GENETIC ALGORITHM APPLICATIONS

Applications

Genetic algorithms have been successfully used in many fields of computer science, including but not limited to the optimization of complex algorithms, the training of text classification systems, and the evolution of intelligent artificial agents in stochastic environments.

Board Games

Board games are a very relevant part of the area of genetic algorithms as applied to game theory problems. Much of the early work on computational intelligence and games was directed toward classic board games, such as tic-tac-toe, chess, and checkers. Board games can now, in most cases, be played by a computer at a higher level than the best humans, even with blind exhaustive search techniques. Go is a noted exception to this tendency, and has so far resisted machine attack. The best Go computer players now play at the level of a good novice. Go strategy is said to rely heavily on pattern recognition, and not just logical analysis as with chess and other more piece-independent games. The huge effective Template: Branching factor required for finding high quality solutions heavily restricts the

look-ahead that can be used within a move sequence search.

Computer Games

The genetic algorithm can be utilised in computer games - for example, to allow an enemy opponent to adapt in order to cater against an effective but repetitive tactic exhibited by a human player. This allows for a more realistic game experience; if a human player can find a sequence of steps which, repeated in different games always lead to success, there can be no challenge left. Conversely if a learning technique such as a genetic algorithm for a strategist can avoid repeating past mistakes, the game will have increased playability.

Genetic algorithms require the following components:

- A method for representing the challenge in terms of the solution (e.g. routing soldiers in an attack in a strategy game)
- A fitness or evaluation function in order to determine the quality of an instance (e.g. a measurement of damage done to an opponent in such an attack).

The fitness function accepts a mutated instantiation of an entity and measures its quality. This function is customised to the problem domain. In many cases,

particularly those involving code optimisation, the fitness function may simply be a system timing function. Once a genetic representation and fitness function are defined, a genetic algorithm will instantiate initial candidates as described previously, and then improve through repetitive application of mutation, crossover, inversion and selection operators (as defined according to the problem domain).

The concept of having a full or partial view of the game changes the approach required significantly. An important point to note is that despite a computer obviously having a full view of the game state, making this fully available to the decision making process of an artificial player is going to make them behave unrealistically. Human-centered games are limited by what can easily be manipulated given human mental capacity and dexterity. Video games, on the other hand, operate under no such constraints and typically have a vastly more complex (internal) state space than even the most complex of human-centered games. This richer complexity encourages the development or evolution of more general purpose AI agents than are necessary for playing board or card games with sufficiently simulated skill levels. Currently, most computer controlled players in games implemented using manual scripting, which is quite tedious and time consuming to develop and test. The use of computational intelligence techniques offers an interesting alternative to scripted approaches, whereby the agent behavior

can be controlled using an evolved neural network, for example, rather than being programmed. Since such a neural approach may result in unique, novel behaviour impossible to achieve with manual implementation, the resulting quality of the product could be far higher. Such approaches often result in significantly more original results for every different player, the replayability of the game is extended. Furthermore, evolved AI players tend to be excellent at exploiting loopholes in a game. Identifying and eliminating these elements (which are seen as problems by the players/developers) can be achieved by genetic algorithms. In addition to this, through the life of a game, human players will also discover these, giving them an unfair advantage. If an AI player can also take advantage of these, the playing field is levelled. To have game characters that effectively exhibit such "human" characteristics improves the game, extends its lifetime, and increases total revenue. These techniques must be developed very carefully, especially in cases where agent difficulty is curved to compete fairly with the player. It has been an established technique in such games for a human player to intentionally play badly earlier on in the game, thus easing the difficulty; which in the end will decrease the quality and longevity of the game. Unfortunately, players tend to blame the developer for allowing it to be possible.

Source:

http://www.juliantrubin.com/encyclopedia/electronics/genetic_algorithm.html