about incorrect states (although the manner of incorrectness can still be important), in Quantum Spectre this claim can be made for correct states. Consider the case where a pair of mirrors on parallel rows of the board is used. These two mirrors can be transposed to many other positions on those rows, without changing the correctness of the solution (Figure 1). The current state representation of the interaction network fails to capture this similarity and as a result would treat students solving the problem in extremely similar ways as having entirely different strategies. The resulting interaction network will neglect this similarity when used to generate hints for these students, and will fail to make the similarity apparent to researchers analyzing problem solving.
To solve these, we must make changes to our evidence model to take into account the potential for unproductive actions (as measured naively) to be useful and to compare equivalent states properly when considering groups of states. To accomplish this, we build clusters of equivalent states considering the most recent output and use these clusters rather than the constituent states to measure whether actions are unproductive (Figure 2). By making these changes, we add another feature to our evidence model, which permits us to discover these types of regions in our interaction networks for these more complex systems.

Figure 2: Corresponding section of interaction network from Figure 1 showing the problem. Naive analysis of the interaction network will draw the false conclusion that the Error state (highlighted in red) is as likely as the other transitions. However, all other transitions represent equivalent strategy and understanding.

Using Contextual Clustering to Enable Interactive Networks for More Complex Rule-Using Systems

Both the environments covered in this paper exhibit problems similar to traditional application of the interaction network technique. However, in both domains, we can use contextual information from the game to cluster related states; in this work, we call this contextual clustering. This method involves developing a derivative state representation that contains all the information from the game state, plus
information used to determine similarity. All the vertices of the interaction network, which contain the same similarity state, are then clustered together. We can build our interaction network over these clusters rather than the underlying states by treating the cluster as a single vertex and condensing all the edges entering or leaving the cluster to single edges. This helps address some of the difficulties previously mentioned.

*Contextual Clustering in BOTS*

In BOTS, we use two primary elements to develop the similarity state: (1) clustering based on compile states rather than individual actions and (2) clustering based on robot movement rather than raw program contents.

We addressed the issue of delayed impact by grouping states according to the next compile state in the player’s log. This serves two purposes. First, it characterizes each step toward a compile as relating to that compile. Second, this allows paths that eventually correspond to the same compile state to be compared with the knowledge that the outcome of both paths, regardless of how different they are, is the same compile state.

We further characterized these compile states by robot movement rather than the raw program contents. In essence, a program that used loops and functions could thus be directly with to a program that accomplished the same movement but did not use loops or functions. Both paths would be contained within the vertex, with the loop/function-using solution taking fewer actions.

*Contextual Clustering in Quantum Spectre*

In Quantum Spectre, we cluster based on laser shape rather than raw state contents. Figure 1 is an example of a situation where this could be applied. To address the problem of delayed impact, we characterize each state according to the shape of the lasers present in the puzzle rather than by the position of the mirrors. Each laser can be described by a number of sequential angle bends or target/object intersections. For example, both states in Figure 1 would be characterized by “Blue +90, +90, Goal” and thus would be clustered together in the resulting interaction network. As with BOTS, states with the same characterization can be grouped together in a single high-level vertex and the edges entering and exiting the cluster condensed. As a result of this generalization, the interaction network in Figure 2 would effectively be reduced to that shown in Figure 3. Again, assuming an equal number of students had taken each approach, the correct strategy on the left would be, at first glance, four times more likely than the erroneous strategy on the right. This enables us to use the interaction network as a Markov Decision Process to suggest policy to users as low-level hints, whereas with the lower level network in Figure 2 we would not have been able to select an appropriate action.
Results

We have already applied these techniques to the environments of BOTS and Quantum Spectre. Here, we present the results and describe how this approach improved our ability to use interaction networks to analyze player behavior and our ability to distinguish unproductive solution steps and graph regions made up of such steps.

Results for BOTS

Our goal was to make BOTS interaction networks suitable for hint generation by applying some method of generalization to related but not identical game states. In this interaction network, “hintable states” are an ancestor of a goal state along a directed path, indicating that a student has reached the goal from the state in question so that a hint could be provided based on that student's actions using the Hint Factory method. The results from that previous study (Peddycord, Hicks, & Barnes 2014) are presented for reference in Table 1.
In our original work with these data, we were primarily concerned with being able to offer low-level hints more frequently, and our primary measure of success was the density of nodes visited. As is evident from Table 1, there was an across-the-board reduction in both the overall size of networks and the proportion of hintable states, indicating that after collecting a corpus of this size, most states would be hintable and in the worst case the distance from a hintable state would be small (Peddycord, Hicks, & Barnes 2014).

Additionally, after creating the interaction network using this reduced state space, we were able to analyze the networks by hand and encode several different types of interactions where we had previously been unable to make such determinations. From the data collected, we divided the set of observed states into classes. First among these is the start state in which the problem begins. By definition, every player’s path must begin at this state. Next is the set of goal states in which all buttons on the stage are pressed. These are reported by the game as correct solutions. Any complete solution, by definition, ends at one such state. Among states that are neither start nor goal states, there are three important classifications: intermediate states (states a robot moves through during a correct solution), unproductive states (states that do not lie on a known solution path but do not indicate erroneous input), and error states (states that result from illegal input, like attempting to move the robot out of bounds). On the basis of these types of states, we classified several types of transitions represented in the data set, useful for analyzing both the problems themselves and the behaviors of users within them.

Unlike in the unclustered interaction network, we can now identify unproductive transitions by their lack of movement toward a goal state. Such an unproductive transition occurs when a student moves from an unproductive state to another unproductive state. We may consider a student to be engaging in wheel-spinning behavior when consecutive examples of such transitions have occurred, indicating multiple

<table>
<thead>
<tr>
<th>Name</th>
<th>#Students</th>
<th>Codes states</th>
<th>World states</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hint</td>
<td>All</td>
</tr>
<tr>
<td>Tutorial 1</td>
<td>125</td>
<td>89</td>
<td>162</td>
</tr>
<tr>
<td>Tutorial 2</td>
<td>118</td>
<td>36</td>
<td>50</td>
</tr>
<tr>
<td>Tutorial 3</td>
<td>117</td>
<td>130</td>
<td>210</td>
</tr>
<tr>
<td>Tutorial 4</td>
<td>114</td>
<td>137</td>
<td>225</td>
</tr>
<tr>
<td>Tutorial 5</td>
<td>109</td>
<td>75</td>
<td>106</td>
</tr>
<tr>
<td>Challenge 1</td>
<td>107</td>
<td>348</td>
<td>560</td>
</tr>
<tr>
<td>Challenge 2</td>
<td>98</td>
<td>201</td>
<td>431</td>
</tr>
<tr>
<td>Tutorial 6</td>
<td>90</td>
<td>107</td>
<td>143</td>
</tr>
<tr>
<td>Challenge 3</td>
<td>89</td>
<td>192</td>
<td>278</td>
</tr>
<tr>
<td>Challenge 4</td>
<td>86</td>
<td>137</td>
<td>208</td>
</tr>
<tr>
<td>Tutorial 7</td>
<td>76</td>
<td>206</td>
<td>383</td>
</tr>
<tr>
<td>Tutorial 8</td>
<td>68</td>
<td>112</td>
<td>134</td>
</tr>
<tr>
<td>Challenge 5</td>
<td>34</td>
<td>17</td>
<td>27</td>
</tr>
</tbody>
</table>
unproductive steps. This evidence could not be reliably derived from the unclustered interaction network. The precise number of transitions needed to characterize this behavior should be determined on the basis of the general length of a correct solution in this environment.

In addition to correctly identifying unproductive transitions for our evidence model, we were able to characterize other types of transitions that previously would have been characterized incorrectly as simple productive or unproductive transitions.

A subgoal transition from a start or intermediate state to another intermediate state occurs when a student moves the robot to an intermediate state rather than directly to the goal. Since players run their programs to produce output, we speculated that these transitions may represent (or be useful for identifying) subgoals such as moving a box onto a specific switch. After accomplishing that task, the user then appends to the program, moving toward a new objective, until he/she reaches a goal state.

A correction transition from an error state to an intermediate or goal state occurs when a student makes and then corrects a mistake. These are especially useful for providing hints, because we can offer hints based on the type of mistake.

A rethinking transition, from any state back to the start state, occurs when rather than appending to the program as in a subgoal transition, the user deletes part or the entire program and then moves toward a new goal. As a result, the first state is unrelated to the next state the player reaches. Offering this state as a hint would most likely not help guide a different user but could help identify when a student is having trouble.

An error transition, from an intermediate state to an error state, corresponds to a program that walks the robot out of bounds, into an object, or other similar errors. While we disregarded these as hints, this type of transition may still be useful. The last legal output before the error could be a valuable state identifying what led users to commit the error in question. Repeated cases of such errors are potentially additional evidence of wheel-spinning behavior.
Using Interaction Networks to Identify Unproductive Solution Steps in Multistep Problems

Table 2: Interaction Networks in Puzzles 14-23

<table>
<thead>
<tr>
<th>Game Level</th>
<th># Players</th>
<th>Total # Moves</th>
<th># Network Edges</th>
<th># Unique States</th>
<th># Laser Shapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>479</td>
<td>3003</td>
<td>462</td>
<td>164</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>473</td>
<td>3866</td>
<td>1009</td>
<td>484</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>462</td>
<td>3218</td>
<td>761</td>
<td>446</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>454</td>
<td>10878</td>
<td>1899</td>
<td>1067</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>439</td>
<td>10314</td>
<td>3458</td>
<td>1800</td>
<td>22</td>
</tr>
<tr>
<td>19</td>
<td>416</td>
<td>15389</td>
<td>7093</td>
<td>4550</td>
<td>330</td>
</tr>
<tr>
<td>20</td>
<td>384</td>
<td>10778</td>
<td>4947</td>
<td>2391</td>
<td>264</td>
</tr>
<tr>
<td>21</td>
<td>349</td>
<td>23080</td>
<td>13919</td>
<td>6261</td>
<td>696</td>
</tr>
<tr>
<td>22</td>
<td>282</td>
<td>3697</td>
<td>1500</td>
<td>1017</td>
<td>146</td>
</tr>
<tr>
<td>23</td>
<td>271</td>
<td>10529</td>
<td>6154</td>
<td>4138</td>
<td>364</td>
</tr>
</tbody>
</table>

Results for Quantum Spectre

We have also used the above approach in work with the Quantum Spectre game. With Quantum Spectre, we had an additional goal: to make the interaction network more usable for developers and to use it to automatically encode similar types of errors without requiring expert assistance. The relevant results from that work (Eagle & Brown, et. al, 2015) are summarized here for puzzles in the first Zone of the Quantum Spectre game.

Using this representation, the game designers were able not only to automate classification of similar basic errors involving movements and rotations that are not present on known solution paths, but also to use the much-reduced interaction network to identify a new class of conceptual error not necessarily associated with an incorrect movement or rotation but with a misunderstanding of the game’s rules. These puzzle errors were shown to account for more than 35% of errors on two specific puzzles, and identification of these puzzle errors has assisted significantly in discovering student misconceptions in those puzzles. Similar to our work with BOTS, these puzzle errors represent erroneous input unlike the other errors that are technically correct moves but that show misunderstanding of the underlying mathematics and science concepts.

In this environment, we were able to use the approach map technique (Eagle & Barnes, 2014) to identify regions of unproductive actions and error actions that the experts coding the networks referred to as confusion regions (Eagle & Brown, et. al 2015). These regions were characterized by few students solving the problem from the regions and correspondingly unproductive moves-transitions observed within
them. These regions were used by puzzle designers to identify pieces of content most likely to elicit this behavior from students.

**Discussion and Future Work**

In both environments, contextual clustering enabled us to construct a better interaction network, which may be used as an evidence model for identifying unproductive actions. The improved interaction networks helped developers of systems that resemble, but are not strictly, rule-using systems to detect successful strategies to use to deploy automated feedback or to detect unsuccessful strategies with different actions but common outcomes. One limitation of this technique is that it is domain specific; for each environment, the state similarity representation must be developed by an expert who understands which elements of the state are important on their own and which are important only in context. Once this state similarity representation is developed, however, the interaction network technique can be used on data from that environment as though the environment were a rule-using system, and developers can dive into the lower level state representation as needed to derive higher level hints or examine different approaches to the same underlying strategy.

Another limitation of this approach is that both systems covered in this paper are turn based or action based, as opposed to real-time systems. To apply interaction networks to an environment in which students act in real time (such as the game Impulse) would require a more complex generalization of the state space than detailed here.

The current extent to which interaction networks can be used to identify unproductive solution steps and wheel-spinning behavior depends strongly on the quality of the historical data used to build the network. The interaction networks can produce unexpected results when the data used do not accurately describe the state space—for example, a correct solution that no previous student had encountered. Additionally, the current model does not take into account student mastery. The technique could be improved by considering student mastery when predicting wheel-spinning behavior on previously unobserved paths since a more skilled student is somewhat more likely to be exploring a new solution by “off-roading” rather than making unproductive steps.
References


Appendix: Design Pattern

Authors

<table>
<thead>
<tr>
<th>First Name</th>
<th>Drew</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Name</td>
<td>Hicks</td>
</tr>
<tr>
<td>Affiliation</td>
<td>NC State University</td>
</tr>
<tr>
<td>E-Mail</td>
<td><a href="mailto:aghicks3@ncsu.edu">aghicks3@ncsu.edu</a></td>
</tr>
</tbody>
</table>

Overview

Summary

- Wheel-spinning behavior, defined as in previous work by Gong and Beck as repeated unproductive attempts at solving a problem without demonstrating mastery. Initially, this was defined as the absence of mastery, demonstrated by solving three consecutive problems over an increasing number of attempts. Specifically, we wanted to detect this behavior using interaction networks, a method that has been used in other systems.
- Data used were from two game-based learning systems: BOTS, a game designed to teach programming fundamentals, and Quantum Spectre, a game designed to teach the basics of optics as well as underlying mathematical concepts.
- The data collected by these systems includes player actions in each environment in terms of the game’s objects (blocks of code or optical devices). In both environments, players’ interactions with these objects are collected. In BOTS, players also may choose to run their program, and this event is also collected.


### Rationale
- Interaction networks provide a data structure that can be used to generate a task model for a given problem from historical data. This model is particularly useful for differentiating problem-solving approaches and identifying actions that are extremely unlikely to lead to a goal. These actions characterize wheel-spinning behavior in the context of a multistep problem, wherein a student makes multiple actions that affect the state but which do not affect progress toward the goal of solving the problem.
- Previous work has shown that wheel-spinning behavior is connected to “gaming the system” behaviors, indicating disengagement with the material and a need for teachers or the system to intervene.
- Identifying wheel-spinning behavior may be useful for indicating where teachers should intervene in a student’s use of an intelligent tutoring system.
- Wheel-spinning behavior from students may indicate places where the content of the intelligent tutoring system needs to be improved.

### Student Model

#### Focal Construct
- **Wheel-Spinning**
  - In context of a multistep problem, a student is making state-changing actions, but these actions do not affect progress toward the goal of solving the problem.

#### Additional knowledge, skills, and abilities
- Prior knowledge of system/interface. Sandboxing or experimenting with the interface can resemble wheel-spinning.
- Face validity of the system. Students’ perception of the educational value or challenge of the system may affect wheel-spinning.

### Task Model

#### Characteristic Features of the Task
- Multistep problem-solving task, where the student makes a series of discrete moves that alter the state.
- Data logged during task include:
  - Each “move” made by student
  - (In BOTS) Student attempt to run code

#### Variable Features of the Task
- Difficulty of the specific problem
- Presence/relevance of automated feedback during task
- Accuracy of interaction network model at identifying productive and unproductive actions for the specific problem

### Potential Task Products
- Sequences of student moves
- Individual student moves

### Evidence Model

#### Potential Observations
- Unproductive actions
  - Individual actions that do not lie on a correct solution path
  - Individual actions that lie on a correct solution path but have low observed likelihood of leading to that path
- Unproductive solution paths
  - Paths of moves along which the distance from the goal state does not improve

#### Potential Frameworks
- Use previously collected student data, along with a fitness algorithm such as the Bellman Backup algorithm commonly used with interaction networks to provide each move a fitness value and identify unproductive solution paths
- If student moves are sparse, use contextual clustering to group similar states with similar output, so that this comparison may be used more effectively