**Convolutional + Artificial Neural Network + Max-Pooling:**

<http://antoniosliapis.com/papers/learning_the_patterns_of_balance_in_a_multi-player_shooter_game.pdf>

**Summary:**

3v3 bots that are given different weapons each play.

Researchers use a top-down view of the level: tiles divided in empty, wall and cover, expressed as 00, 01 and 10.

Next, they run algorithms to produce different graphs (basically image/edge analysis) via a CNN and they intertwine the process with Max-Pooling (taking significant samples).

They use the ANN to take both level nodes and weapon nodes (different for each play) and determine if the map is balanced or if any team has advantage + weapon win changes for each team and weapon.

Used to test how balanced the level is.

**Genetic Algorithm:**

<http://julian.togelius.com/Cardamone2011Evolving.pdf>

In this case, 4 generators are employed to generate levels:

* Take an empty level and place blocks
* Take a full level and delete them
* Place blocks where the player goes
* Random

To evolve these levels there are 2 requirements:

1. Have a “score” that can be assigned to a level, in this case T \* S, where T is the time until the bot gets killed and S is the total surface of the level.

*(too simplistic of an heuristic)*

1. Have a set of parameters used to construct the level, which will constitute the variables in the genetic algorithm.

*(the paper doesn’t mention them)*

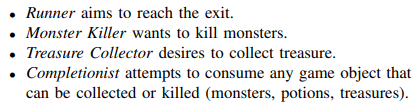
Then, the genetic algorithm is applied within each generator to select those levels with the highest score (T \* S) so that they can be mixed together (probability) and mutated (probability) over and over again.

Used to test how “good” the level is.

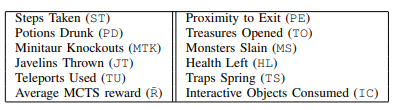
**Procedural Personas / MCTS + Evolved heuristics / Genetic Alg.**

<https://arxiv.org/pdf/1802.06881.pdf>

The experiment involves 4 types of items, enemies and player personas:

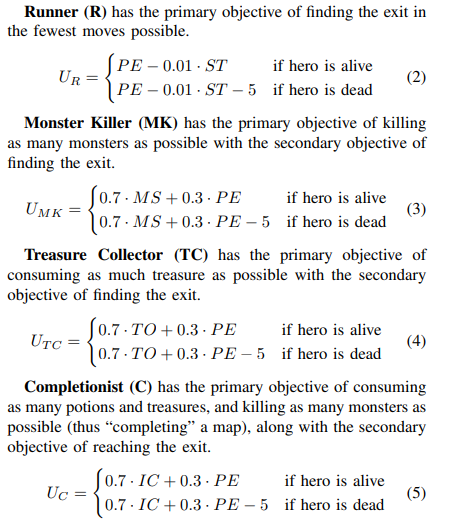


It uses MCTS as the way for a player to calculate the next best move.



To get things going (rollout phase), 10 random moves are simulated.

From that point on and for each player type, there is a formula (UCB1 of the MCTS) that gives a score for each potential move node:



check how MCTS works: <https://int8.io/monte-carlo-tree-search-beginners-guide/>

Each generation, 100 players are generated. From what I understood, they are given extra variables (with random values) on top of the ones expressed above in the move-selecting formula

Once a player has played x amount of time or killed x amount of enemies or died (or…) its “fitness” is calculated (eg if monster killer indeed killed enough monsters) on average in multiple maps.

Next, the heuristic used for calculating how “good” a move is for a player is now updated via genetic algorithms: those that perform the best are chosen to reproduce and mix their parameters (the extra ones), so that a new move-selecting formula is created.

Note: multiple simulations can be employed with 2 purposes:

1. Variance between simulations
2. Evolving the level (score) on each simulation

**Real Game examples**

**Sea Of Thieves (other genre)**

<https://www.gdcvault.com/play/1026366/Automated-Testing-of-Gameplay-Features>

<http://eej.dk/gain/2017/slides/AUTOMATED_TESTING_FOR_MULTIPLAYER_GAME-AI_IN_SEA_OF_THIEVES.pdf>