Evolving Strategies for a Turn-Based Game

Bentley James Oakes

McGill University

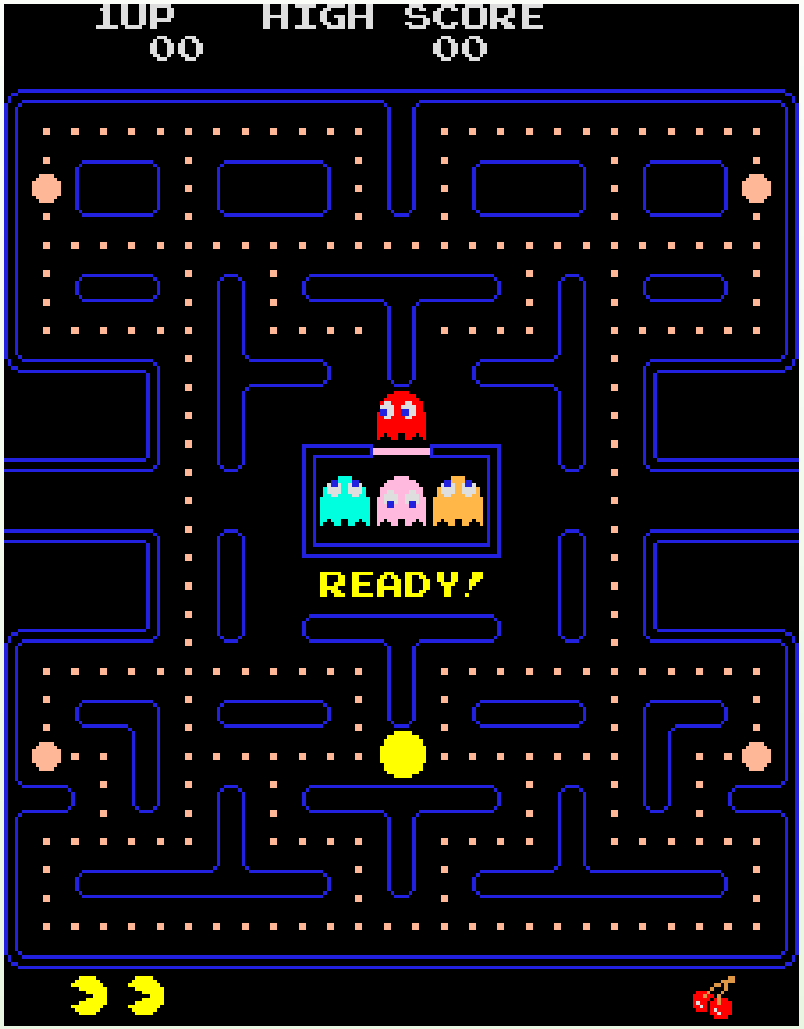
bentley.oakes@mail.mcgill.ca

May 8, 2013

Outline

About Video Games

- Electronic software for entertainment or serious uses
- Multi-billion industry
- Different genres and mechanics
- Computer vs. player, player vs. player



Why Artificial Intelligence

- Not always player vs. player competition
- AI controls elements of the game world
  - Teammates
  - Enemies
  - Level structure
  - Item placement
- Can have huge impact on player enjoyment

AI Methods

Different domains require different strategies

Game	**Properties**	**AI Structure**
Chess	Turn-based	Look-ahead tree
First-person shooter	Highly tactical and reactable	Sight/sound/cover locations
Real-time strategy	Multiple units/building structures	Rules or influence map

Creating and Tuning Rules

Important AI Query:

When to build more tanks?

If Num(Soldiers) $>$ 10, Build 10 Tanks

Creating and Tuning Rules

Important AI Query:

When to build more tanks?

If Num(Soldiers) $>$ 10, Build 10 Tanks

- Can be seen as a point in a strategy space
- Difficult/time-consuming to hand-tune
- Requires effort by AI engineers/designers
- Can we automatically search this space?

Outline

Battle for Wesnoth

- A turn-based strategy game
- Multi-platform and open source
- Single-player campaigns or multi-player battles
- Written in C++ and Lua

Battle for Wesnoth

- Multiple units on a team
- Multiple teams
- Unit defence based on terrain type of hex
- Highly random battles

AI Strategies

Problem Statement:

Create an AI that can defeat the opposing team by ordering units to move and attack

AI Strategies

Problem Statement:

Create an AI that can defeat the opposing team by ordering units to move and attack

Questions:

- What structure should strategies have?
- How to evaluate strategies?
- How to modify strategies to make them better?

Outline

Evolutionary Computing

Inspired by biological evolution/natural selection

Evolutionary Computing

Inspired by biological evolution/natural selection

Algorithm:

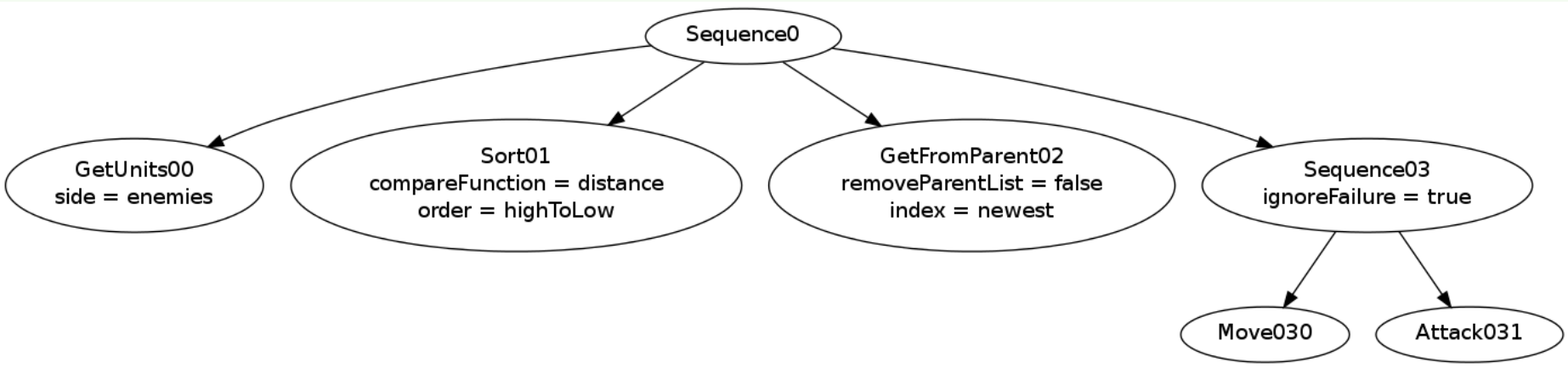
1. Define the problem space
2. Create a number of random solutions to the problem
3. Evaluate each solution
4. Combine elements from good solutions to produce new solutions
5. Repeat the last two steps

Three Main Questions

1. How are solutions to the problem represented?
2. How are solutions evaluated?
3. How are solutions combined together?

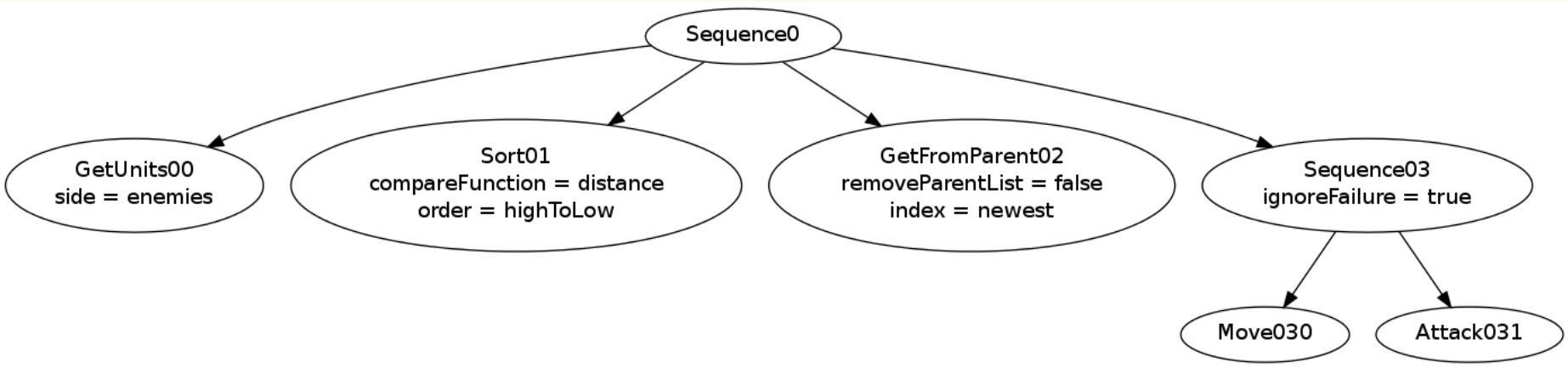
Representation

- Behaviour trees will represent AI strategies
- AI will query tree to determine action for each unit
- Requires hand-built nodes



Representation

- Behaviour trees will represent AI strategies
- AI will query tree to determine action for each unit
- Requires hand-built nodes



- Behaviour trees are easy to evaluate and combine
- Can this represent all strategies?

Evaluation

- Fitness/objective function measures strength of solution
- Strategies must be evaluated in-game
- How to determine strategy strength?

Fitness Function

Fitness Function

Team health = sum of units' health

Turn fitness = team1 health - team2 health

Battle fitness = average(turn fitnesses)

Fitness Function

Fitness Function

Team health = sum of units' health

Turn fitness = team1 health - team2 health

Battle fitness = average(turn fitnesses)

- Will this lead to interesting or broken behaviour?
- What's the fitness function for 'fun'?
- Infinite fitness functions to be tried

Combination

Fitness function gives a ranking of solutions
Combine elements from 'fit' solutions

Combination

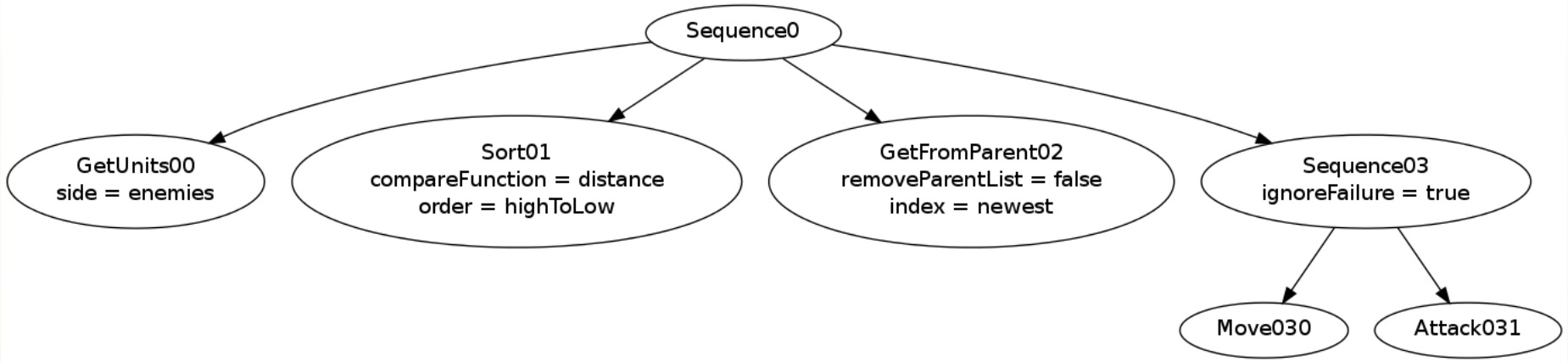
Fitness function gives a ranking of solutions
Combine elements from 'fit' solutions

Crossover

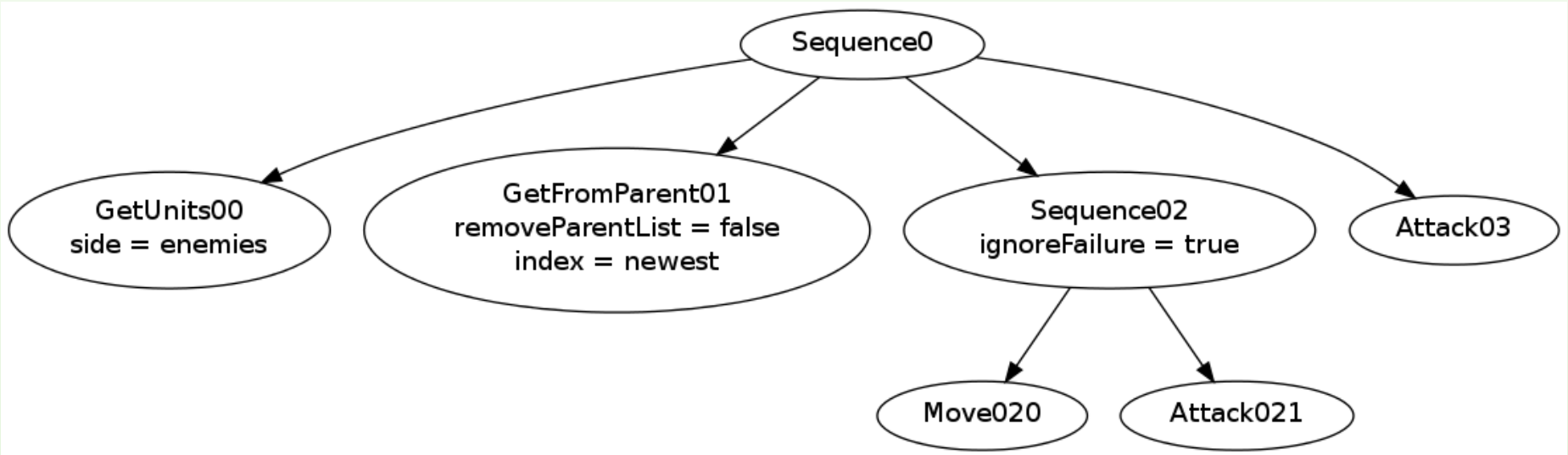
Similar to sexual recombination in genetics
Pick two 'parent' solutions and create a child solution
Pick subtree on one parent, and replace with subtree from other parent

Crossover

Parent A:



Parent B:



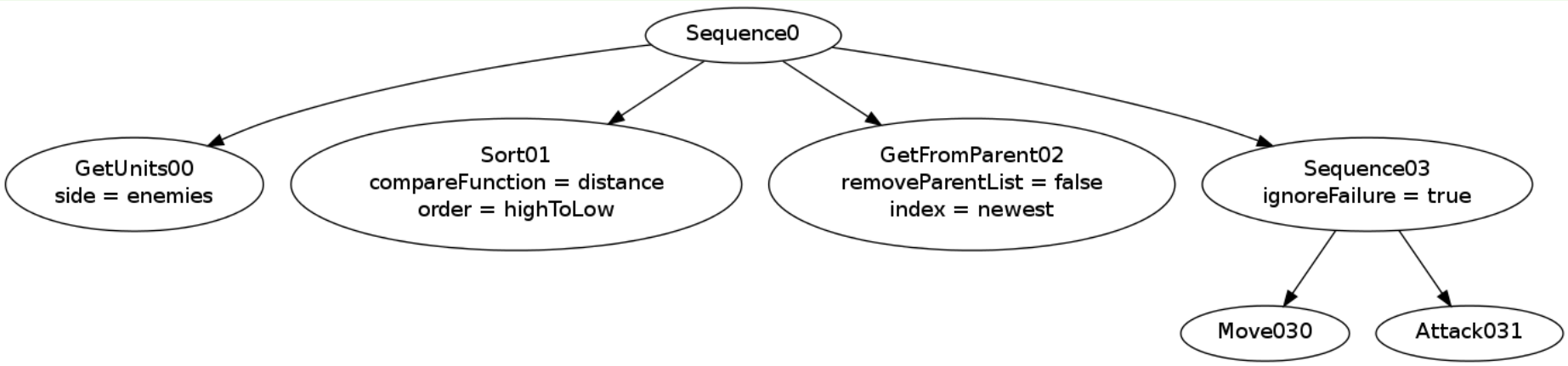
Mutation

Similar to genetic mutation

Changes solutions to explore new ideas

Easy to do with tree structure

- Add/delete nodes
- Modify node attributes



Repetition

- New child solutions have been created, so repeat the evaluation/combination process
- Keeps searching solution space
- 'Directed' random search
- Can get arbitrarily close to optimal with time

Outline

Evaluate AI Performance

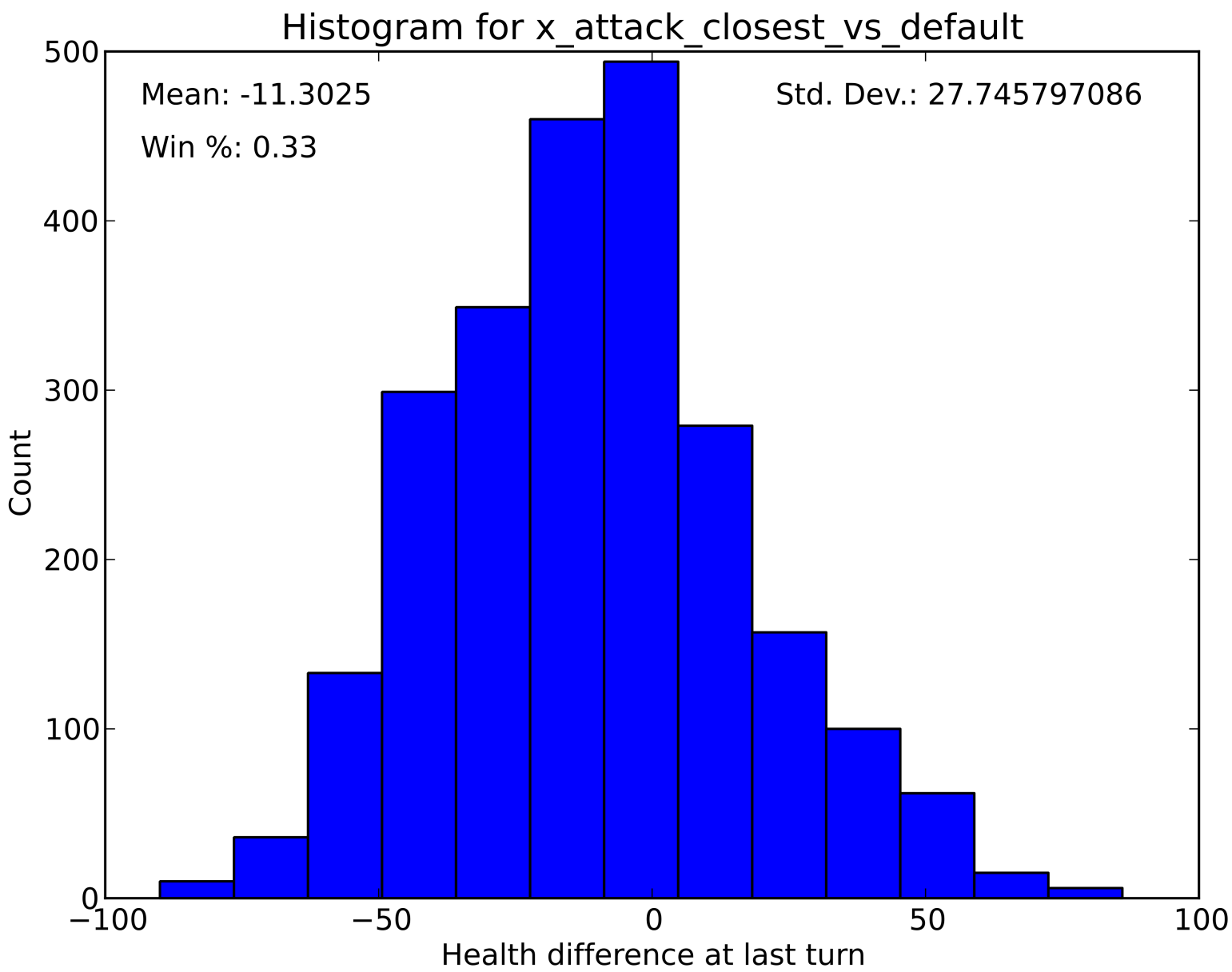
- Randomness in battles may cause noise in win percentage
- Create another metric to compare AIs
- Record health difference between teams at the end of a battle
- Ran 2400 battles between pairs of all AIs

Baseline Results Hand-Built

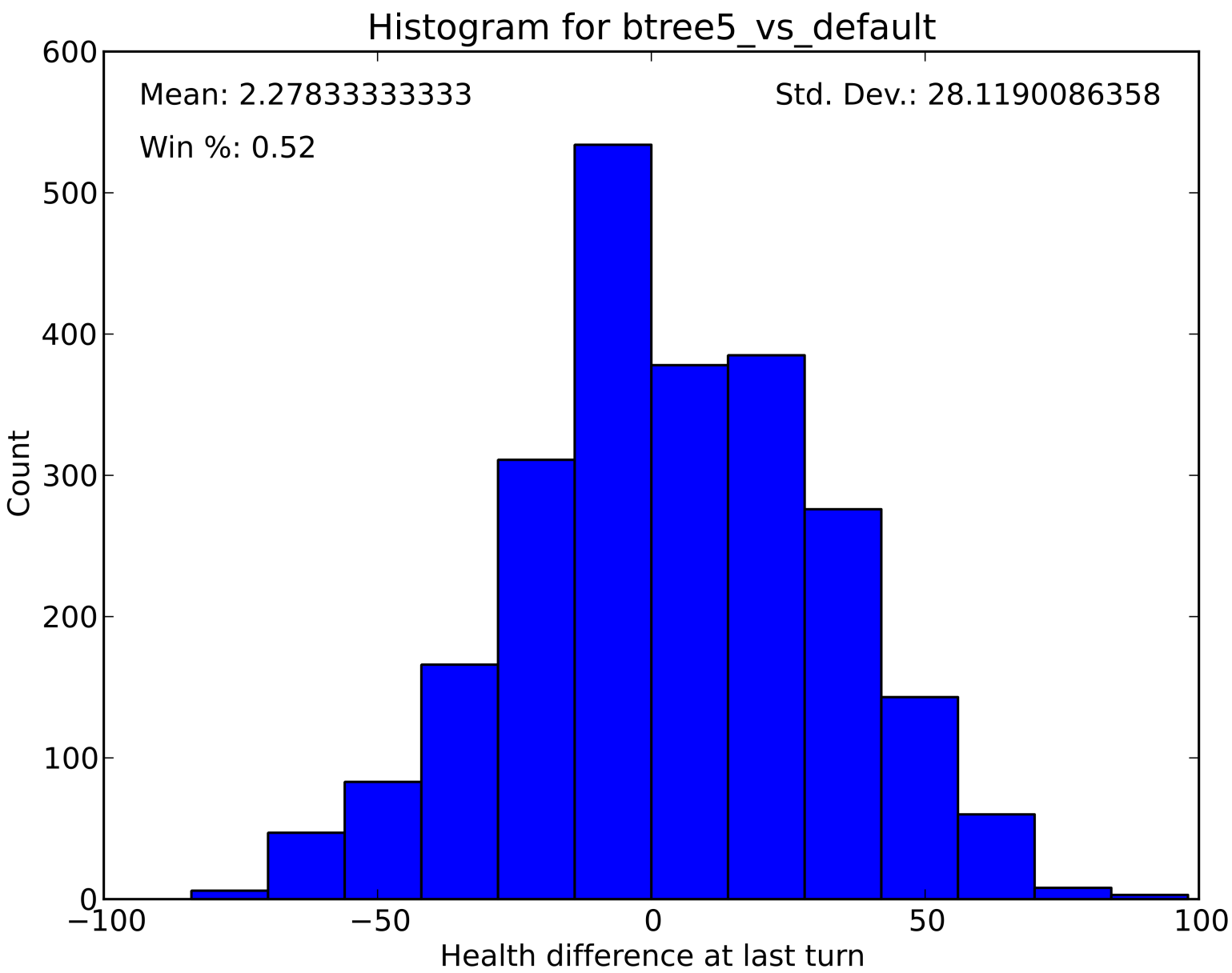
AI Name	Win %	Mean	Std. Dev.
Attack Weakest	24	-22.88	34.11
Attack Closest	40	-7.89	29.59
Pick Strong Terrain	37	-12.11	39.75
Default	60	6.21	35.42

Table: Hand-built trees vs the default AI

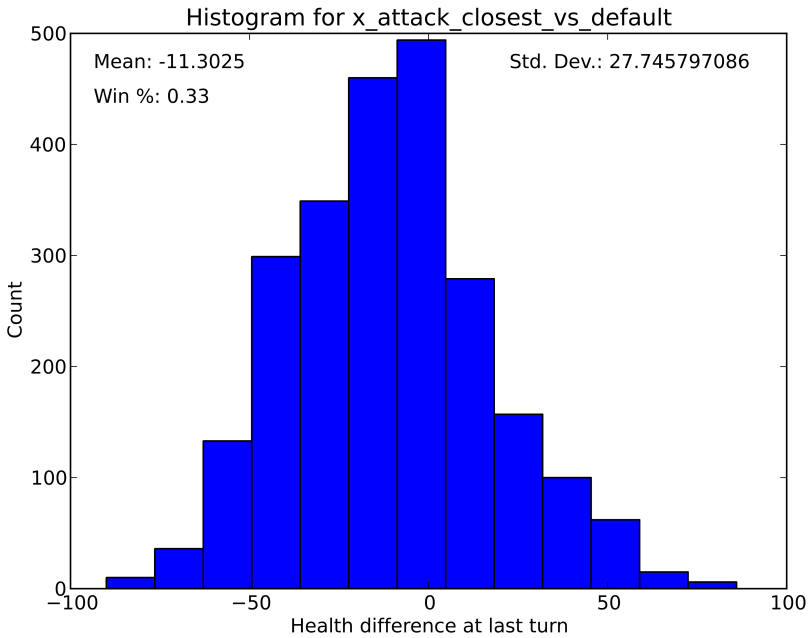
Health Difference Histogram

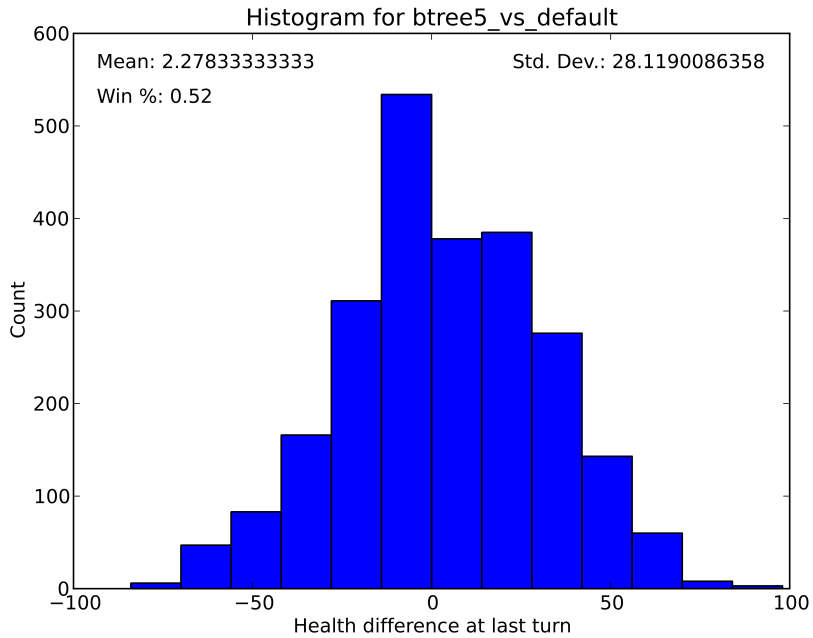


Evolved Tree Results



Comparison





AI Name	Win %	Mean	Std. Dev.
Attack Closest	33	-11.30	27.75
Evolved Tree	52	2.28	28.11

Outline

Summary

- Designing an artificial intelligence is extremely non-trivial
- Must define structure, capabilities, and goals
- How do you define 'fun, interesting, challenging' mathematically?
- Genetic algorithms offer framework for these questions
- Evolving AI strategies shows promise

Future Work

- Define more behaviour tree actions
- Test other fitness functions
- Test evolved strategies on other maps/under different conditions
- Recruit Wesnoth players to design behaviour tree AIs

Thank You

Questions?