SDIS - 2nd Project Specification

- 09/04/2018

T5 G07 G08

- André Cruz - up201503776
- António Almeida - up201505836
- Diogo Torres - up201506428
- João Damas - up201504088

Purpose of the Application

We propose the development of an application to distribute adversarial search computations.

Adversarial search algorithms are often computationally heavy, and run on live applications. For instance, in the case of an application for chess, time is often limited to *n* minutes per player, and running an adversarial search algorithm on a common computer is infeasible.

Subsequently, distributing the computations between several computers is a major advantage, both resource and timewise. For example, it might free the client's CPU to run a better graphical interface, an increasingly common aspect of todays' games, while leaving the heavy computations to run on other computers.

Main Features

There are several different algorithms for adversarial search, but all rely on the same structure and interface: a state for the game, a *successor* function, and a utility function. This allows us to generalize a common interface between different games and different algorithms.

Additionally, this type of algorithm operates by expanding the game tree through possible actions, and attributing a score for each one. With this in mind, we can distribute the workload of the possible actions at each turn through several computers, as the computations are completely independent. For instance, at the start of a *TicTacToe* game, the player has 9 choices for placement of the piece, and as such the computations may be distributed through 9 different computers. Eventually, each of these 9 actions may be subsequently distributed through another 8 computers.

Furthermore, as different algorithms have a common interface, it is also possible to use different algorithms to solve different parts of the game tree. In this sense, and knowing all adversarial search problems should comply with a common interface, the peers should be problem agnostic (*e.g.* an implementation of the common interface may be sent on an initial handshake to the peer).

In sum, the main features of our application are: * Distributing workload for a generic adversarial search problem; * Use of different adversarial search algorithms for different state successors; * Full compatibility between clients with the full game specification and recently booted up peers (peers are problem agnostic).

Security

Security will be assured through secure channel communication, and hash of relevant information in order to better tolerate network flooding.

Fault Tolerance

The application should recover seamlessly from peer disconnections, redistributing the uncompleted workload through the rest of the network.

Scalability

Our application should scale linearly in the number of peers and complexity of the problem. In fact, the application's usefulness increases as more peers join the network. This inherent to the proposed architecture.

Proposed Grade Ceiling

Given the project's requirements, we consider our proposal to be fully compliant, and thus have a ceiling of 20.