

## Wirtualne zespoły robocze - instrukcja do zadania uczenie agentów w środowisku JADE

Celem zadania jest stworzenie systemu uczenia agentów w środowisku JADE w interakcji z innymi agentami, czego efektem może być czysta rywalizacja pomiędzy agentami, strategia zespołowa – drużynowa lub różne pośrednie warianty współpracy w zależności od parametrów środowiska.

### Zadanie podstawowe:

Uzupełnić algorytm szukania optymalnej strategii metodą uczenia ze wzmocnieniem (*Reinforcement learning*) lub z użyciem algorytmu genetycznego. Zadanie należy wykonać poprzez uzupełnienie kodu w odpowiednich klasach *UczenieZeWzmocnieniem* lub *AlgorytmGenetyczny*.

Uwagi: Po skompilowaniu projektu należy uruchomić jednego agenta typu *ControllerAgent*, np. za pomocą linii poleceń:

```
-agents cont1:ControllerAgent -gui
```

Uruchamianie lub umieszczanie agentów na zdalnych komputerach może się odbywać poprzez użycie zdalnych kontenerów (w wersjach JADE < 4.0 możliwa jest migracja agentów pomiędzy platformami) W tym celu na każdy zdalny komputer należy skojarzyć ze zdalnym kontenerem np.:

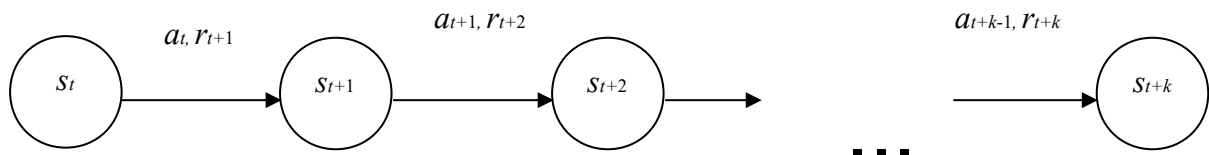
```
java jade.Boot -container -container-name C -host <adres IP hosta>
```

Tworzenie projektu w środowisku NetBeans:

- utworzenie projektu typu Java Application bez pliku głównego (należy odhaczyć opcję Create New Class)
- dodanie plików \*.java do projektu
- w zakładce Libraries należy umieścić plik jade.jar
- w zakładce Run należy podać nazwę klasy głównej: jade.Boot, zaś argumenty uruchomienia to `-agents cont1:ControllerAgent -gui`

### Uczenie ze wzmocnieniem

Uczenie ze wzmocnieniem (*Reinforcement learning*) jest metodą pozwalającą na szukanie optymalnej strategii zachowania agenta w wieloetapowych procesach decyzyjnych. Przykładowy proces przedstawiony na rysunku polega na przechodzeniu od stanu początkowego do innego stanu, następnie do kolejnego itd., przy czym każde przejście wymaga wykonania akcji  $a$  i wiąże się z otrzymaniem nagrody  $r$ . W ogólności proces nie jest deterministyczny: każdej akcji przypisane są prawdopodobieństwa przejść do kolejnych możliwych stanów oraz średnia wartość nagród.



Prawdopodobieństwa przejść oraz średnie wartości nagród nazywane są modelem środowiska, gdyż to środowisko pełni rolę arbitra oceniającego każdą akcję oraz wyznacza stan kolejny. Znając model środowiska możliwe jest zastosowanie programowania dynamicznego do obliczenia wartości akcji w poszczególnych stanach dla danej strategii, a następnie modyfikacja strategii na podstawie tablicy wartości. Powtarzając oba kroki możemy otrzymać strategię optymalną w momencie ustabilizowania się funkcji strategii. Jeśli model środowiska nie jest znany (tak jest np. gdy w środowisku działają inne agenty o nieznanym lub zmieniającym się strategiami), czasami można go wyznaczyć metodą

symulacji Monte Carlo: wielokrotnie symulować zachowanie się agenta w interakcji ze środowiskiem przy danej strategii, obliczać średnie nagrody oraz modyfikować strategię; jednak jest to metoda o dużej złożoności obliczeniowej. Inną metodą jest tzw. *Q-learning* będąca implementacją metody różnic czasowych (*Temporal Difference Learning*):

```
Zainicjuj  $Q(s, a)$ 
Repeat (dla kolejnych epizodów):
  Zainicjuj  $s$ 
  Repeat (dla kolejnych kroków epizodu):
    1.) Z prawdop.  $1-\epsilon$  wykonaj akcję  $a$  w stanie  $s$  o
        najwyższej wartości  $Q$  lub akcję losową
        z prawdop.  $\epsilon$  przechodząc do stanu  $s'$ 
    2.) Zmodyfikuj wartość akcji  $a$  w stanie  $s$ :
        
$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

  until  $s$  jest stanem końcowym
until spełniony warunek końca
```

Dla każdej akcji  $a$  w stanie  $s$  obliczana jest wartość  $Q(s, a)$ , która przybliża średnią sumę nagród (kary to nagrody ujemne) uzyskiwaną po wykonaniu akcji  $a$  w stanie  $s$  podczas całego epizodu (np. przejścia labiryntu). Nagrodę  $r$  można uzyskać bezpośrednio po wykonaniu akcji, jak w ogólnym wzorze na modyfikację wartości  $Q$  lub po wielu krokach. W tym drugim przypadku jej wartość odzwierciedlona jest w wyrażeniu  $\max Q(s', a')$  do którego przybliżana jest wartość  $Q(s, a)$  w danym kroku.

W fazie eksploatacji systemu, w celu maksymalizacji zysków w każdym kroku wybierana jest akcja, dla której wartość  $Q$  w bieżącym stanie jest maksymalna. W fazie uczenia, oprócz modyfikacji wartości  $Q$ , należy z pewnym prawdopodobieństwem wybierać akcję losową, tak aby obliczyć wartości wielu akcji w wielu możliwych stanach.

Więcej informacji można znaleźć na stronie:

<http://www.incompleteideas.net/sutton/book/ebook/node65.html>

### Algorytm genetyczny

Uczenie polega na wyznaczeniu populacji rozwiązań, a następnie ich selekcji, krzyżowaniu oraz mutacji. Rozwiązaniem jest w tym przypadku strategia, a więc informacja o akcji wykonywanej w każdym ze stanów środowiska.

### Zadania dodatkowe

Wykonać zadanie w maksymalnie 3 wariantach (liczba wariantów jest ustalana przez prowadzącego):

1. Strategii reprezentowanej informacją o akcji w każdym z pól planszy dla agenta w interakcji ze środowiskiem. Agent powinien nauczyć się chodzenia po labiryncie, unikania kar (czerwone kółka) i zdobywania nagród (żółte kółka).
2. Uwzględnić interakcje z innymi agentami w trakcie uczenia (każdy z agentów przesyła pozostałym swoją aktualną strategię). Agent powinien nauczyć się unikania tłoku (kary za kolizje) oraz w pewnym stopniu współpracy.
3. Rozszerzyć reprezentacje stanu i strategii o informacje o położeniu innych agentów. W najprostszym przypadku liczba stanów = liczba pól  $\wedge$  liczba agentów. Ze względu na wykładniczo rosnącą liczbę stanów wraz z liczbą agentów można dokonać uproszczeń, np.

dodać tylko informację o tym czy inny agent znajduje się daleko czy blisko i w której ćwiartce planszy w stosunku do naszego agenta. Agent powinien nauczyć się współpracy lub rywalizacji z innymi agentami.

### Informacje praktyczne

Z uwagi na wymaganą dużą liczbę epizodów, proces uczenia warto jest umieścić w oddzielnym wątku, bez interakcji z użytkownikiem czy z agentem zarządzającym. Równoległe powinna być wyświetlana plansza pokazująca zachowanie agenta korzystającego z aktualnie wyuczonej strategii (czysta eksploatacja).