Motivated Reinforcement Learning

Curious Characters for Multiuser Games

Fangkai Yang

Computational Science and Technology
KTH Royal Institute of Technology

04/25/2017
Who is Fangkai

• PhD candidate.

• Research on real-time virtual characters and crowd simulation.

• Game developer:
  
  Just Cause 3
  
  War Rage

• fangkai@kth.se
Outline

• Non-player Characters and Reinforcement Learning
• Developing Curious Characters Using Motivated Reinforcement Learning
• Curious Characters in Games
Non-Player Characters: characters controlled by the computer through artificial intelligence

Enemies: characters that oppose human players in a pseudo-physical sense by attacking the virtual human player with weapons or magic.

Partners: opposite role to enemies, and attempt to protect or help players.

Support: support the storyline of the game by offering quests, advice, goods for sale or training.
Non-Player Characters in Multiuser Games

Massively Multiplayer Online Role-Playing Games (MMORPGs): a very large number of players interact with NPCs and each other within a persistent virtual world.

Multiuser Simulation Games: characters can respond to certain changes in their environment with new behaviors.

Open-Ended Virtual Worlds: (text-based), object-oriented multiuser dungeons (MMOs).
Artificial Intelligence Techniques for NPCs

**Reflexive Agents:** use state machines and rule-based algorithms, have been common in enemy and support characters.

**Learning Agents:** modify their internal structure in order to improve their performance with respect to some task, have been used in partners and some enemy characters.

**Evaluationary Agents:** use evolutionary approaches such as genetic algorithms to simulate the process of biological evolution by implementing natural selection, reproduction, and mutation.

**Smart Terrain:** discards the character-oriented approach to reasoning using AI and embeds the behaviours and actions associated with a virtual object within the object itself.
Reflexive Approaches for NPCs

Rule-based approach: defines a set of rules about states of the game world.

*If* <condition> *then* <action>

An example rule from a warrior character in *Baldur’s Gate*
**State Machine:** divide a NPC’s reasoning process into a set of internal states and transitions. Each state contains a number of events constructs that cause actions to be taken.

An example of part of a state machine for a *Dungeon Siege* Gremel.
Fuzzy Logic: provides a way to infer a conclusion based on facts that may be vague, ambiguous, inaccurate or incomplete.

If \( <X \text{ is } A> \text{ then } <Y \text{ is } B> \)

\( X, Y \): linguistic variables representing characteristics being measured – such as temperature, speed or height.
\( A, B \): fuzzy categories – such as hot, fast, tall.

Difference:
- Balls are targets for kicking in State Machine.
- Any object fits the description of "being round" as a target for kicking in Fuzzy Logic.
Learning Approaches for NPCs

**Decision Tree:** hierarchical graphs learned from a training set of previously made decisions. Internal nodes in the tree represent conditions about states of the environment, while leaf nodes represent actions. The action can be taken when all conditions on the path to leaf node are fulfilled.

**Neural Networks:** examples of correct actions in different situations are fed into network to train a character. When a character encounters a similar situation it can make a decision about the correct action to take.

**Reinforcement Learning:** RL agents learn from trail-and-error and reward. The agent records the reward signal by updating a behavioural policy, and chooses an action which attempts to maximise the long-run sum of the values of reward.
Motivation: the reason one has for acting or behaving in a particular way.

- **Biological Motivation:** explain behaviour in terms of energies and drives that push an organism towards certain behaviour. Design of NPCs such as *enemies* (which have a predator—prey relationship with player) and *support characters* (e.g. animal herds).

- **Cognitive Motivation:** abstract computational structures such as states, goals, and actions that form the basis of cognitive inspired computational models of motivation. Design of humanoid characters capable of advanced planning or learning.

- **Social Motivation:** what individuals do when they are in contact with one another.

- **Combined Motivation:** unified approach to motivation: comprehensive algorithms that describe the causes of action at the simulated biological, abstract reasoning and multiagent level.
Biological Motivation

**Drive Theory:** homeostatic requirements drive an individual to restore some optimal biological condition when stimulus input is not congruous with that condition.

\[ \text{Tendency for a behavioural response} = \text{Habit} \times \text{Drive} \]

**Motivational State Theory:**
extends one-dimensional drives to multidimensional motivational states.

**Arousal:** pushes individuals to maintain a level of internal stimulation.
Curiosity: motivated by a need to bring stimulation nearer to some optimal level.
- Under-stimulated (Boredom): an individual seeks out new stimuli to replace the habituated ones
- Over-stimulated: an individual seeks out familiar or simple stimulation and ignore the remainder.

Operant: motivated by important goals by perceptions and cognitions. When an individual does something that is rewarded, it is not influenced by any real or imagined loss of drive but by the idea of being rewarded.

Achievement: motivated on the expectancy of attaining a goal. Motivation to succeed or to avoid failure.

Intrinsic: motivated to satisfy the desire to feel self-determining and competent, i.e. Skydiving "for fun".
**Social Motivation**

**Conformity:** an individual engages in because of a real or imagined group pressure.

**Cultural Effect:**

- what skills and thoughts are cognitively available to an individual (eat insects as a means of satiating hunger).
- what selections an individual will make from those that are cognitively available (not eat insects even if be informed).

**Evolution:** a society of individuals with computational models of chromosomes that can combine and mutate. It allows adaptation to occur over generations that failure or destruction of a single individual can be tolerated and be used for learning within the society.
Combined Motivation

Maslow’s Hierarchy of Needs:

- **Self-actualisation** (Achieving individual potential)
- **Esteem** (Self-esteem and esteem from others)
- **Belonging** (Love, affection, being part of a group)
- **Safety** (Shelter, removal from danger)
- **Physiological** (Health, food, sleep)

Existence Relatedness Growth Theory (ERG):

- **Existence Needs**
  - Satisfaction/Progression
  - Frustration/Regression
- **Relatedness Needs**
- **Growth Needs**
  - Satisfaction/Strengthening
Reinforcement Learning: Learn what to do by trial-and-error. RL agents learn how to map situations to actions so as to maximize a numerical reward signal.

- Dynamic Programming
- Monte Carlo Methods
- Temporal Difference Learning

Challenges:
- Dynamic programming is inappropriate in many complex or unpredictable environments such as virtual worlds.
- Monte Carlo Methods are not suited for step-by-step, incremental computation (lifelong learning).
- Typically rule-based representation (fixed, task-oriented) of reward limits the learning in dynamic virtual worlds where tasks may only be relevant for short periods and new tasks may arise.
Partially Observable Environments: sensed states are subsets of the actual world states. Partial observability can be an advantage which permits the agent to focus attention by deliberately sensing only part of the world states or sensed states, ignoring not relevant stimuli.

Function Approximation: represent the value function or action-value function as a parameterized functional form with parameter vector. Changing one parameter changes the estimated value of many states.

Hierarchical Reinforcement Learning: improves the scalability of RL in structured environments by creating temporal abstractions of repeated structures in the state space which can be recalled and reused during learning.
Motivated Reinforcement Learning (MRL) introduces motivation signal into the RL framework.

- **Category (I):** use a motivation signal in addition to a reward signal.
  - Direct learning by identifying subtasks of the task defined by the reward signal.
  - Use motivation as an automatic attention focus mechanism to speed up existing RL algorithms.

- **Category (II):** use a motivation signal instead of a reward signal.
  - Achieve NPCs capable of adaptive, multitask, online learning.
  - Identify novel design tasking and search for novel solutions to those tasks.

**Motivation signal:** be computed online as a function of an agent’s experiences using a computational model of motivation.

**Reward signal:** a set of predefined rules mapping values to known environmental states or transitions.
**MRL(I) models** incorporate both a reward signal from the environment and a motivation signal with RL.

Huang and Weng define the motivation signal using a computational model of novelty:

$$N(t) = \sqrt{\frac{1}{|S|} \sum_{i=1}^{[S]} \frac{(s_i'(t) - s_i(t+1))^2}{\sigma_i^2}}$$

Primed sensations are computed using an Incremental Hierarchical Discriminant Regression (IHDR) tree that derives the most discriminating features from sensed states.

To overcome the case of random occurrences regarded as high novelty, human teacher is incorporated to direct the robot’s learning through the provision of ‘good’ and ‘bad’ reward.

$$\{ R_{m(t)}, R_{(t)} \} = \alpha F^+ + \beta F^- + (1 - \alpha - \beta) N(t)$$

Using a Motivation Signal in Addition to a Reward Signal

Schmidhuber used the predictability of a learned world model to represent curiosity and boredom as reinforcement and pain units in curious neural controllers.

\[ P_{t+1} = s_{t+1}(1 - s_{t+1}) \left( \sum_n w_n P_{t+1} + \delta s_{t+1} \right) \]

The model is designed to identify states where the model network’s prediction performance is not optimal as the most highly motivating, in order to encourage an agent to revisit those states and improve its network model.

Maximum motivation is generated for moderate levels of predictability to represent curiosity about states in which an "ideal mismatch" occurs between what is expected and what is sensed. i.e. zero motivation for maximum predictability and for very low predictability to simulate boredom.

MRL(II) models incorporate a motivation signal with RL instead of the reward signal from the environment.

Huang and Weng use a Habituated Self-Organising Map (HSOM) to represent the set of sensed states and model novelty. However, it suffers the similar problems previously that may contain random occurrences.

Kaplan and Oudeyer used an approach designed to motivate a search for situations that show the greatest potential for learning. These situations are defined by: predictability, familiarity and stability of the sensory-motor context of a robot.

**Sensory-motor vector:** 
$$SM(t) = (s_{1(t)}, s_{2(t)}, \ldots, s_{|S|}(t), \ldots, A_{1(t)}, A_{2(t)}, \ldots).$$

**Predictability:** current error for predicting the sensed state given the sensory-motor vector 
$$P(t) = 1 - e(SM(t-1), S(t)).$$

**Familiarity:** a measure of how common the transition is between sensory-motor vector and the sensed state.
$$\Gamma(t) = f_\tau(SM(t-1), S(t)).$$

**Stability:** a measure of the distance of an observation in the sensed state from its average value in a recent period.
$$\sigma_i(t) = 1 - \sqrt{(s_{i(t)} - s_{i(\tau)})^2}$$

Using a Motivation Signal Instead of a Reward Signal

The motivation signal is constructed from predictability, familiarity and stability using the intuition that reward should be the highest when stability is maximized and when predictability and familiarity are increasing.

\[
R_{m(t)} = \sigma_{1(t)} + \sigma_{2(t)} + \ldots + \begin{cases} 
\Gamma(t) - \Gamma_{(t-1)} : \Gamma(t) > \Gamma_{(t-1)} \\
0 : \Gamma(t) \leq \Gamma_{(t-1)} 
\end{cases} + \begin{cases} 
P(t) - P_{(t-1)} : P(t) > P_{(t-1)} \\
0 : P(t) \leq P_{(t-1)} 
\end{cases}
\]

Increasing predictability and familiarity precludes highly novel stimuli like random occurrences from being highly motivating unless they become more predictable and familiar and thus less random.

Evaluate the behavior of NPCs in a complex problem:

- Believable, realistic or intelligent behavior
- Support for game flow
- Player engagement and satisfaction

Games in the flow zone offer an optimal level of challenge for a player’s ability. This avoids player boredom or anxiety and increases enjoyment.
Behavioral cycles of states and actions can be illustrated using finite state automata.

(a) shows a behavioral cycle of complexity one for a maintenance task satisfied in the state $S_1$. (b) Shows a behavioral cycle of complexity $n$ for $n$ achievement tasks.

The complexity of a behavioral cycle refers to the number of actions required to complete a cycle that starts and finishes in a given state.
There are established performance metrics for RL algorithms where the reward is task-specific, but performance metrics for MRL algorithms vary according to the model of motivation and the domain of application (be measured without reference to a specific, known task).

Statistical model to identify learned tasks in order to evaluate learning in adaptive, multitask learning settings:

\[
c_v(K) = \sqrt{\frac{1}{h-1} \sum_{i=1}^{h} (a_i - \bar{a}_K)^2}
\]

A task \( K \) is learned when \( c_v(K) \) is less than some error threshold for the first time.
Behavioral Variety

Behavioral variety evaluates the behavior of an agent by measuring the number of behavioral cycles for different tasks. The measurement is made by analyzing the agent’s experience trajectory at time $t$:

$$V(t) = \text{count_unique}(K) \text{ where } c_t(K) < r$$

Multitask learning can be visualized as instantaneous behavior variety.
**Behavioral Complexity**

**Behavioral complexity** evaluates learning performance by measuring the complexity of a learned task in terms of the average length of the behavioral cycle required to repeat the task. The complexity of the task can be measured as the mean number of actions required to repeat $K$:

$$X_K = \bar{a}_K$$

Multitask learning can be compared in terms of the different maximum complexity of tasks learned by a certain time $T$.

Confidence intervals without overlap indicate a statistically significant difference in multitask learning ability.

Multitask learning can be visualized in terms of maximum behavior complexity.
Developing agents that can learn in complex, dynamic environments requires a representation of the world or environment states and a flexible labelling structure to accommodate the appearance and disappearance of elements.

This can be achieved with the partially observable Markov decision process (POMDP) formalism and a context free grammar (CFG).
In dynamic environments, the traditional, fixed-length vector representation for sensations becomes inappropriate as it does not allow the addition or removal of MDP elements. The sensed state can be represented as a string from a CFG \((V_S, \Gamma_S, \Psi_S, S)\):

- \(V_S\) is a set of variables or syntactic categories;
- \(\Gamma_S\) is a finite set of terminals such that \(V_S \cap \Gamma_S = \emptyset\);
- \(\Psi_S\) is a set of productions (or rules) of the form \(V \rightarrow \nu\) where \(V\) is a variable and \(\nu\) is a string of terminals and variables;
- \(S\) is the start symbol.

\[
\begin{align*}
S & \rightarrow <\text{sensations}> \\
<\text{sensations}> & \rightarrow <P_1\text{Sensations}><\text{sensations}> | \varepsilon \\
<P_1\text{Sensations}> & \rightarrow <s_j><P_1\text{Sensations}> | \varepsilon \\
<s_j> & \rightarrow <\text{number}> | <\text{string}> \\
<\text{number}> & \rightarrow 1 | 2 | 3 | ... \\
<\text{string}> & \rightarrow ... 
\end{align*}
\]
The action space can also be represented using a CFG $(V_A, \Gamma_A; \Psi_A, A)$

- $V_A$ is a set of variables or syntactic categories;
- $\Gamma_A$ is a finite set of terminals such that $V_A \cap \Gamma_A = \emptyset$;
- $\Psi_A$ is a set of productions of the form $V \rightarrow v$ where $V$ is a variable and $v$ is a string of terminals and variables;
- $A$ is the start symbol.

\[
\begin{align*}
A & \rightarrow \text{<actions>} \\
\text{<actions>} & \rightarrow \text{<P1Actions><actions>} \mid \varepsilon \\
\text{<P1Actions>} & \rightarrow \text{<A1><P1Actions>} \mid \varepsilon \\
\text{<A1>} & \rightarrow \ldots
\end{align*}
\]
A General Experience-Based Motivation Function

Modelling motivation for experience-based attention focus.
An observation is essentially a (unordered) combination of sensations from the sensed state.

Observations containing fewer sensations have greater spatial selectivity as they describe only a small proportion of the state space, vice versa.

---

**Table: Observation Function**

<table>
<thead>
<tr>
<th>Observation Function 1</th>
<th>Observation Function 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O_1(t) = (a_{1(t)}, 0, 0, 0, \ldots) )</td>
<td>( O_1(t) = (a_{1(t)}, a_{2(t)}, \ldots, a_{L(t)}, \ldots) )</td>
</tr>
<tr>
<td>( O_2(t) = (0, a_{2(t)}, 0, 0, \ldots) )</td>
<td>( O_2(t) = (0, 0, 0, \ldots) )</td>
</tr>
</tbody>
</table>

**Rule**

\[ a_i(t) = \begin{cases} s_{i(t)} & \text{if } i = L \\ 0 & \text{otherwise} \end{cases} \]

**Number of Observations**

| \( |O(t)| \) | 1 |

**Attention Focus**

| High: each observation focuses on one sensation. | Low: each observation focuses on all sensations. |
Events differ from actions in that a single action may cause a number of different transitions, depending on the situation in which it is performed while an event describe a specific transition. Events are represented in terms of the difference between two sensed states.

\[ S(t) - S(t') = (\Delta(s_{1(t)}, s_{1(t')}), \Delta(s_{2(t)}, s_{2(t')}), \ldots, \Delta(s_{L(t)}, s_{L(t')}), \ldots) \]

<table>
<thead>
<tr>
<th>Difference Function 1</th>
<th>Difference Function 2</th>
<th>Difference Function 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ s_{L(t)} \text{ if } \exists s_{L(t')} ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ -s_{L(t')} \text{ if } \exists s_{L(t)} ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ s_{L(t)} - s_{L(t')} \text{ if } s_{L(t)} - s_{L(t')} \neq 0 ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 if ( s_{L(t)} &gt; s_{L(t')} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1 if ( s_{L(t)} &lt; s_{L(t')} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 if ( s_{L(t)} = s_{L(t')} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>null otherwise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>null otherwise</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Output Range | \((-\infty, \infty)\) | \([-1, 1]\) |
| Attention Focus | Low: all information about the size and direction of change in a sensation is available. | Moderate: attention focused on whether a sensation has increased or decreased. | High: attention focused only on whether a sensation has changed or not. |

<table>
<thead>
<tr>
<th>Event Function 1</th>
<th>Event Function 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>[ E_{1(t)} = (\Delta(s_{1(t)}, s_{1(t')}), 0, 0, 0, \ldots) ]</td>
</tr>
<tr>
<td>Recognised</td>
<td>[ E_{2(t)} = (0, \Delta(s_{2(t)}, s_{2(t')}), 0, 0, \ldots) ]</td>
</tr>
<tr>
<td>Rule</td>
<td>[ E_{L(t)} = (0, \ldots, \Delta(s_{L(t)}, s_{L(t')}), 0, \ldots) ]</td>
</tr>
<tr>
<td>[ e_{e(t)} = \begin{cases} \Delta(s_{e(t)}, s_{e(t')}) &amp; \text{if } e = L \ 0 &amp; \text{otherwise} \end{cases} ]</td>
<td></td>
</tr>
<tr>
<td>[ E_{L(t)} = \Delta(s_{L(t)}, s_{L(t')}) ]</td>
<td></td>
</tr>
</tbody>
</table>

| Number of Events | \( |S(t) - S(t')| \) | 1 |
| Attention Focus | High: each event focuses on one difference. | Low: each event focuses on all differences |
Tasks and Task Selection

Two assumptions to model subsets of an experience trajectory:
- Recent experiences are likely to be the most relevant at the current time.
- Similar experience from any time in the past are likely to be relevant for determining what actions to take in the present.

**Self-organizing Maps (SOMs):**
SOM neurons represent the current set of tasks to learn and observations/events are input for the SOM. The SOM update function progressively modifies each neuron $K$ to model tasks that are relevant to the most recent observations or events, but also influenced by past observations or events.

**K-means clustering:**
A set of centroids represent the current set of tasks to learn and observations/events represent input. The K-means update function progressively modifies each centroid $K$ to model tasks that are relevant to the most recent observations or events, while influenced by past observations or events.
Saunders modelled interest by applying the Wundt curve:

\[
R(N(t)) = F^+(N(t)) - F^-(N(t)) = \frac{F_{\max}^{+}}{1 + e^{-\rho^+(2N(t) - F_{\min}^+)}} - \frac{F_{\max}^{-}}{1 + e^{-\rho^-(2N(t) - F_{\min}^-)}}
\]

It peaks at a maximum value for the most interesting events are those that are similar-yet-different to previously encountered experiences.

R. Saunders, Curious design agents and artificial creativity, Faculty of Architecture, University of Sydney, Sydney, 2001.
Arbitration function output the motivation signal by arbitrates between the motivation values produced for different tasks or by different motivation functions.

### Multiple computational models of motivation

<table>
<thead>
<tr>
<th>Arbitration Function 1</th>
<th>Arbitration Function 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{m(t)} = \max_i R_i(K_{(t)})$</td>
<td>$R_{m(t)} = \sum_i w_i R_i(K_{(t)})$</td>
</tr>
<tr>
<td>Motivations compete to influence learning.</td>
<td>Motivations co-operate to influence learning.</td>
</tr>
</tbody>
</table>

### Multiple motivating tasks

<table>
<thead>
<tr>
<th>Arbitration Function 1</th>
<th>Arbitration Function 2</th>
<th>Arbitration Function 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{m(t)} = \max_{K \in K_{soj}} R(K)$</td>
<td>$R_{m(t)} = \text{avg}<em>{K \in K</em>{soj}} R(K)$</td>
<td>$R_{m(t)} = \sum_{K_i \in K_{soj}} w_i R(K_i)$</td>
</tr>
<tr>
<td>Tasks compete to motivate learning.</td>
<td>Tasks cooperate equally to motivate learning.</td>
<td>Tasks cooperate to motivate learning.</td>
</tr>
</tbody>
</table>
A General Experience-Based Motivation Function

Modelling motivation for experience-based attention focus.

1. Observe $O_{(t)}$ from $S_{(t)}$ using the observation function
2. Subtract $S_{(t)} - S_{(t-1)}$ using the difference function
3. Compose $E_{(t)}$ using the event function
4. Distinguish $K_{(t)}$ using the task selection function
5. Repeat (for each $K_i \in K_{(t)}$):
   6. Repeat (for each $R_j \in R$):
      7. Compute $R_j(K_i)$ using experience-based reward fn $R_j$
8. Arbitrate over $R_j(K_i) \forall i, j$ to produce $R_{(t)}$
Curiosity as Interesting Events:
Curiosity is a kind of motivation that is based on interesting events in the environment. A curious NPC will be able to respond to changes in the environment by shifting his attention to novel events and focus on behaviors that reinforce that change.

\[
\Delta(S_L(t), S_L(t-1)) = \begin{cases} 
S_L(t) & \text{if } \exists s_{L(t')} \\
-S_{L(t')} & \text{if } \exists s_{L(t)} \\
S_L(t) - S_{L(t')} & \text{if } S_L(t) - S_{L(t')} \neq 0 \\
\text{null otherwise} & \text{otherwise}
\end{cases}
\]

1. Subtract \( S(t) - S(t') \) using
2. Compose \( E_{S(0)} = \{E_{(0)}\} \)
3. Distinguish \( K(t) \) using a SOM.
4. Compute \( I(K(t)) \)
5. Output \( R_{\pi(t)} = I(K(t)) \)
Curiosity as Interest and Competence:

A model of motivation based purely on interest does not always allow the agent enough time to become competent at any task.

Combining them presents a second kind of curiosity: one that allows the agent to be distracted by an interesting event when the value of being distracted is greater than the value of becoming competent at the current task.

\[
\Delta(s_L(t), s_L(t-1)) = \begin{cases} 
  s_L(t) & \text{if } s_L(t) \neq s_L(t') \\
  -s_L(t') & \text{if } s_L(t') \neq s_L(t) \\
  s_L(t) - s_L(t') & \text{if } s_L(t) - s_L(t') \neq 0 \\
  \text{null otherwise} 
\end{cases}
\]

1. Subtract \(S(t) - S(t')\) using

2. Compose \(E_{S(0)} = \{E(0)\}\) where \(E_t = (\Delta(s_1(t), s_1(t-1)), \Delta(s_2(t), s_2(t-1)), \ldots \Delta(s_L(t), s_L(t-1)), \ldots)\)

3. Distinguish \(K(t)\) using a SOM.

4. Repeat (for each \(R_j \in \{I, C\}\)):

5. Compute \(R_j(K(t))\) using the reward function \(R_j\)

6. Output \(R_m(t) = \max(I(K(t)), C(K(t)))\)
A General Motivated Reinforcement Learning Model

Difference between MRL algorithms and existing TD learning algorithms:

- The reward function implements experience-based attention focus based on computational model of motivation.
- The state-action table or equivalent structure is initialized incrementally.
- The state and action spaces are implemented using a context free grammar (CFG).

```
1. \( Y(0) = \emptyset \)
2. Repeat (forever):
3.   Sense \( S(t) \)
4.   if \( Q(S(t), A) \) not initialised:
5.     initialise \( Q(S(t), A) \) arbitrarily \( \forall A \)
6.   Choose \( A(t) \) from \( S(t) \) using the policy improvement fn
7.   Update \( Y(t) \)
8.   if \( S(t-1) \) is initialised:
9.     Compute \( Q(t) \)
10.    Update \( Q \) for \( S(t-1) \) and \( A(t-1) \) using policy eval fn
11.   \( S(t-1) \leftarrow S(t) \); \( A(t-1) \leftarrow A(t) \)
12.   Execute \( A(t) \)
```
Motivated Flat Reinforcement Learning

Flat reinforcement learning agents take a reward signal from the environments, but motivated flat reinforcement learning agents incorporate a motivation process to compute an experience-based reward signal.

(a) Flat reinforcement learning agents          (b) motivated flat reinforcement learning agents
Motivated Flat Reinforcement Learning

Q-learning can be thought of as the more aggressive learning approach. SARSA can be thought of as the more cautious learning approach.

(a) The motivated Q-learning algorithm

(b) The motivated SARSA algorithm
Recall is implemented in a MRL setting by integrating motivated reflexes with option learning to create motivated, multioption reinforcement learning.
Motivated Multioption Reinforcement Learning

An option \( B = (I, \pi, \Omega) \) is a temporal abstraction that is initiated, takes control for some period of time and then eventually ends.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B.I )</td>
<td>Initiation set</td>
<td>Precup et al.'s [1]</td>
</tr>
<tr>
<td>( B.\pi )</td>
<td>Option policy</td>
<td>original option</td>
</tr>
<tr>
<td>( B.\Omega )</td>
<td>Termination function</td>
<td>framework.</td>
</tr>
<tr>
<td>( B.K )</td>
<td>Task to learn</td>
<td>Additional structures for MMORL.</td>
</tr>
<tr>
<td>( B.A )</td>
<td>Last action selected by this option</td>
<td></td>
</tr>
<tr>
<td>( B.t )</td>
<td>Number of actions selected by this option since the last occurrence of ( B.K )</td>
<td></td>
</tr>
</tbody>
</table>

The MMORL model incorporates three reflexes for creating, disabling and triggering behavioral options.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
</table>
| 8. if \( R_{e(t)} > \Phi_1 \):
| 9. \[ \text{Repeat (for each } K \in K_{s(t)} \text{):} \]
| 10. \( B_{(t)} = B_{(t-1)} + B_{x(t)} \) | Rule defining when to create new behavioural options. |
| 11. if \( B_{(t-1)} \notin A \) and \( B_{(t-1)} \cdot t > \Phi_2 \):
| 12. \( B_{(t)} = B_{(t-1)} - B_{x(t-1)} \) | Rule defining when to disable existing behavioural options. |
| 13. if \( (B_{(t-1)} \in A \text{ or } B_{(t-1)} \cdot \Omega(S_{(t-1)}) = 1) \):
| 14. if \( R_{e(t)} > \Phi_3 \text{ and } \exists B \text{ for } K_{(t)} \text{ and } B.t < \Phi_2 \):
| 15. \( B_{(t)} = B \) | Rule defining when to trigger existing behavioural options. |

Motivated Reinforcement Learning – Curious Characters for Multiuser Games
MHRL further expand the policy improvement and evaluation equations to the hierarchical setting compared with MMORL algorithm (reuse and recall).
Motivated Reinforcement Learning in MMORPGs

A small-scale, isolated game scenario.
Two Markov decision processes, \( P_1 \) and \( P_2 \), describing two regions of the village.

\( P_1 \): mine iron-ore and forge weapons.
\( P_2 \): cut timber and craft furniture.
Motivated Reinforcement Learning in MMORPGs

S \rightarrow \langle\text{sensations}\rangle
\langle\text{sensations}\rangle \rightarrow \langle\text{P:sensations}\rangle<\text{P2:sensations}\rangle
\langle\text{P:sensations}\rangle \rightarrow \langle\text{P:location}\rangle<\text{P1:inventory}\rangle<\text{P:visibleObjects}\rangle
\langle\text{P:location}\rangle \rightarrow \langle\text{mine}\rangle | \langle\text{smithy}\rangle
\langle\text{mine}\rangle \rightarrow 1
\langle\text{smithy}\rangle \rightarrow 2
\langle\text{P:inventory}\rangle \rightarrow \langle\text{P:objects}\rangle
\langle\text{P:visibleObjects}\rangle \rightarrow \langle\text{P:objects}\rangle
\langle\text{P:objects}\rangle \rightarrow \langle\text{P:object}\rangle<\text{P:objects}\rangle | \varepsilon
\langle\text{P:object}\rangle \rightarrow \langle\text{pick}\rangle | \langle\text{forge}\rangle | \langle\text{smelt}\rangle | \langle\text{iron-ore}\rangle | \langle\text{iron}\rangle | \langle\text{weapons}\rangle
\langle\text{P:sensations}\rangle \rightarrow \langle\text{P:location}\rangle<\text{P:inventory}\rangle<\text{P:visibleObjects}\rangle
\langle\text{P:location}\rangle \rightarrow \langle\text{forest}\rangle | \langle\text{carpenter-shop}\rangle
\langle\text{forest}\rangle \rightarrow 3
\langle\text{carpenter-shop}\rangle \rightarrow 4
\langle\text{P:inventory}\rangle \rightarrow \langle\text{P:objects}\rangle
\langle\text{P:visibleObjects}\rangle \rightarrow \langle\text{P:objects}\rangle
\langle\text{P:objects}\rangle \rightarrow \langle\text{P:object}\rangle<\text{P:objects}\rangle | \varepsilon
\langle\text{P:object}\rangle \rightarrow \langle\text{axe}\rangle | \langle\text{lathe}\rangle | \langle\text{timber}\rangle | \langle\text{furniture}\rangle
\langle\text{pick}\rangle \rightarrow 1
\langle\text{forge}\rangle \rightarrow 1
\langle\text{smelt}\rangle \rightarrow 1
\langle\text{iron-ore}\rangle \rightarrow 1
\langle\text{iron}\rangle \rightarrow 1
\langle\text{weapons}\rangle \rightarrow 1
\langle\text{axe}\rangle \rightarrow 1
\langle\text{lathe}\rangle \rightarrow 1
\langle\text{timber}\rangle \rightarrow 1
\langle\text{furniture}\rangle \rightarrow 1
\langle\text{actions}\rangle \rightarrow \langle\text{P:actions}\rangle<\text{P2:actions}\rangle
\langle\text{P:actions}\rangle \rightarrow \text{pick-up} <\text{P:object}> | \text{move} <\text{direction}> | \text{use} <\text{P:object}>
\langle\text{P:actions}\rangle \rightarrow \text{pick-up} <\text{P:object}> | \text{move} <\text{direction}> | \text{use} <\text{P:object}>
\langle\text{direction}\rangle \rightarrow \text{north} | \text{south} | \text{east} | \text{west}
\langle\text{P:object}\rangle \rightarrow \text{pick} | \text{forge} | \text{smelt} | \text{iron} | \text{iron-ore} | \text{weapons}
Case Studies of Individual Characters

Six types of agent models are:

- **ADAPT_INTEREST**: A MFRL agent motivated to achieve interesting events.
- **ADAPT_COMPETENCE**: A MFRL agent motivated by interest and competence.
- **RECALL_INTEREST**: A MMORL agent motivated to achieve interesting events.
- **RECALL_COMPETENCE**: A MMORL agent motivated by interest and competence.
- **REUSE_INTEREST**: A MHRL agent motivated to achieve interesting events.
- **REUSE_COMPETENCE**: A MHRL agent motivated by interest and competence.
(a) Emergent behavioral policy for travelling.
Behavioral cycles by an ADAPT_INTEREST Agent

(b) $t = 2,965$

$S_4((\text{location:3})(\text{inventoryPick:1})(\text{inventoryAxe:1})(\text{inventoryIron:1}))$

$S_5((\text{location:3})(\text{inventoryAxe:1})(\text{inventoryIron:1})(\text{inventoryLog:1}))$

$S_7((\text{location:1})(\text{inventoryPick:1})(\text{inventoryAxe:1})(\text{inventoryIron:1})(\text{inventoryLog:1}))$

$S_4((\text{location:2})(\text{inventoryAxe:1})(\text{inventoryIron:1})(\text{inventoryLog:1})(\text{visibleSmelt:1})(\text{visibleForge:1})(\text{visibleWeapons:1}))$

$S_4((\text{location:4})(\text{inventoryPick:1})(\text{inventoryAxe:1})(\text{inventoryIron:1})(\text{inventoryLog:1})(\text{visibleLathe:1})(\text{visibleFurniture:1}))$

$S_4((\text{location:4})(\text{inventoryPick:1})(\text{inventoryAxe:1})(\text{inventoryIron:1})(\text{visibleLathe:1})(\text{visibleFurniture:1}))$

(b) Emergent behavioral policy for timber cutting and furniture making.
Behavioral cycles by an ADAPT_INTEREST Agent

(c) Emergent behavioral policy for iron mining and weapons-smithing.
Agents use the same MRL model can develop different focuses of attention, and thus different characters, based on their experiences.

Focus of attention by two ADAPT_INTEREST agents over 50000 time-steps. Agents that focus attention differently represent different game characters.
General Trends in Character Behavior

Average behavioral variety achieved by the six different agent models in the first 5000 time-steps.

Average maximum behavioral complexity achieved by the six different agent models in the first 5000 time-steps.
General Trends in Character Behavior

In MMORL and MHRL, option learning is initiated by motivation, but directed at an option level by the termination function which is a binary function. In contrast, the motivation functions directing learning in MFRL setting have continuous valued outputs and reward all actions related to smelting iron highly, including using the pick to mine iron-ore, and moving between the mine and the smithy.

<table>
<thead>
<tr>
<th>Time</th>
<th>Action</th>
<th>Motivation to achieve interesting events</th>
<th>Termination Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>6639</td>
<td>A(use, smelt)</td>
<td>0.961563</td>
<td>1</td>
</tr>
<tr>
<td>6640</td>
<td>A(move, north)</td>
<td>0.935992</td>
<td>0</td>
</tr>
<tr>
<td>6641</td>
<td>A(use, pick)</td>
<td>0.953195</td>
<td>0</td>
</tr>
<tr>
<td>6642</td>
<td>A(move, south)</td>
<td>0.896915</td>
<td>0</td>
</tr>
<tr>
<td>6643</td>
<td>A(use, forge)</td>
<td>0.983858</td>
<td>0</td>
</tr>
<tr>
<td>6644</td>
<td>A(use, smelt)</td>
<td>0.953195</td>
<td>1</td>
</tr>
<tr>
<td>6645</td>
<td>A(move, north)</td>
<td>0.939472</td>
<td>0</td>
</tr>
</tbody>
</table>
General Trends in Character Behavior

Cumulative behavioral variety by three of the agents motivated to achieve interesting events.
Designing Characters that Can Multitask

Add four additional MDPs, $P_3$ (farming), $P_4$ (fishing), $P_5$ (pottery), $P_6$ (wine-making).

Average behavioral variety achieved by the six different MRL agents.

Average maximum behavioral complexity achieved by the six different MRL agents.
Designing Characters for Complex Tasks

Increase the number of raw materials required to make a finished item from one to five.

Average behavioral variety achieved by the six different MRL agents

Average maximum behavioral complexity achieved by the six different MRL agents.
Games That Change While Characters Are Learning

Monster is spawned after 5000 time-steps and damage the forge and the lathe so that the actions for using the forge or lathe no longer produce weapons or furniture.
Change in attention focus over time exhibited by a single agent motivated by interest and competence in a dynamic environment.
General Trends in Character Behavior

Cumulative behavioral variety by six types of MRL.
Questions?

Reference: