Matt Dayley Jacob Meline Joseph Richardson

(Informal) Final Project Proposal

We’ve planned out a few things, but before we get too far into actually coding this, we wanted to make sure that it is an acceptable project, or, if not quite, to know what sort of thing you’d expect to make it become acceptable.

We plan on assessing strategies for the game of Risk. There are many, many published strategies for Risk, many of which are effective in different circumstances. We wish to

1. Implements the game of RISK, which will be able to accept a strategy for each team involved
   1. We would create a simple UI (maybe a GUI, maybe not) to test the game, and a human-controlled strategy option, but neither would be used after testing
2. Implement a wide variety of strategies
3. Run many games of risk combining different strategies in order to see which do well against which, whether there is a “best” or whether there are cycles in the domination graph, and see whether the good strategies are stably good or whether they are high-risk and work only sometimes. We can also assess other things such as the average number of rounds for given strategies, whether it matters who plays first, etc to see if there are other advantages.

The parallelization would be handled in part 3, by running many games at once. Even though the gameplay itself is more or less fundamentally serial, there are so many combinations of strategies (the number of players is variable), and so many repetitions are required for good statistics, that the advantage of parallelizing are considerable, and getting enough data without parallelism on the USU high-performance computer would require absurd amounts of time.

There are some variations we can use if more parallelism is desired, though from the perspective of speed rather than learning is it likely more effective not to implement all of them:

* We can also vary the rules of risk by requiring that all players pre-plan their move and submit it all-at-once each round, rather than modifying what to do mid-turn (for example, if their armies are doing badly). Were this done, players could compute their move in parallel before submitting it, though it would still have to run in serial.
* There is also the possibility of using parallelism to implement “look ahead” in strategies that require it. Slave processors can be available made for that purpose.
* Top strategies can be combined using crossover techniques from genetic algorithms, to see if a superior strategy can be thusly formed. This differs from a normal genetic algorithm, however, in that it does not begin randomly and uses only crossover, and never mutation. Specific functions of the “Strategy Interface” would be swapped or re-combined.

Anyhow, these are our ideas. Once we know if the main idea is acceptable, and whether the optional variations would be required or not, we can go ahead with our planning past the brainstorming stage and submit a new proposal with what exactly we plan to do.