**4. Evaluation**

**4.1 Solvable Problems**

To determine the level of quality of the final Go problem solver created, a series of evaluation techniques were carried out. As the original requirements required a problem solver that can solve life and death problems, the most deterministic approach of measuring the success of implementation was through finding life and death problems that the finalized program could successfully complete. Also found were several unsolvable problems that the program could be extended to solve in future through added heuristics. Both types of problems identified during evaluation have been included in "*Solvable Problems*" and "*Unsolvable Problems*" directories that come with the problem solver. These directories are automatically displayed when the "*Load Problem*" option is chosen under the file menu, hence allowing user access to the problems found during evaluation.

**Solvable Problems With Brute Force**

Using brute force alone, several life and death problems were able to be solved by the problem solver.

Add examples of solvable brute force problems (Include difficulty estimates/time to solve)

**Solvable Problems With Heuristics**

Once brute force was extended through the use of heuristics, as described in *Section 3.6*, the program was able to solve several new, more specialised problems.

Add examples of solvable heuristic problems (Include difficulty estimates/time to solve)

**Not Yet Solvable Problems**

Aside from solvable problems, there are also an array of unsolvable problems that the problem solver could be extended to be able to solve in future. *Section 3.7* describes several heuristics that could be added with further development, and the following unsolvable problems relate to these specific heuristics.

Add examples of not yet solvable problems (Include difficulty estimates)

The original requirements (see *Section 3.2*) desired a problem solver which could solve 15 Kyu problems, being of around a median difficulty, and a success rate of 80% in dealing with this problems. The collection of solvable problems found of varying difficulty prove the success of these requirements. Whilst it is hard to determine the percentage of the program's success rate, as the number of life and death problems is virtually endless, the large collection of solvable problems found does prove that the combination of artificial intelligence and heuristics has enabled the creation of a capable problem solver which will allow human players to work upon their strategy and play through life and death problems outwith a full game of Go.

**4.2 Subject Testing**

Whilst evaluation through the finding of solvable problems was carried out by the team itself, testing using subjects outwith the team was also performed and provided valuable feedback. A testing document was prepared (see *Appendix [insert number here]*) and was given to participating test subjects. The document asked the tester to input a set problem given, or their own problem, into the program and attempt to run it in a mode of their choice: AI vs. human, AI vs. AI or human vs. human. Following this, the testers were asked to write down their feedback on the general problem solver, as well at its two modes: problem creation and competitive play. Before beginning testing, participant's consent was acquired, they were given the team's contact details and they were also informed them they could withdraw at any point during testing. Once testing was complete, the testers participated in a discussion focused on the evaluation. Hence, subject testing was carried out as required by ethics procedures.

**General Interface Testing**

Through the evaluation conducted by real life subjects, a wide range of feedback was supplied concerning the general interface of the problem solver itself, including graphical representation and layout.

Insert interface feedback/evaluation

**Problem Creation Mode Testing**

As described above, participants were asked to create a problem within the problem solver during problem creation mode. The problem itself was simple and required a range of white and black stones, as well as user set objective and bounds. User attempts at problem creation provided a wide range of feedback relating to the creation of problems within the problem solver.

Insert problem creation mode feedback/evaluation

**Competitive Play Mode Testing**

The testing subjects were also asked to play through the problem they had previously created using competitive play mode. They were asked to use any specific mode they wished (human vs. human, AI vs. human, etc.) and provide written feedback on their selection and thoughts on play through following their use of competitive mode.

Insert competitive mode feedback/evaluation

**4.3 Other Testing**

The most valuable evaluation feedback for the Go problem solver was provided through the finding of solvable problems and user testing, as described above, but other miscellaneous testing also helped to prove the program's degree of success as well as improve it during actual implementation.

Insert description of JUnit tests/attempted bug finding, etc.

**Testing Overview**

Overall, a wide variation of testing procedures were carried out using the program, both during development and following its completed implementation.

Describe outcome of solvable problems (number of found problems)/highlight recurring user feedback/reiteration of other testing