Adaptive Game AI

(based on slides of Pieter Spronck UvT)
Goals of Adaptive AI

• Self-correction
  – Automatic reparation of “exploits”
  – Online or offline
  – Direct or indirect

• Creativity
  – Being able to deal with new situations
  – Usually online
  – Direct

• Scalability
  – Appropriate challenge for novices and experts
  – Online
  – Direct or indirect
Is It Needed?

• No benefits?
  – Then don’t
  – How smart should an enemy be if he lives for only 15 seconds?

• Can it be faked?
  – Then fake it

• Will the publisher accept it?
Why the Hesitation?

• Learning the wrong lessons
• Difficulty to get desired results
  – Suitable fitness function
  – Increased entertainment
• Difficult to modify, test, debug, and understand
• Low efficiency of common techniques
  – Neural networks
  – Evolutionary algorithms
• It does not always make sense
Problem of Complexity

- Huge state-action space
- Uncertainty
  - Non-deterministic
  - Incomplete information
  - Multiple parallel agents
- Often real-time

“When dealing with problems such as Stratagus you might as well throw the three chapters on search in my book in the garbage because these are irrelevant.”
(Stuart Russell, IJCAI05)
Requirements

• Computer-controlled
• Online computational requirements
  – Speed
  – Effectiveness
  – Robustness
  – Efficiency
• Functional requirements
  – Clarity
  – Variety
  – Consistency
  – Scalability
Necessities

• Use as much prior knowledge as possible
  – Improves all computational requirements
• Good performance measure (fitness)
• Learning by
  – Optimisation
  – Reinforcement
  – Imitation
• Avoid overfitting
• Minimise dependencies between behavioural aspects
  – E.g. learning of “kill hotspots” and “weapon preference”
• Balance between exploration and exploitation
Elements of Adaptation

- Domain knowledge
- Player model
  - Explicit or implicit

A fireball is effective when thrown in the middle of a closely-knit group of enemies

The opponent always starts a fight with all enemies huddled together

At the start of a fight, throw a fireball in the middle of the group of enemies

Knowledge Base

Player Model

Agent

Game AI

Game World
Adaptation Techniques
Decision Trees

• Learn by building and tuning a decision tree

• Advantages
  – Robust
  – Simple
  – Readable

• Disadvantages
  – Need tuning
  – Only suitable for relatively simple reasoning problems

• Can be applied both offline and online
  – Depending on the precise implementation

• Used in Black & White

- Diagram showing a decision tree with criteria for determining if an object is being eaten. Criteria include whether the object is animate or inanimate, and the type of object (human or animal). The diagram indicates different weights for each criteria.
Neural Networks

• Learn by training weights in a simulated brain
• Advantages
  – Flexible
  – Can learn any non-linear function
• Disadvantages
  – Need tuning
  – Only a few outputs possible
  – Hard to choose input variables
  – Slow to train
• For creating new behaviour offline only
  – Can be further tuned online
• Used in Creatures, Black & White, Colin McRae Rally 2.0
Evolutionary Algorithms

- Learning inspired natural selection and natural genetics
- Advantages
  - Flexible
  - Robust
  - Can explore large, noisy search spaces
- Disadvantages
  - Need tuning
  - Slow to evolve
  - Not guaranteed to find a solution
- Offline only
- Used in Creatures
Reinforcement Learning

• Learning by rewarding good behaviour and punishing bad behaviour

• Advantages
  – Simple, elegant
  – Usually easy to implement
  – Can solve wide variety of problems
  – Can find close to optimal solution
  – Learns during interaction with environment

• Disadvantages
  – Suitable state-space representation hard to determine

• Can be applied both offline and online
  – Depending on the precise implementation
Evolutionary Learning for RTS Games
Evolving Winning Strategies

Note: a chromosome represents a strategy

2nd note: This research was performed by Marc Ponsen and Pieter Spronck
Wargus Game States

- The possible tactics during a game mainly depend on available units and technology.
- The availability of units and technology depends on the buildings the player possesses.
- Therefore, the utility of tactics depends on the available buildings.
- The 20 states in Wargus are manually predefined and correspond to the set of buildings a player possesses.
Chromosome

Chromosome: Start | State 1 | State 2 | ... | State m | End

State: State marker | State number x | Rule x.1 | Rule x.2 | ... | Rule x.n

Rule: Rule ID | Parameter 1 | Parameter 2 | ... | Parameter p

Start | S | 1 | C1 | 2 | 5 | def | B | 4 | S | 3 | E | 8 | R | 15 | B | 3 | S | 4 | ...

Rule 1.1 | Rule 1.2 | Rule 3.1 | Rule 3.2 | Rule 3.3

State 1 | State 3
Strategy-Genes Translation

• Chromosome = {"Start","S",1,"C1",1,9,"attack", "C1",8,2,"defend","B",2,"",....

• Corresponding strategy in Lua:

```lua
--We have reached state 1

function() return AiForce(1, {AiSoldier(), 9}) end,
function() return AiWaitForce(1) end,
function() return AiAttackWithForce(1) end,
function() return AiForce(8, {AiSoldier(), 2}) end,
function() return AiForceRole(8,"defend" ) end,
function() return AiNeed(AiBarracks()) end,
function() return AiWait(AiBarracks()) end,
```

TACTICS FOR STATE 1
Genetic Operators

- State Crossover
- Rule Replace Mutation
- Rule Biased Mutation
- Randomization
Tactics

• Two ‘balanced’ tactics
  – Small Balanced Land Attack (SBLA)
  – Large Balanced Land Attack (LBLA)

• Two ‘rush’ tactics
  – Soldier Rush (SR)
  – Knight Rush (KR)
Fitness Determination

• Based on success against ‘rush’ tactics (soldier rush and knights rush)
• Fitness is based on fraction of military points scored by counter-tactic (win>0.5)
• Goal: target-fitness (SR: 0.75, KR: 0.70)
• Abort: target-fitness reached or 5 generations
# Evolution Results

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Tests</th>
<th>Low</th>
<th>High</th>
<th>Avg.</th>
<th>&gt;250</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>10</td>
<td>0.73</td>
<td>0.85</td>
<td>0.78</td>
<td>2</td>
</tr>
<tr>
<td>KR</td>
<td>10</td>
<td>0.71</td>
<td>0.84</td>
<td>0.75</td>
<td>0</td>
</tr>
</tbody>
</table>
Analysis of Counter-Tactics

• Counter soldier rush with enhanced soldier rush
• Against knights rush
  – Expand territory when defendable
  – Quickly develop strong units
  – Use catapults
Reinforcement Learning with Dynamic Scripting
Dynamic Scripting

Knowledge Base A

Knowledge Base B

Script A

Script B

computer-controlled team

human-controlled team

Combat

generate script

script control

weight updates

human control
Validation Experiments

• How quick is dynamic scripting able to adapt to an unchanging tactic?

• Or, can dynamic scripting quickly force a (human) player to change tactics?
Encounter Setup

• Two parties of four characters
  – Two fighters
  – Two wizards

• Each character
  – Static armament and weaponry
  – Two (out of three possible) potions
  – Potion assignment according to script

• Each wizard
  – Seven (out of 21 possible) spells
  – Spell assignment according to script
// OFFENSIVE WIZARD SCRIPT

if healthpercentage < 50 then
drink( "Potion of Healing" );

if roundnumber < 1 then
cast( "Mirror Image" );

if distance( closestenemy( "Wizard" ), furthestenemy( "Wizard" ) ) < 200 then
cast( "Fireball", centrenemy( "Wizard" ) );

if distance( closestenemy( "Fighter" ), furthestenemy( "Fighter" ) ) < 200 then
cast( "Stinking Cloud", centrenemy( "Fighter" ) );

cast( strongoffensive, closestenemy );

if distance( closestenemy ) > 200 then
rangedattack( defaultenemy );

meleeattack( closestenemy );
Rulebase

- One rulebase for every opponent
  - 50 rules for wizards
  - 20 rules for fighters
- Script extracted from rulebase
  - 10 rules + 2 static rules for wizards
  - 5 rules + 1 static rule for fighters
- Selection according to rule weights
- Ordering according to priority/weight
Party Fitness

• Indicates how well the party functions as a whole

• Takes into account
  – Win or loss
  – Number of surviving party members
  – Total health of surviving party members

• Is used to decide when one party outperforms the other
Individual Fitness

• Indicates how well an individual functions in the party

• Takes into account
  – Party fitness
  – Individual’s health at the end of the encounter
  – Individual’s time of death
  – Damage done to enemies

• Determines how the weights in the individual’s rulebase will be changed
Reward/Penalty Determination

- Maximum reward at individual fitness 1
- Maximum penalty at individual fitness 0
- No reward or penalty at fitness $b$ (break-even)
- Linear extrapolation

\[
\Delta W = R_{\text{max}} (1 - F) + P_{\text{max}} F
\]
Tactics

• Basic
  – Offensive
  – Disabling
  – Cursing
  – Defensive

• Composite
  – Random team
  – Random agent
  – Consecutive
Party Fitness Progression

- Absolute

- Average over last 10 encounters
<table>
<thead>
<tr>
<th>Tactic</th>
<th>$P_{\text{max}}=30$</th>
<th></th>
<th></th>
<th></th>
<th>$P_{\text{max}}=70$</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg.</td>
<td>StD.</td>
<td>High</td>
<td>Top5</td>
<td>Avg.</td>
<td>StD.</td>
<td>High</td>
<td>Top5</td>
</tr>
<tr>
<td>Offensive</td>
<td>58</td>
<td>35</td>
<td>314</td>
<td>155</td>
<td>53</td>
<td>25</td>
<td>120</td>
<td>107</td>
</tr>
<tr>
<td>Disabling</td>
<td>12</td>
<td>5</td>
<td>51</td>
<td>31</td>
<td>13</td>
<td>8</td>
<td>79</td>
<td>39</td>
</tr>
<tr>
<td>Cursing</td>
<td>137</td>
<td>334</td>
<td>1767</td>
<td>1461</td>
<td>44</td>
<td>50</td>
<td>304</td>
<td>222</td>
</tr>
<tr>
<td>Defensive</td>
<td>31</td>
<td>19</td>
<td>93</td>
<td>77</td>
<td>24</td>
<td>15</td>
<td>79</td>
<td>67</td>
</tr>
<tr>
<td>Rnd. Team</td>
<td>56</td>
<td>74</td>
<td>595</td>
<td>310</td>
<td>51</td>
<td>65</td>
<td>480</td>
<td>271</td>
</tr>
<tr>
<td>Rnd. Agent</td>
<td>53</td>
<td>67</td>
<td>398</td>
<td>289</td>
<td>41</td>
<td>41</td>
<td>251</td>
<td>178</td>
</tr>
<tr>
<td>Consecutive</td>
<td>72</td>
<td>100</td>
<td>716</td>
<td>424</td>
<td>52</td>
<td>56</td>
<td>393</td>
<td>238</td>
</tr>
</tbody>
</table>
Good Results?

- Low turning points
  - In the dozens instead of thousands
  - Generated with initial weights all equal
- Turning points and standard deviations even much lower with biased initial weights
Reinforcement Learning with Monte Carlo Control
Q-learning and Games

• Game states do not satisfy the Markov property*
  – Agents in games cannot fully observe the behaviour of other agents
  – Agents are not fully aware of other agents’ actions
  – State transitions depend on actions of all agents
• Therefore Q-learning and related reinforcement learning algorithms cannot be expected to function well for games (Sutton & Barto, 1998)
  – We confirmed this for our simulation

* Markov property: the conditional probability distribution of future states of the process, given the present state, depends only upon the current state
Monte Carlo Control

• Sutton & Barto (1998): Monte Carlo Control is
  – Particularly suitable for non-Markovian environments
  – Able to quickly start exploiting learned behaviour

• Monte Carlo Control should therefore be ideally suitable for learning in games
MC Control Algorithm

- Identify suitable states
- Create state-action space
- Assign random $Q$-value to each action in each state
- Loop
  - Choose actions
    - Most of the time (e.g., 95%): action with highest $Q$-value in state (exploitation)
    - Else: random action in state (exploration)
  - When fitness can be calculated
    - For each selected action: calculate new $Q$-value by “averaging in” the calculated fitness
- Converges to $Q$-values that represent fitness of actions in defined states
Simulation State-Action Space

• States identified by
  – #agents alive in friendly team
  – #agents alive in enemy team
• Total of 16 states (1-4 agents in each team)
  – Simulation terminates when at least one team has zero agents
• Actions defined as rules from the rulebases
  – Removing tests on “initial encounter”
  – Removing all rules with random behaviour (30%)
  – Allow agents to select spells “on the fly”
Properties of MCC

• Disadvantages compared to DS
  – (Much) higher turning points
  – Cannot deal with changing tactics

• Advantages over DS
  – Can discover effective state-action pairs unforeseen by game developer
  – Does not need priority mechanism
  – Will converge in a static environment
Dynamic Player Modeling
Two Forms of Player Modeling

• Implicit
  – AI adapts while playing against a specific opponent
  – Example: dynamic scripting

• Explicit
  – AI creates a model of the opponent, then chooses actions which it knows work well against such an opponent
  – Naturally, it can learn those actions, and can still adapt online
  – Very suitable for multi-player games
Player Model

- **Modeling**
  - Player actions
  - Player preferences
- **Model types**
  - Evaluation functions
  - Neural networks
  - Finite state machines
  - Probabilistic models
  - Case-based models
- **Model application**
  - Strong AI
  - Appropriate AI
Formations

• Cohesive team behaviour
  – V-shape
  – Phalanx
  – Wedge
  – Shield wall
  – Etc...

• Should fit the enemy’s tactics
Formations in RTS Games

• Static: each unit assigned a specific slot; or
• Emergent: units positioned relative to each other
Dynamic Formations

• General formation framework
• Learning algorithm to create suitable formations
  – Online adaptation of 8 formation parameters
  – Offline learning of suitable parameters against player models
• Fast modeling algorithm to determine a model for the current player
  – Player model based on 3 observations

Note: This research was performed by Marcel van der Heijden, Sander Bakkes, and Pieter Spronck
Dynamic Formation Shape

• Centered around leader

• Parameters 1-5
  – 1-n lines
  – 1-m units per line
  – $\alpha$, $\beta$, $\gamma$
Replacement of Destroyed Units

- Neighbouring units take place of destroyed units
  - Fast and cheap
Movement

- Parameter 6: speed
- Leader moves
- Formation moves with leader
- Units attempt to stick to their place in the formation
- If units fall seriously behind, the speed of the formation is adjusted
Target Selection (Parameter 7)

(a) Relative

(b) Leader

(c) Centre

(d) Shortest
Combat Behaviour (Parameter 8)

• Choice of five
  – Overrun – keep pushing forward
  – Hold – stop when opponent is within weapon range
  – Retreat – hit and run
  – Bounce – hit an run, but return after half of time-out
  – Border- stay out of weapon range during time-out
Performance Measures

• Absolute performance: percentage of games won during last 50 games
  – Sequence of 200 games against each opponent
• Relative performance: fitness value obtained
  – Range: [-50,50]
• Turning point: win-loss ratio of last 20 games becomes 15-5 or better
  – Statistically, it is 98% certain that learning opponent is now better than static opponent
Dynamic Formations w/o Modelling

(a) Blekinge

(b) NUS

(c) UBC

(d) UM

(e) WarsawA

(f) WarsawB
Player Model in 3 Parameters

• Number of formations
  – Determined with k-means clustering

• Unit distribution
  – Proportion of width and height of rectangle around all units

• Unit distance (density)
  – Average distance of each unit to its closest neighbour
Player Modelling During Game

• Measure the three parameters ~100 timesteps after the game started
• Calculate the combined likelihood of observed parameters for each known model (Bayes’ theorem)
• Select the model that best explains the observations
  – For the first few trials
  – After that, do regular adaptation
## Performance Comparison

<table>
<thead>
<tr>
<th>Opponent</th>
<th>No player modelling</th>
<th>With player modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Abs. perf.</td>
<td>Rel. perf.</td>
</tr>
<tr>
<td>Blekinge</td>
<td>100%</td>
<td>37</td>
</tr>
<tr>
<td>NUS</td>
<td>12%</td>
<td>-31</td>
</tr>
<tr>
<td>UBC</td>
<td>94%</td>
<td>15</td>
</tr>
<tr>
<td>UM</td>
<td>66%</td>
<td>4</td>
</tr>
<tr>
<td>WarsawA</td>
<td>44%</td>
<td>-1</td>
</tr>
<tr>
<td>WarsawB</td>
<td>98%</td>
<td>17</td>
</tr>
</tbody>
</table>
Adapting to Entertain
Learning and Enjoying Chess

Deep Blue (1997)

Chess Challenger (1978)
Iida’s Theory of Refinement

Seesaw game:
Optimal length of time outcome is uncertain

\[ \frac{\sqrt{B}}{D} = 0.07 \]

Complexity
Noble uncertainty:
New tactics are possible

Fairness
Draw ratio:
Matching opponents
Enjoying Games

• Computer should be able to play stronger than the human player
• Computer should adapt to the level of skill of the human player
• Computer should constantly offer new challenges

In short: Computer and human increase their playing skill in parallel
• Manual
• Coarse
• Simple
Automatic Scaling of Game AI

- **High-fitness penalising**
  - Award the highest fitness to the “most equal” AI, instead of to the “best” AI

- **Weight clipping**
  - Increase AI variety when the computer plays too well
  - Decrease AI variety when the computer plays badly

- **Top culling**
  - Remove the currently “best” knowledge when the AI plays too well
  - Reactivate the “best” knowledge when the AI plays badly
Difficulty Scaling Tests

• Against five different basic tactics
• Against three different composite tactics
• Against Neverwinter Nights game AI

• Novice tactic
  – Imitates novice player
  – Knows obvious good tactics
  – Does not know subtle good tactics
Histograms

Optimisation

High Fitness Penalising

Weight Clipping

Top Culling
Scaling Results

• *Without automatic scaling*, dynamic scripting wins against all tactics
• With *high-fitness penalising* an even game is achieved against 2 out of 8 tactics
• With *weight clipping* an even game is achieved against 7 out of 8 tactics
• With *top culling* an even game is achieved against 8 out of 8 tactics, combined with the lowest standard deviation
Difficulty Scaling

• Adaptive effectiveness can be easily converted to difficulty scaling

• Often not useful against strong players
  – Losing a sense of accomplishment
  – Best results against novices

• For highest enjoyment
  – Should the AI play stronger than the human player?
  – How much stronger?
  – To what extent does this depend on the type of game?
  – Is this equal for all human players?
Adapting to the trainee
Serious Gaming
Agent Approach

- Agents part of the design process
- Reasoning agents
- Adapting agents

Example:
- Trainee is fire commander
- 2 fireman agents
- 1 agent controlling spreading of fires
- 1 victim agent
Adapt the game to the user

A user only learns if performing on his own level
Current approaches

• Fixed difficulties
• Central control or no coordination
• Mainly adjust simple subtasks
Dynamic Difficulty Adjustment

- Online adaptation:
  Continuously balance challenges in the game with (developing) skills of the trainee
Aspects

• User
  – Evolving skills (when learning)

• Characters
  – Characters adapt independently
  – Characters active for long periods, so, adaptation should be believable

• Keep storyline
  – Learning goals have to be maintained!

• Adaptation must be coordinated!

• Performance can not be measured separately for each skill and influence of each agent
Adapt the game to the user

Use agents to have characters adapt natural and independent
Adapt the game to the user.

Use agents to have characters adapt naturally and independently.
Adapt the game to the user

Use an agent organization approach to ensure the goals of the overall system
Story-line

• Guarantee certain states are reached
• Subtasks defined by scene scripts and landmarks
• Connected by interaction structure
  – Describes game progress
  – Connecting scenes
  – Tasks in parallel

Start → Get to site

Gather info → Search building

Secure area → Evacuate victims

→ Extinguish fire → Clear area → End
Agent Implementation

- Adaptable BDI-agents (2APL extension)
  - Equivalent plans
  - Preference relation
  - Environmental information also used
Adaptation Engine

• Coordinates task difficulty
• Check with game model
• Combinatorial auction
  – User model
  – Agent preferences
Task Difficulty

- Task dependent on behavior of the agents
- Behavior variations are fixed
- Estimated by domain expert
- Updated by offline learning
- Much faster adaptation
Agent Perspective

• Agents Propose actions to adaptation engine at “natural” synchronization points
• Created to facilitate trainee’s objectives (optimize agent behavior relative to trainee’s performance!)
• Not responsible for suitable combination
• Conflict:
  – Stay as consistent as possible
  – Propose enough actions
• Adaptation engine can request agents to terminate behavior if necessary for coordination
Conclusion

• Continuous adaptation to the trainee
• Agent based approach
  – Complex individual behavior and adaptation possible
• Agent organisation for coordination
  – Balance between individual flexibility and global story line maintaining learning goals
  – Constrain based only on global landmarks and learning goals
  – Minimal central control for more efficiency and more flexibility
Overall conclusions

• Adaptation and learning can take place in many parts and levels of the games
• Traditional techniques are used for tuning parts of the game
• Agent based approach is useful for adapting the story line of a game to the trainee