README

Patrick Yeo Ho Yoon

December 18, 2012

Contents

#+TITLE: CIS399-03 Final Project Writeup: HASKELL ANTS BOT

1 How to run code

./make.sh to build the Ants. ./run$_{\text{gamebrowser}}$.sh to simulate one game and display the replay on web browser.

2 Source Code

- BFS.hs : BFS and modified DFS code for exploration
- Ants.hs : General ants related function
- MyAnts.hs : Contains DoTurn function which is executed in every turn.
- AStar.hs : Has A* algorithm implementation
- alphabeta.hs(unfinished, not included) : Alpha beta pruning implementation in minimax algorithm for combat

3 External Packages Used

I used Haskell Starter Package(http://aichallenge.org/starter_packages.php) to get started and used PSQueue from Hackage (http://hackage.haskell.org/packages/archive/PSQueue/ PSQueue.html) for priority queue in A* search.

4 Overall Description of the project

I implemented A* Search, Breadth-First-Search(BFS), Depth-First-Search(DFS) for path-finding to foods and exploration, and Alpha-Beta Pruning in Minimax tree for combat strategy.

4.1 Exploration

At first I used BFS to find the first unexplored Point and move the ant toward it, but somehow it was not effective for diffusion ants to unexplored regions. (The current bot uses this strategy)

Therefore I used a strategy inspired by the exploration strategy of xantis(http://xathis.com/posts/ai-challenge-2011-ants.html) who is the winner of the aichallenge 2011. I assigned an exploreValue to each tile on the map which is increased by one at the beginning of each turn. In the explore stage, if a tile is reachable within 10 steps by free ants that are not assigned with any of the combat or food order, its exploreValue is reset to 0. We do this during the modified DFS which stops deepening at the detph(step) 11. Once we finish DFS upto 11 steps, we sum up all exploreValues of tiles reachable within 10 steps from each left, right, up, down direction and pick the direction with the maximum explore value. This was hard to implement in Haskell as I needed to update the exploreValue in a real time and the World data structure that keeps MetaTile data which ontains ExploreValue data is immutable. I haven't finished implementing this yet.

4.2 Pathfinding to Foods

I used A* search which uses cost+husristic for deciding which path to take. I used manhattan distance as the heuristic function as it never overestimates the actual cost(x+y distance).

4.3 Combat Strategy

I first find the group of my own ants and enemy ants that can be involved in the combat. (TODO: Need to figure out how to find this set) Then, I use minimax algorithm with alpha beta pruning to figure out the combat strategy for this local set of game. This portion of the Bot is still work in progress. The WIP can be found in alphabeta.hs.

5 Recursion?

The game loop is basically implemented using recursion. I used the basic framework from the Starter Package, but modified the data types and added helper functions to accomodate by need for pathfinding for foods and enemy hills. It took a while to understand the game logic and how the looping in implemented in Haskell using loops.

AStar and BFS uses recursion. Also updateTurn functions uses recursion to add orders to Turn data recursively. I also tried to implement minimax algorithm using haskell, but it turned out to be hard because I haven't figure out the evalution function yet (I adopted an idea from xathis's bot using number of dead enemy ants and dead my ants, but the function itself needed implemetation of defining local set of ants involved in the combat which I haven't figure out how to implement in Haskell..)

6 Takeaways

I have implemented ants bot in Python and it was much easier to implement various strategy because I could keep the state in the program and didn't need to pass around immutation data strauctures in recursion. But I learned a lot about using recursion to implement recursive algorithm involving state changes in Haskell and wish I had enough time to go deeper and implement all akgorithms I first came up with. I enjoyed doing research on various AI alrogithms for game agent including other Ants implementation strategies and general papers including "(Rehman Tariq Butt) Performance Comparison of AI Algorithms", especially the anytime algorithm should be useful because I found my ants timeout after the number of ants goes over a certain amount(I manually increased turntime in the included simulation file.). Overall it was very enjoyable to implementing Ants in Haskell.