Milestone 5 Progress Evaluation

Automatic Program Feedback

1.

Team Members:

Andre Leone  Ryan Hartman  Calvin Winget
aleone2013@my.fit.edu  rhartman2014@my.fit.edu  cwinget2014@my.fit.edu

Faculty sponsors:

Dr. Philip Chan  Dr. Ryan Stansifer
pck@cs.fit.edu  ryan@cs.fit.edu
### 2. Progress of current milestone (progress matrix)

<table>
<thead>
<tr>
<th>Task</th>
<th>Completion %</th>
<th>Andre Leone</th>
<th>Ryan Hartman</th>
<th>Calvin Winget</th>
<th>To do</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Start using the parallel processing and setup the code to run on a fast environment</td>
<td>100%</td>
<td>75%</td>
<td>25%</td>
<td>0%</td>
<td>none</td>
</tr>
<tr>
<td>2. Run experiment only using get/set/add/remove frequency</td>
<td>100%</td>
<td>15%</td>
<td>15%</td>
<td>70%</td>
<td>none</td>
</tr>
<tr>
<td>3. Run experiment where file size is included as part of the distance measurement</td>
<td>100%</td>
<td>50%</td>
<td>50%</td>
<td>0%</td>
<td>none</td>
</tr>
<tr>
<td>4. Run experiment where whether or not the program timed out is included in the distance measurement</td>
<td>80%</td>
<td>60%</td>
<td>40%</td>
<td>0%</td>
<td>All necessary implementation is in place to include this we just need to add this as a feature</td>
</tr>
<tr>
<td>5. Brainstorm possible updates that can be done to improve the feedback results</td>
<td>100%</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
<td></td>
</tr>
</tbody>
</table>
3. Discussion of Tasks

- Task 1:
  - One of the main goals for this milestone was to improve our implementations to allow experiments to be ran more quickly. We now have a parallel implementation of how we generate feedback and have seen some performance increase. We are also working on adding a parallel version of recording behaviors since this task also lends itself well to parallelization. To further help with runtimes, Ryan now has access to Blueshark and has been using it to run experiments which has lowered the run time as well.

- Task 2:
  - After talking with Dr. Stansifer for the last milestone we came up with the idea of ignoring array state completely and just using frequency counts of get/set/add/remove to compare programs. To do this comparison we construct a feature vector for each program which a dimension corresponding to one of get/set/add/remove. With the feature vector constructed for each program we then compute the cosine distance between two programs and use this instead of edit distance. With just using the frequencies, our accuracy decreased to 19/28 where it was 20/28 when just using edit distance. These numbers are close but the frequency method had less ties than the edit distance, but more incorrect matches. After looking at the matrix to see what went wrong it appears that there is less room for improvement in the frequency method compared to the edit distance. The one benefit this version has is that it runs much quicker than the edit distance.

- Task 3:
  - One of the new metrics that we added to work in conjunction with edit distance is what we call “line distance”. The line distance is the absolute value of the number of non-commenting lines of code in a submitted program minus the number of non-commenting lines of code in the reference program being compared. We expected this to fix the situation where empty programs matched with reference programs that were not empty but didn’t have any output. It turns out that introducing the line distance did not increase the number of correct matches and we are looking into why for next milestone.
  
  - In order to combine line distance with the preexisting edit distance, we changed how the generate feedback portion of the code was implemented. The code now stores each distance metric (edit distance, line distance, and get/set/add/remove distance) in a vector. This vector represents the distance between a single submitted program and a single reference program. The set of vectors for each submitted program is then normalized to contain values between 0 and 1. Next, each component of the vector is multiplied by its corresponding weight. Lastly, the magnitude of the vector is taken to represent the distance between the single submitted program and a reference program.
• Task 4:
  ○ Using the technique stated in the second part of task three, we are able to easily add on additional distance metrics. Our current implementation is able to recognize time outs for the program. We just now need to make a note that these occur and incorporate it as a part of the vector that we create and use it for distance.

• Task 5:
  ○ Our main ideas for further improving the program relate to including additional information in the final distance vectors we are constructing. This can include things such as number of test cases passed, finishing the inclusion of timeouts and other metrics related to the running of the program such as runtime or memory used. These metrics should further allow us to distinguish programs from each other.

4. Discussion of contributions

• Andre Leone:
  ○ My main contributions this milestone where in tasks 1, 3, and 4. For task one, we implemented parallel code and and then I was able to run experiments testing the speed improvements. We were not as successful as we had hoped to be in this area due to Python’s implementation of using a single processor. For tasks 3 and 4 I made the necessary code changes to allow for an infinite number of distance metrics to be taken into consideration when computing the distance between a submitted and reference program. This involved, magnitude of vectors, normalizations, and a few other code changes to ensure that we had the right information available. In addition to making the code extendable to consider multiple distance metrics, I implemented the Line Distance metric that computes the number of lines in a .java file, excluding comments and blank space. We can now in the future add more distance metrics and weight them however we would like.

• Ryan Hartman:
  ○ My main contributions related to tasks 1 through 4. With task one, I worked with Andre to figure out why we were having issues running our software on a Linux environment and also how to do parallel processing in Python. For task 2, I worked with Calvin to help get the cosine distance for the feature vectors implemented properly so that we could use the frequencies to compare between programs. For task 3, Andre and I worked on making sure that the code necessary was in place for us to be able to add any other metrics and have them be taken into account for the program distance. This will allow us to easily finish task 4 completely for the next milestone as well as add other metrics.

• Calvin Winget:
  ○ My main contributions were with tasks 2 and 4. In task 2 I implemented the experiment to match programs based on their get/set/add/remove frequency, and
with Andre and Ryan's help I got it connected to the rest of the code. I ran the experiment and then using the generated matrix tried to understand where this method went wrong and how it could be improved.

5. Plan for the next Milestone (task matrix)

<table>
<thead>
<tr>
<th>Task</th>
<th>Andre Leone</th>
<th>Ryan Hartman</th>
<th>Calvin Winget</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Finish including timeout data</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>2. Add test case data as well as other metrics to the distance calculation</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>3. Rerun experiments using the new metrics added to the distance calculation</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>4. Construct a demo of how the whole process works</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
</tbody>
</table>

6. Discussion of each planned task for the next milestone

- **Task 1:**
  - As mentioned above, the majority of the code is in place to include timeout data we just need to finish off the final bit of the implementation.

- **Task 2:**
  - Currently, test case results are not incorporated anywhere in the program. Adding these will further allow us to differentiate programs to break any possible ties and should help to improve our accuracy. We will also work to include other metrics about the program to help with this process.

- **Task 3:**
  - Once the changes are finalized we need to rerun our experiments to see how our results change as we add and remove features. This will be an ongoing process throughout the milestone as we work to improve accuracy.

- **Task 4:**
  - We also want to construct a demo to be able to show how the software works and the steps that would be necessary for other people to use this work to help assist with giving feedback to their students.
7. Sponsor feedback on each task for the current milestone

- Task 1:

- Task 2:

- Task 3:

- Task 4:

- Task 5:

Sponsor Signature: _________________________________ Date: ________________
8. Sponsor Evaluation

- Sponsor: Detach and return this page to Dr. Shoaff
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or right down a real number between 0 and 10)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5.5</th>
<th>6</th>
<th>6.5</th>
<th>7</th>
<th>7.5</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andre Leone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>5.5</td>
<td>6</td>
<td>6.5</td>
<td>7</td>
<td>7.5</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
</tr>
<tr>
<td>Calvin Winget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>5.5</td>
<td>6</td>
<td>6.5</td>
<td>7</td>
<td>7.5</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
</tr>
<tr>
<td>Ryan Hartman</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>5.5</td>
<td>6</td>
<td>6.5</td>
<td>7</td>
<td>7.5</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
</tr>
</tbody>
</table>

- Sponsor Signature: _______________________________ Date: ______________