Vulture Combat

Potential Fields, Reinforcement Learning and Bayesian Networks Applied to Starcraft Broodwar

Analysis of the Game

Basic Game Description

Elements for Winning a Match

Build Orders

Information Gathering

Macro-management

Micro-management

Terran Tactics

Timing Attack

Pushing

Harassment

Unit Analysis

Vultures

Hit points	80
Range	5
Weapon Cooldown	30
Price	75m
Building needed	Factory

Wraiths

Hit points	120
Range	5
Weapon Cooldown	22/30
Price	150m 100g
Building needed	Starport

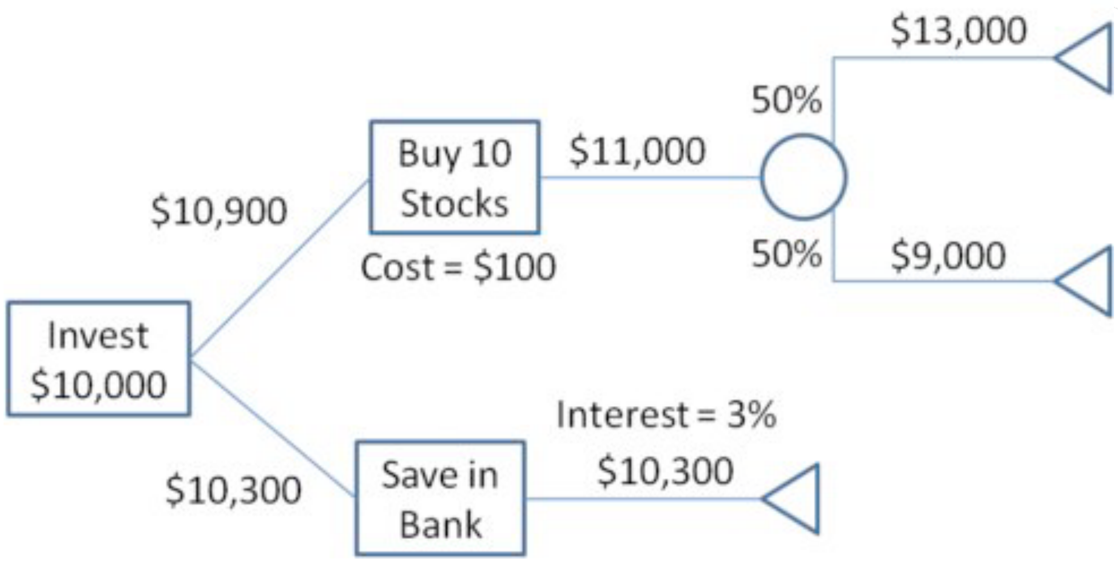
Marines

Hit points	40
Range	4
Weapon Cooldown	7.5/15
Price	50m
Building needed	Barracks

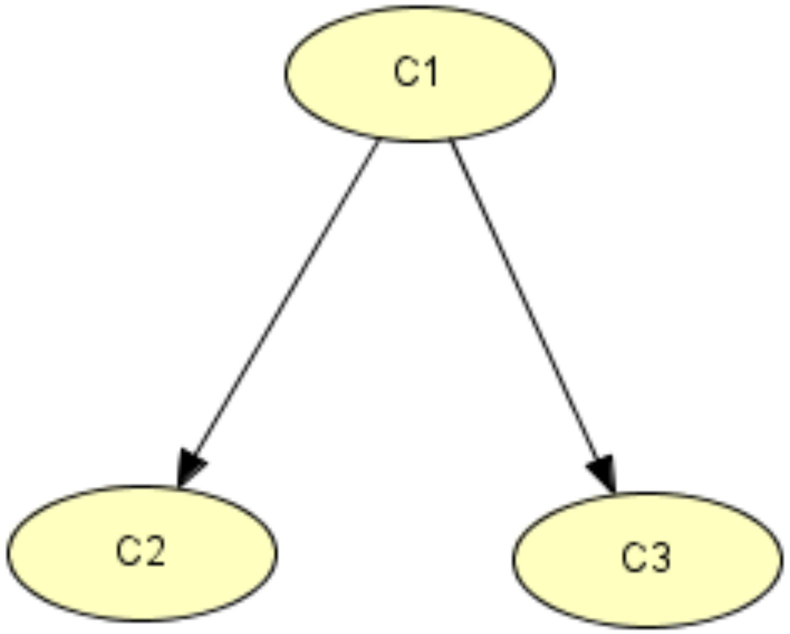
Bayesian Networks

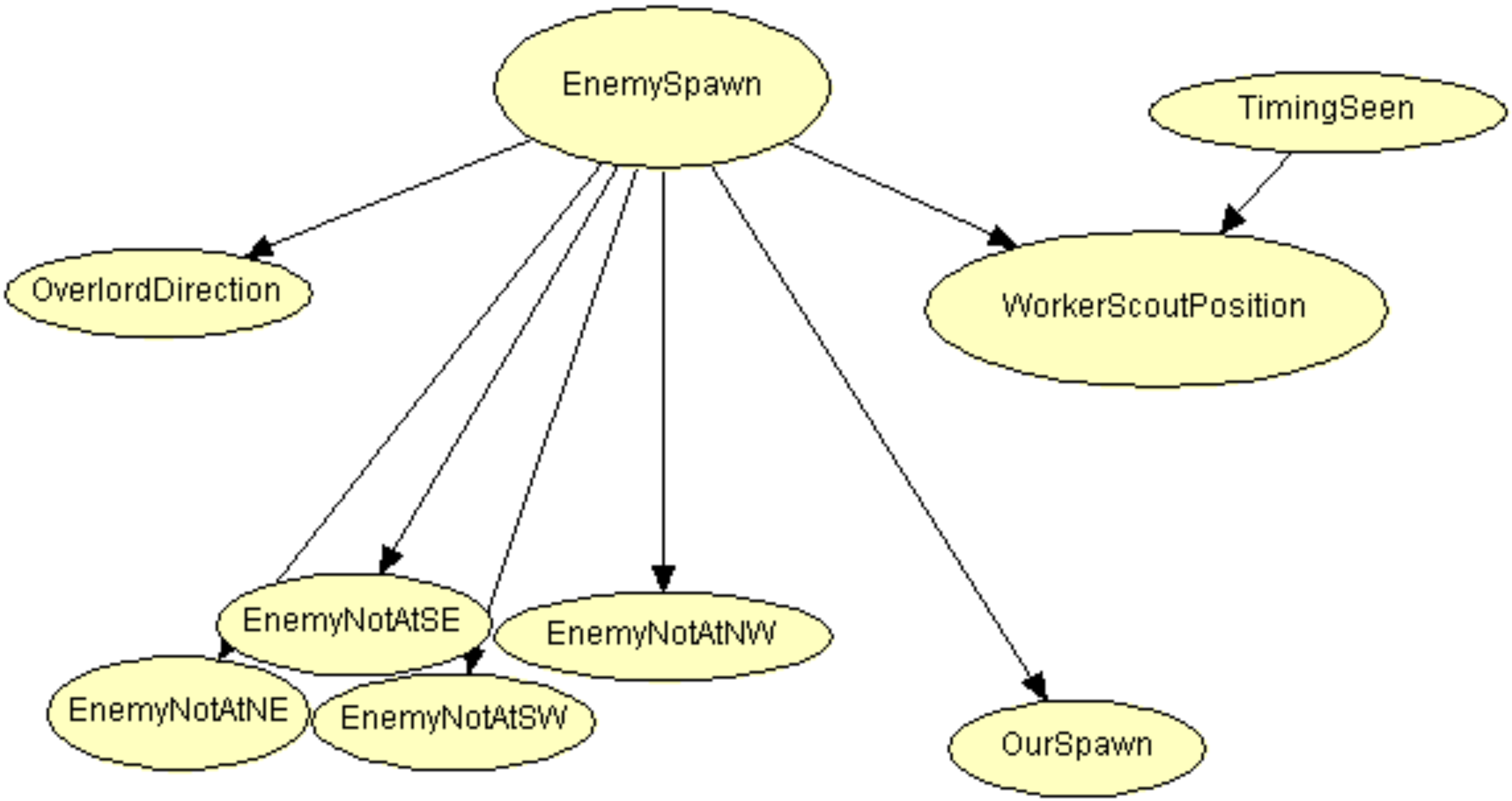
Choice of Decision Model

Decision Trees

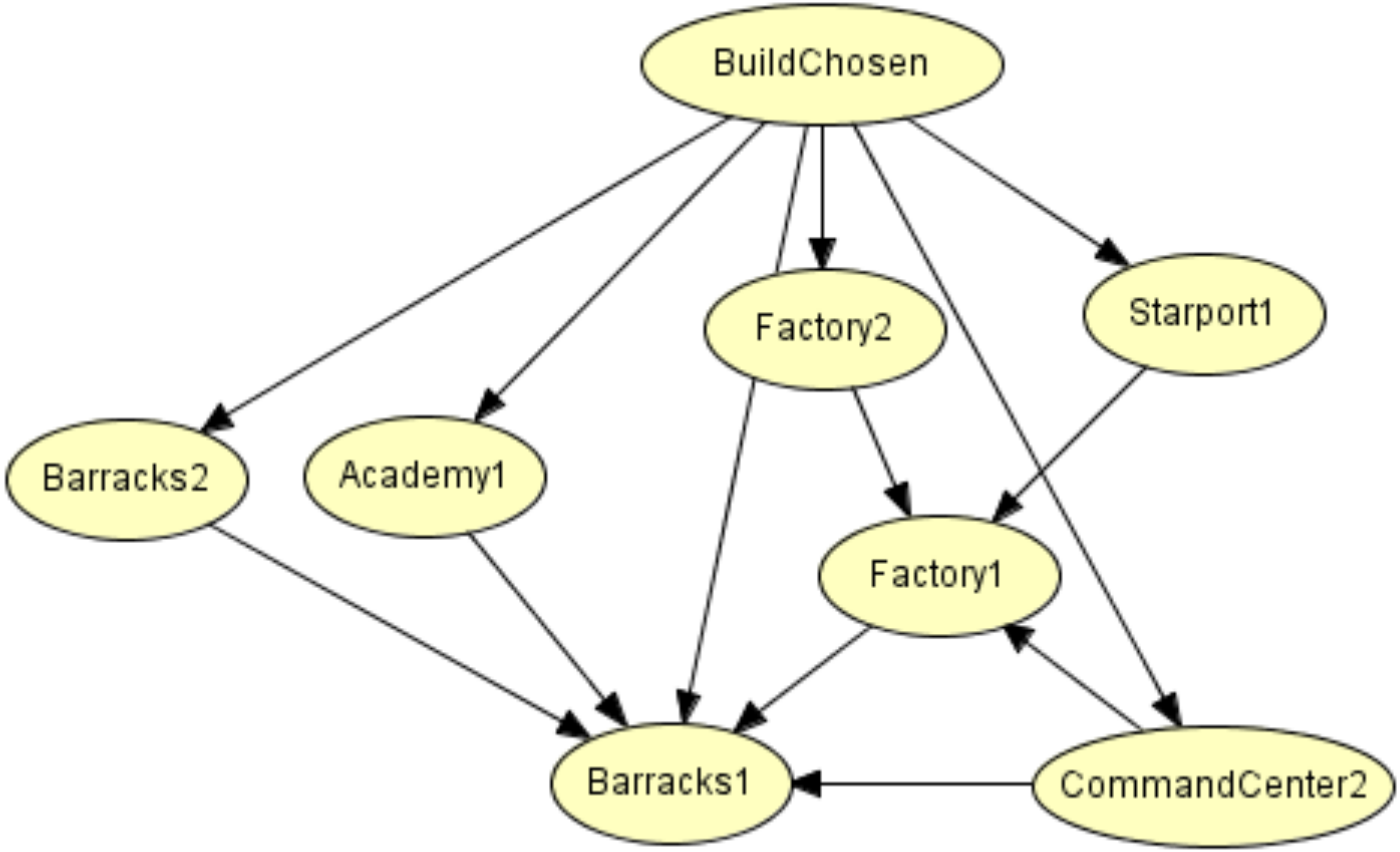


Bayesian Networks

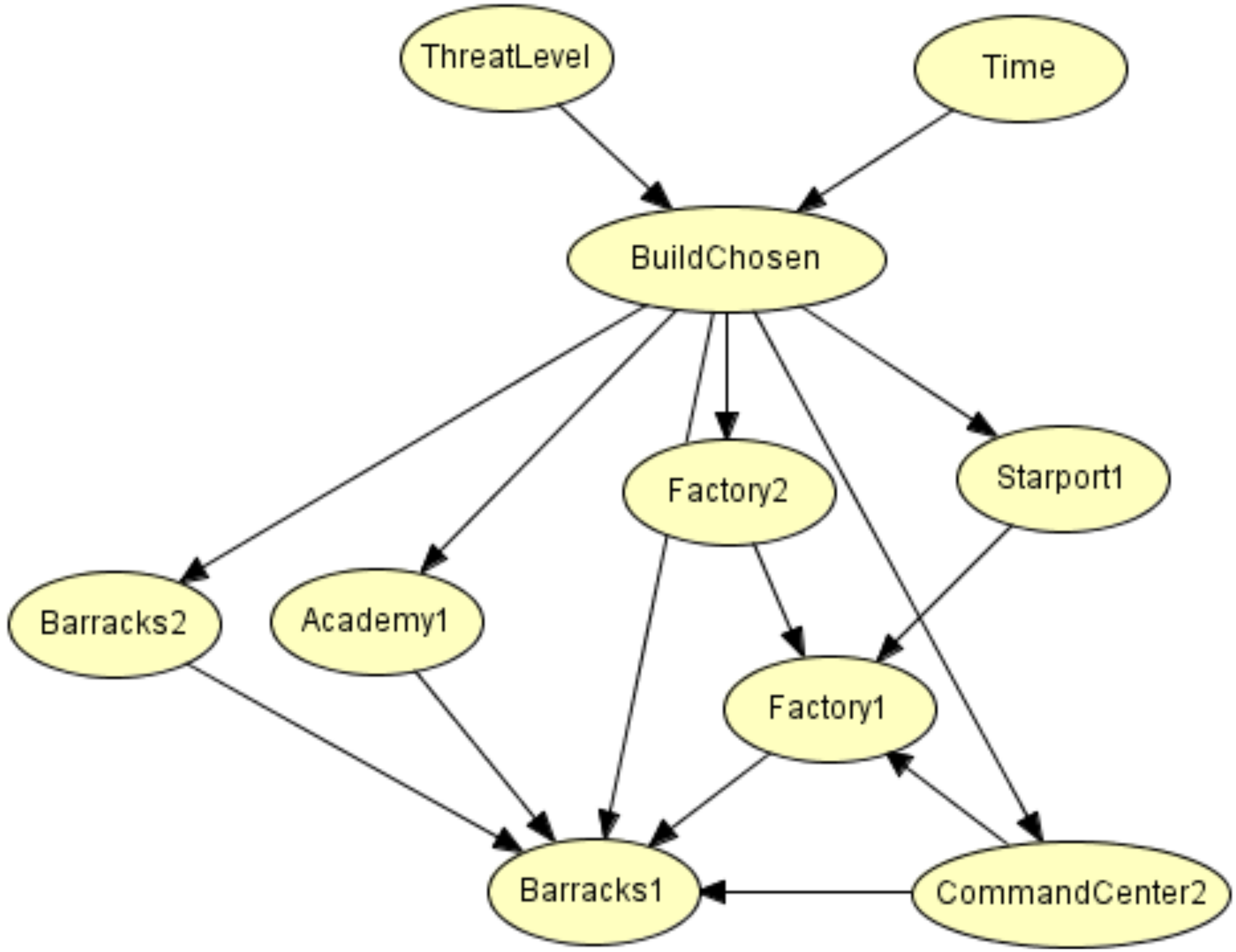




Spawn Prediction



Build Order Prediction



Threat Level Prediction

Potential Fields

Choice of Movement Model

Shortest Path

Potential Fields

- Easy to modify
- Dynamically updating environment

General Behavior

$$Attractive = \begin{cases} f * c & \text{if } d > s \\ 0 & \text{else} \end{cases}$$

$$Repulsive = \begin{cases} -f * c & \text{if } d > s \\ 0 & \text{else} \end{cases}$$

Using the numerical representation of a vector.

Example of a Potential Field Function

- Behavior determined by sign (+ or -)
- Difference between *due* and *de*
- Using *(2de - due)*

$$MDP = \begin{cases} f_{MDP} \times (2de - due) & \text{if } de < sr \\ 0 & \text{if } de > sr \end{cases}$$

List of Potential Fields

Maximum Distance Positioning

Edges and Cliffs

Squad Center

Ally Units

Weapon Cooldown

Changes Made During Implementation

All weights or forces are positive.

The magnitude of the forces is learned.

Reinforcement Learning

Generalization of Q-Learning

Environment Variables

A State is defined as the combination of all the following characteristics:

Number of units

Health Lost

Damage Dealt

Number of Units

Time

Weapon's Cool Down

Shooting Range

Distance to Ally

Distance from Current Tile to Ally

Distance to Center of Squad

Distance from Current Tile to Center of Squad

Distance to Enemy

Distance from Current Tile to Enemy

Distance to Cliff or Edge

Distance from Current Tile to Cliff or Edge

Formulas

Q-Approximation

$$\hat{Q}_f = f_{MDP}(2de - due) + f_{AU}(2da - dua) + f_{EAC}(2dc - duc) + f_S(2ds - dsv) + f_{CD}(2de - due)$$

Reward

$$R(s) = C_1 numberOfUnits - C_2 healthLost + C_3 damageDealt + C_4 numberOfKills - C_5 time$$

Updating Rules

General Rule

$$f_i \leftarrow f_i + \alpha[R(s) + \gamma(max\hat{Q}_f(a', s')) - \hat{Q}_f(a, s)]\frac{\partial \hat{Q}_f(a, s)}{\partial f_i}$$

Application to Ally Unit Coefficient/Force

$$f_{AU} \leftarrow f_{AU} + \alpha[R(s) + \gamma(max(\hat{Q}_f(a', s'))) - \hat{Q}_f(a, s)](2da - dua)$$

Learning Algorithm

In Every Frame

1. From the current unit position select the highest $Q(a,s)$ value.
2. Select the highest value for $Q'(a',s')$ form this state $s' = \delta(a,s)$.
3. Calculate the reward R(s,a).
4. Save all the values.
5. Move to the new position.

Every 25 frames

1. For every set of values saved in the buffer:

   Calculate the new weights with their corresponding update rule.

2. Save the new accumulated weights to the weights/coefficients used to calculate the next move (*Q* and *Q'* values).

Tests

Base Case, without reinforcement learning or potential fields.

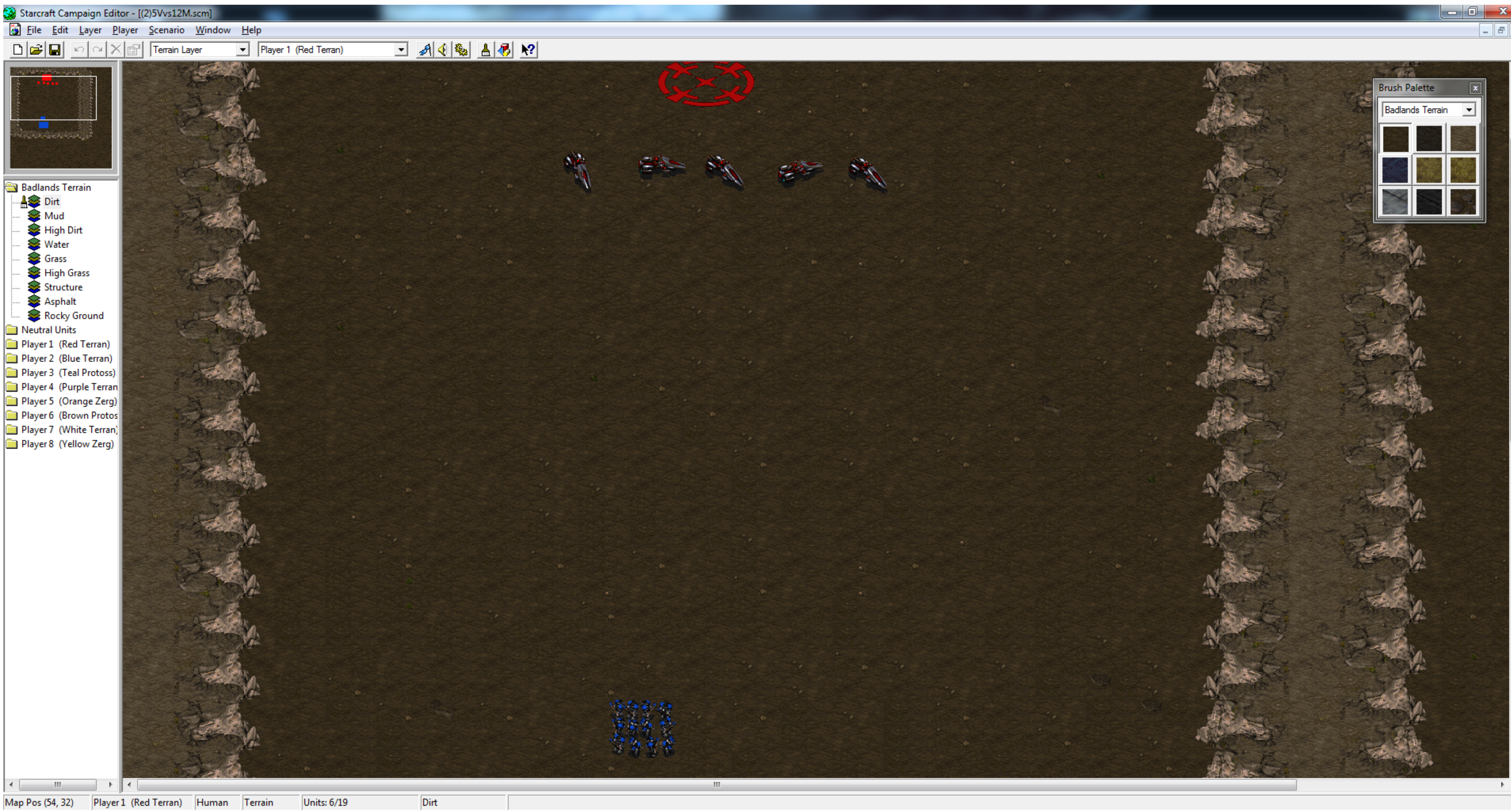
Base Case, with potential fields.

Reinforcement Learning Results.

Predict spawn and build order.

Threat level prediction.

Building a Test Map



Base Case

Testing without reinforcement learning or potential fields:

Test results from first map

Players	Produced units	Killed units	Lost units
Player with vultures	5	9	5
Player with Zerglings	30	5	9

Test results from second map

Players	Produced units	Killed units	Lost units
Player with vultures	5	6	5
Player with marines	12	5	6

Base Case

Testing with potential fields, but not reinforcement learning:

Test results from first map

Players	Produced units	Killed units	Lost units
Player with vultures	5	30	0
Player with Zerglings	30	0	30

Test results from second map

Players	Produced units	Killed units	Lost units
Player with vultures	5	6	5
Player with marines	12	5	6

Reinforcement Learning Tests

Running with different values

Comparing Different Alpha and Gamma Values

Alpha 0.4, Gamma 0.6 (135936 iterations)

Damage taken	Damage given	Units lost	Enemies killed
395,51	338,29	4,86	7,52

Alpha 0.2, Gamma 0.9 (30852 iterations)

Damage taken	Damage given	Units lost	Enemies killed
398,78	282,28	4,96	6,31

Predict spawn and build order

Threat Level Prediction

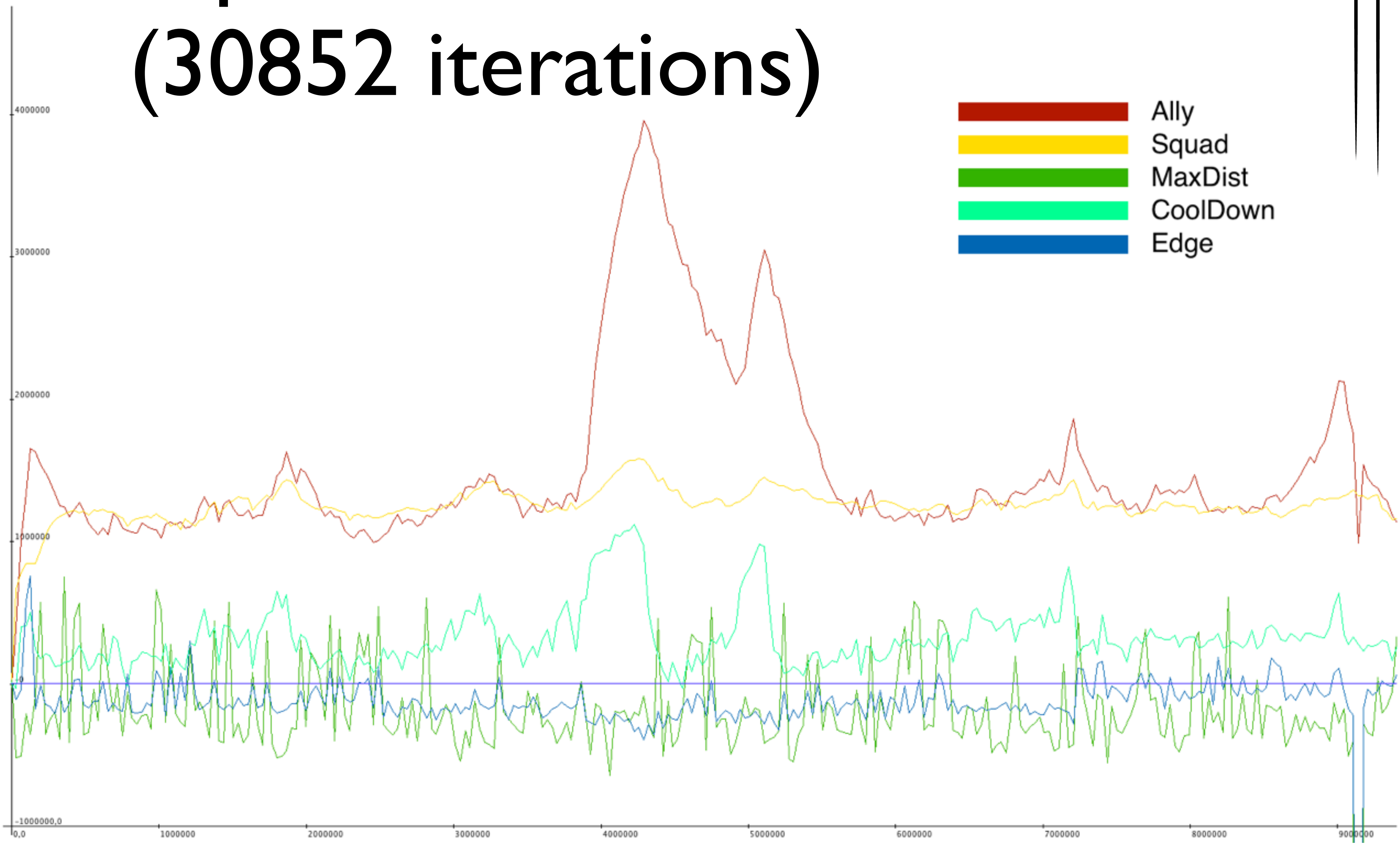
Convergence

Converging Results

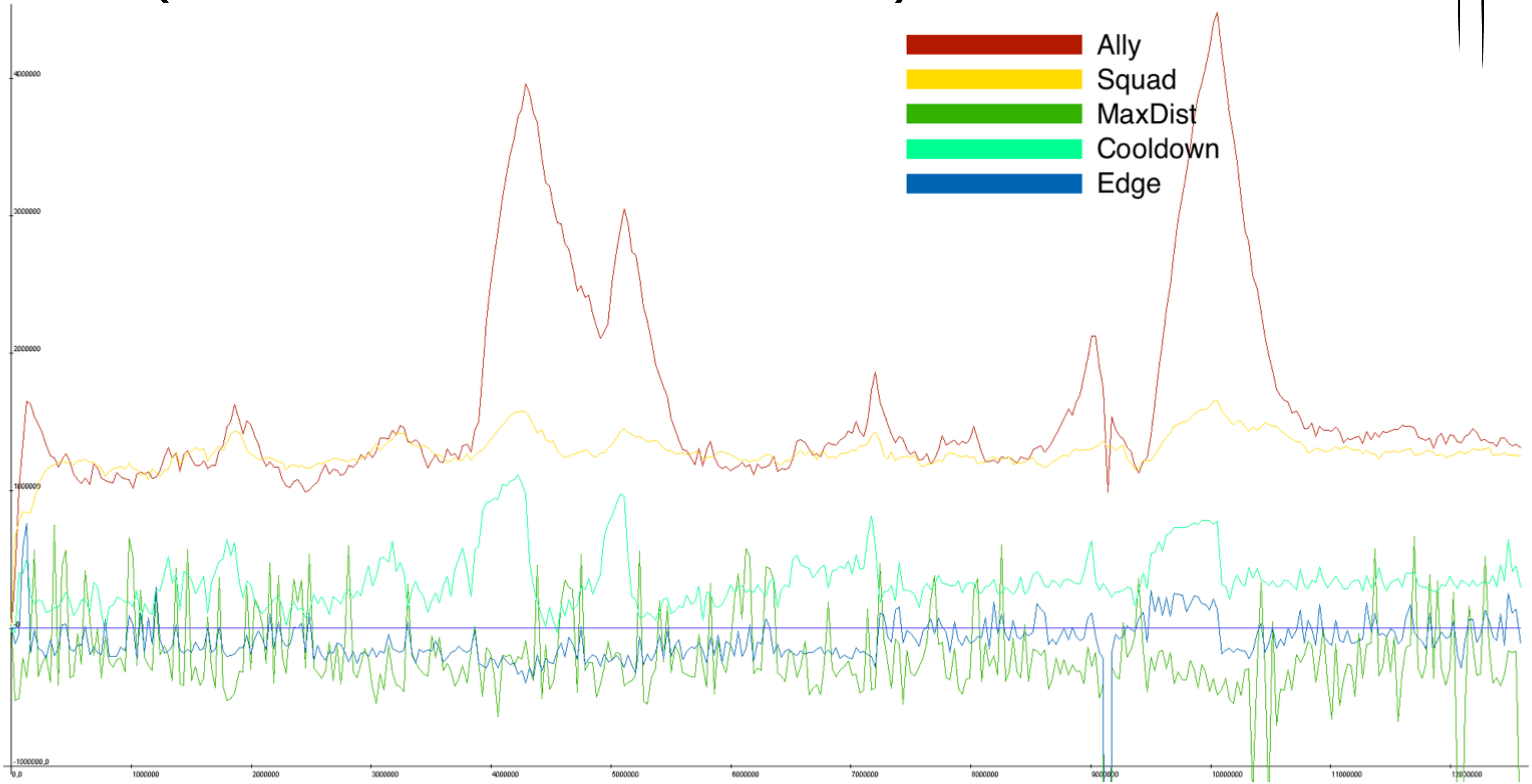
How do different Alpha and Gamma values affect convergence.

How many iterations are needed?

Alpha 0.2, Gamma 0.9 (30852 iterations)



Alpha 0.4, Gamma 0.6 (135936 iterations)



Future Works

Potential Fields

Does not work well on full games.

Add more variables.

Repair vultures.

Apply movement speed upgrade.

Reinforcement Learning

Compare different reward functions, adapting to aggressive/defensive scenarios.

Train on different enemy race - unit types.

Keep bot running until all of the values converge.

Make a more comprehensive Q-Approximation: number of units, enemy unit type.

Bayesian Networks

Use data mining.

Add more build orders.

Create extra networks.

Conclusion

Identify parts of the games were machine intelligence was applicable.

Apply several concepts of machine intelligence for different circumstances.

- Potential Fields + Reinforcement Learning.

- Bayesian Networks.

Show improvement through testing.

Demonstration