エージェントの知覚と学習による行動決定の実現

奈良女子大学 理学部 情報科学科 4 回
新出研究室 亀村美佳

概 要

本論文では、連続的な仮想世界において BDI モデルを用いたエージェントの学習及び行動決定の実装について述べる。

近年、実世界において自律的に振る舞うロボットの実現に向けて、研究が進められている。連続な実世界では仮想世界とは異なる様々な問題に直面するため、それらに対処できる能力が求められる。これに際し様々な実験を必要とするが、実世界で直ちに実行に移すことは難しい。従って、連続的な仮想世界におけるシミュレーション環境の開発は、自律型ロボットの研究において重要課題である。

先行研究は、連続した状態空間上での行動シミュレーションを想定したものではあるものの、エージェントが学習を行って自律的に自己の行為を変更することはできなかった。そこで我々は、エージェントの学習及びそれに伴った行動選択の実現を目指した。本論文では、エージェントの知覚と行動選択について述べる。

1 はじめに

BDI モデルとは、信念（B）、願望（D）、意図（I）と呼ばれる 3 つの心的状態を用いて意思決定を行うモデルである。この BDI モデルを用いた合理的かつ自律的なエージェントを BDI エージェントと呼ぶ。近年、実世界において自律的に振る舞うロボットの実現に向けて研究が進められている。連続な実世界では、仮想世界とは異なる様々な問題に直面するため、それらに対処できる能力が求められる。これに際し様々な実験が必要となるが、実世界で直ちに実行に移すことは難しい。従って、連続的な仮想世界におけるシミュレーション環境の開発は、自律型ロボットの研究において重要課題である。

仮想世界における BDI モデルの有用性が [1] で示されていることから、我々は BDI モデルを用いたエージェントを持つシミュレータを提案してきた。[2, 3]

実世界は常に動的に変化し、仮想世界とは異なる様々な問題に直面する。このため、離散的な仮想世界では、実世界を想定するものとして不十分であり、シミュレーション環境の作成においても同様のことが言える。先行研究 [2, 3] では、Jason と呼ばれる言語処理系と Java を用いて、身体性を持つエージェントが基本行為を行うシミュレーションを実現した。それまでの研究において仮想世界が離散的に記述されていた点については解決したものの、エージェントが連続的な世界で情報を取り出し、学習及び自律的な行動選択を行うことはできなかった。

そこで我々は動的に変化する環境を想定したシミュレーション開発に向けて [2, 3] を拡張し、エージェントの学習及びそれに伴った行動選択の実現を目指した。

本研究は奈良女子大学理学部 4 回生林との共同研究である。本論文では以下の点について述べる。

- エージェントによる知覚の実現
- 学習結果を用いた行動選択の実現

それ以外、特に学習過程の実現と、実装におけるクラス設計などについては [5] で述べられている。

2　シミュレーション概要

2.1　環境設定

環境を次のように設定する (図 1)。

- 川の形状、及び流れの向きと速さを予め定めておく。
- エージェントが下る川には障害物がないものとする。
- 川を 8 × 6 マスに区切り、一つのマスの中の川の流れの向きと速さは同じであるものとする。

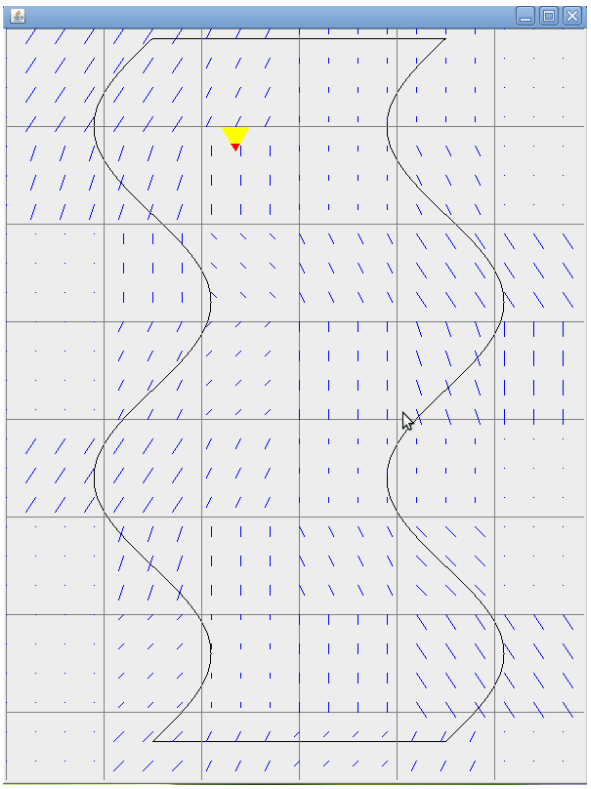

図 1: 川の形状

2.2 問題設定

本研究では、一つの BDI エージェントを用い、カヌーレーシングをテストベッドとしたシミュレータを新たに拡張した。ここでのエージェントとは、具体的にはカヌーを指す。エージェントは、基本行為を繰り返すことで目標の達成を目指す。今回の問題設定では、エージェントが持つ目標は「川を下り終える」ことである。川の上端をスタート地点とし、下端に到達したらゴールとみなす。学習手法としては強化学習を用い、以下の性質を満たすような行為の獲得を目指す。

- より早く川を下り終える
- 川の下端に到達するまでに川から外れない

そこで、強化学習においては川をスタートしてからゴール到達あるいは川の外に出るまでを一つのエピソードとし、上記の 2 性質を満たす性質を学習できるよう報酬を設定して学習する。

2.3 基本行為

基本行為とは、それ以上の分解を考えない最低限の一段階の行為である。本研究では、この基本行為を 5 つに定めた。すなわち、「何もしない」、「前進」、「後退」、「右へ曲がる」、「左へ曲がる」である。基本行為で「何もしない」という行為は、本研究においては川の流れに身をまかせる状態を指す。また、「右へ曲がる」、「左へ曲がる」は、それぞれ「左側のみをこぐ」「右側のみをこぐ」ことと同意である。

3 拡張内容

3.1 従来までの問題点

先行研究 [2, 3] では、連続的な仮想世界におけるシミュレータではあったものの、エージェントが学習することはできなかった。これによりエージェントは自己の行動結果を分析及び評価することができず、行動を選択することはできなかった。そこで本研究では、まずエージェントによる知覚を可能にすることで、知覚情報をもとにエージェントが学習する機能を追加し、学習結果を利用してエージェントが行動選択するよう拡張した。

3.2 追加機能

知覚

エージェントが強化学習を行うためには、まずエージェントによる知覚が必要である。本研究では、環境情報として位置と川の流れに着目し、エージェントの現在地及び現在地における川の流れを知覚することを可能にした。

現在地を知覚できることにより、川岸に迫った際に川から外に出ることを事前に認識し、これを回避することが可能となる。今後、障害物を設定、あるいは複数のエージェントとの競争を想定する等、難易度をより高くしていく際にもこの知覚は有効である。実世界ではロボットないしレーサーは、視覚によって様々な情報を得ることになる。現在地の知覚は、これを想定して実現したものである。

川の流れもまた、本研究においては重要な情報である。カヌーレーシングでは、より早く川を下り終えることを一つの目標としている。このため、流れの早い地点を判別しその経路を選別することは、より効率的な手段となる。従って本研究ではこの 2 点の知覚を実装した。

具体的には、既存の Java クラスに知覚のメソッドを追加することで、知覚した情報をエージェントに伝達できるようにした。知覚機能の追加に伴い、エージェントのプランも変更した。これについては 4 章で述べる。

強化学習

次に、エージェントが知覚した情報をもとに強化学習を行う機能を追加した。これにより、エージェントの行動結果は数値化され評価することができる。実際には、ある状態における状態行動価値を推定し、これを用いて自己の行動結果を分析することを可能にした。学習方式としては Q 学習を用いた。

強化学習実装のため、Java クラスを新たに作成した。作成したクラスの設計や実装については [5] で述べる。

行動選択

強化学習で得られた結果をもとにエージェントが行動を選択する機能を追加した。具体的には、強化学習によって求めた状態行動価値の推定値が最も高い行動を導き出す。

行動選択のための Java クラスを新たに作成すると共に、Jason のプランに変更を加え、これを可能にした。行動選択に関するプランの変更は 4 章で述べる。

4　シミュレーション開発環境

開発環境は、先行研究 [2, 3] 同様、Jason と Java を用いた。BDI モデルを基礎としたインタプリタである Jason にプランを記述しアクションを定義、Java で環境を実装した。Jason のエージェントのプランは、学習結果を用いて行動できるよう変更を加えた。この変更については 4.1 で述べる。

4.1　エージェントのプラン変更

本研究では、エージェントのプランに変更を加えた。これを図 2 及び図 3 で示す。エージェントの名前は本質的ではないので、ここでは適当な名前（aaa）とした。

cpos()、action()、csp() は、それぞれ現在地を得る述語、エージェントの行動を得る述語、現在地のスピードを得る述語である（図 3）。これらは、RiverEnv に新たに作った UpdatePercepts メソッドによってエージェントに情報が伝達されることで、取得が可能となった。

3 つの述語のうち cpos() と action() はプランの前提条件として呼び出している。cpos() で得た現在地は、RiverUpdate で強化学習を行う際に利用している。また action() は、RiverDecide が導き出した状態行動価値の推定値が最も高い行動を、RiverAgent を介して取得し、エージェントの次の行動に作用している。

```
// Agent aaa in project canoe.mas2j

!initial_goal.
+!initial_goal<-
!goal.

+!goal
: true
<-
//do_nothing;
//move_toward;
//move_back;
right;
left;
   !goal.


-!goal<- .print("      ").
```

図 2: エージェントのプラン 変更前

```
// Agent aaa in project canoe.mas2j

   !initial_goal.
   +!initial_goal<-
   !goal.

   +!goal
   : cpos(loc(X,Y)) & action(Z) & csp(S)
      //現在地を知覚する cpos() と行動を得る action() を追加。
      //csp() は現在地のスピードを知覚する述語であるが、
      //現時点では学習には使用していない。
<-
.print(csp(S));
.print(Z);
if(Z<0){
 .print(Z);
}else{
  if(Z=2){ //何もしない
    .print("do_nothing");
    move_back;
  }else{
    if(Z=1){ //前進
      move_toward;
    }else{
      if(Z=0){  //後退
         move_back;
      }else{
         if(Z=3){ //右にこぐから左に行く
              left;
         }else{ //Z>=4 左
           right;
        }}}}}
         //action() で得た結果によって場合分けし、
         //エージェントが行う基本行為を選択する。


 !goal.

-!goal<- .print("pppp").  //目標達成に失敗した場合に表示される
                          //デバッグ用メッセージ
```

図 3: エージェントのプラン 変更後

4.2 環境のクラス概要

先行研究 [2, 3] でのシミュレータ環境に二つの Java クラスを追加し、以下の 8 つのクラスで環境を定義した。以下のうち RiverUpdate と RiverDecide が追加されたクラスである。

RiverEnv

[3] に、知覚するメソッド updatePercepts を追加した。メソッド内では、エージェントが一回基本行為を行うごとに前の知覚情報を削除し、新たな情報を知覚する。知覚情報は、エージェントの現在地及び、現在地における川のスピードである。

また、現在地における最も望ましい基本行為をエージェントに伝達する作業もこのメソッドで行った。

[2, 3] では、川岸にぶつかると自動的に川の中へ戻るようプログラムされていた。しかし本研究では、強化学習によって最善の行為を導き出すことで、エージェントが川の外に出なくなるよう改善されていくシミュレータを目指していた。従って、川岸に達した場合はそのまま川の外へ出るプログラムである必要があった。この点に関する変更点は [5] で述べられている。

RiverModel

詳細は [3, 5] で述べられている。

RiverView

詳細は [3] で述べられている。

RiverShape

詳細は [3, 5] で述べられている。

Location

座標を返すメソッドを持つ。特定の位置の座標を求める際に他のクラスから呼び出される。

RiverAgent

[2, 3] に新たにメソッドを追加した。RiverUpdate による強化学習の結果を持ち、RiverDecide によって導き出された、行動推定価値が最も高い行動を取得する。そしてこれは RiverEnv によって呼び出され、エージェントへ伝えられる。

RiverDecide は RiverUpdate から強化学習の結果を得るが、実際には強化学習の結果は RiverAgent に伝達及び、保持されており、これを RiverDecide が呼び出す仕組みになっている。RiverAgent は RiverUpdate と RiverDecide を仲介する役割を担っている。

RiverUpdate

知覚によって得られた情報をもとに強化学習を行うクラスである。エージェントの行動結果を数値化する。詳細については [5] で述べられている。

RiverDecide

RiverRpdate.java による強化学習によって得られた、エージェントの行動結果を数値化したものをもとに、次の行動を選択するクラスである。ある状態でのある行動の状態行動価値の推定値を RiverAgent から取得し、推定値の最も高い行動を RiverAgent に返す。

5 結果

先行研究の実行結果を図 4 で示す。図 4 では、エージェントが川岸に到達しても川の外に出ることはない。これは、エージェントが川の外に出ることがないよう予め環境でプログラムされているからである。これでは正しく強化学習を行うことができない。これの変更については [5] で述べられている。

また、エージェントが知覚できないため、状況によって漕ぎ方を選択できない。図 4 では、図 2 のプランに従い、「右へ曲がる」と「左へ曲がる」を繰り返している。これらを改善し、知覚による行為決定と、学習で得た行為の利用ができるようにした結果が図 5 及び図 7 である。

図 5 は学習前の実行結果であり、エージェントが知覚しながら進行している。図 7 の [aaa] はエージェントの応答を示すものである。cpos() は現在地を XY 座標で取得していることを表し、csp() は川の流れの左右の向きと速さの取得を表している。川の流れの左右の向きと速さは、speed(正の数, 速さ) ならば右向き、speed(負の数, 速さ) であれば左向きである。

学習を行っていない図 5 では、途中で川岸に衝突しゴールできていないことが分かる。図 5 から学習を繰り返すと、図 6 の実行結果が得られる。これは、20000 回目のエピソードの結果である。

学習前後を比較すると、エージェントが川岸への衝突を回避し、更に、効率的な行為を選択しより早くゴールへ到達できるように改善されている。

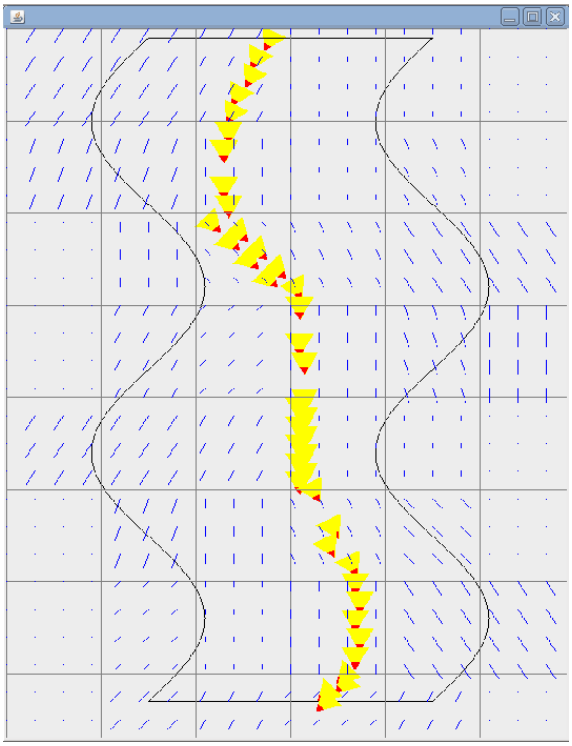

図 4: 先行研究の実行結果

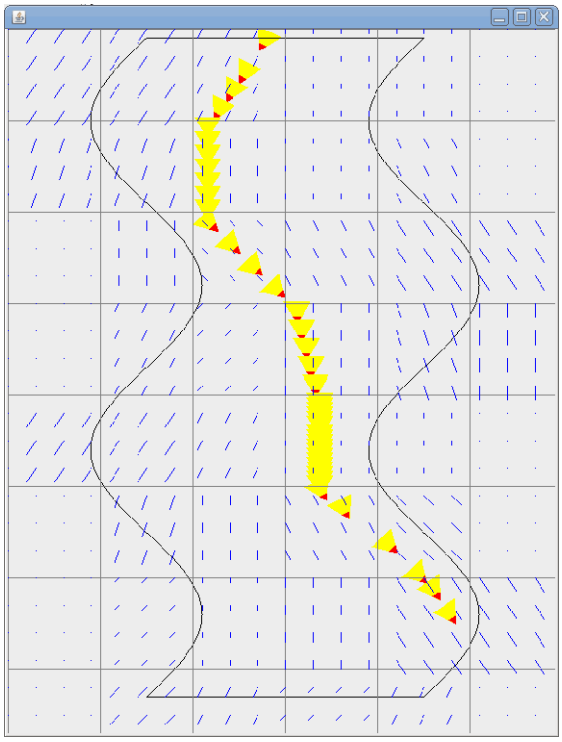

図 5: 学習前の実行結果

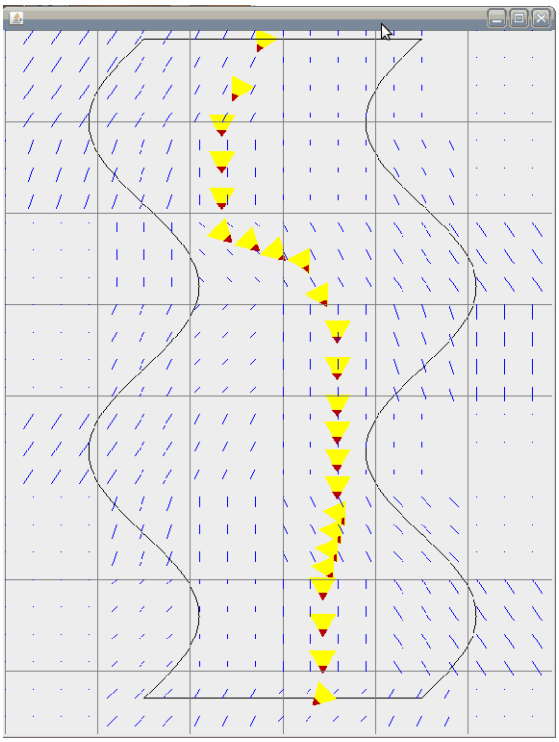

図 6: 学習後の実行結果

```
[aaa] cpos(loc(216.4589803375032,195.52786404500043)) ←現在地の知覚
[aaa] csp(speed(0,2))                                  ←現在地のスピードの知覚
[aaa] 0                                                ←行為を取得
result=true
[aaa] cpos(loc(216.4589803375032,210.52786404500043))
[aaa] csp(speed(1,1))
[aaa] 1
result=true
[aaa] cpos(loc(240.60111596123414,234.66999966873138))
[aaa] csp(speed(1,1))
[aaa] 1
result=true
[aaa] cpos(loc(264.7432515849651,258.8121352924623))
[aaa] csp(speed(1,1))
[aaa] 1
result=true
[aaa] cpos(loc(288.88538720869605,282.95427091619325))
[aaa] csp(speed(1,1))
[aaa] 1
result=true
[aaa] cpos(loc(313.027522832427,307.0964065399242))
[aaa] csp(speed(0,2))
[aaa] 4
result=true
[aaa] cpos(loc(318.027522832427,327.0964065399242))
[aaa] csp(speed(0,2))
[aaa] 4
result=true
[aaa] cpos(loc(323.027522832427,347.0964065399242))
[aaa] csp(speed(0,2))
[aaa] 4
result=true
```

図 7: コンソールの一部抜粋

6　まとめ

本研究では、カヌーレーシングをテストベッドとした連続世界に基づいたシミュレーション環境において、エージェントの知覚と学習、及び行動選択の実装を目指した。その結果、エージェントは環境情報を取得し、学習を行いながら効率的な行為を選択することが可能となった。

しかし、エージェントが行為を選択することは可能になったものの、状況に応じたプラン作成までは至っていない。エージェントの自律的に振る舞いを実現するには、学習結果で獲得した行動をプランに活用させる必要がある。これに向けて、シミュレーションの設計と実装の観点から今後の展望を挙げる。

- 知覚した複数の情報を強化学習に利用する

  本研究では、複数の情報を知覚できるものの、実際に学習に利用したのは現在地のみであった。実世界では、川の流れに応じて漕ぐ速さを変える等の必要が出てくるため、複数の情報を学習に利用できるべきである。

- 障害物を設定するなど、難易度を高めても問題解決できるようにする

  本研究におけるシミュレーションでは、障害物を設定せず、川下りとしては難易度の低いものであった。実際の川には岩等様々な障害物をあることが想定されるため、問題設定の難易度を高くし、より実世界に近い設定に対応できる必要がある。

- 基本行為の種類を増やす

  エージェントの基本行為は 5 種類に定めていた。本研究では漕ぐ方向のみに着目していたが、実世界では 4 方向以上に細かな動きを要し、更に、漕ぐ速さも必要に応じて変えなければならない。基本行為の種類を増やすことで、エージェントはより細かな動きをすることが可能となり、シミュレーションの精度を上げることができる。

- エージェントを増やし、相互的な動作を考慮する

  実世界では様々なものが相互的に作用している。カヌーレーシングをテストベッドとしたシミュレーションでは、別のエージェントと競うことが前提にあり、いかにして他のエージェントより速く下るか、あるいは追い越すか等を考慮した動作を実現することが求められる。

- 川の形状が変化しても対応できるようにする

今後は、これらの問題について取り組むことが課題である。

7　謝辞

本研究を遂行するにあたり、いつも親身にご指導くださった新出尚之准教授に深く感謝し、厚く御礼申し上げます。また新出研究室の皆様に感謝の意を表します。ありがとうございました。

参考文献

[1] 高田司郎, 新出尚之. 行為のアトラクター状態を考慮した知能ロボットについて. In Proc.of JAWS2012, 2012. 2012

[2] 濱田百合. 連続世界におけるエージェントの学習と意思決定に向けて. 奈良女子大学理学部情報科学科 2012 年度卒業論文, 2013.

[3] 久妻さゆり. 実世界におけるエージェント構築に向けたシミュレーション環境の作成. 奈良女子大学理学部情報科学科 2012 年度卒業論文, 2013.

[4] Richard S.Sutton, Andrew G.Barto（著）, 三上貞芳, 皆川雅章（訳）. 「強化学習」. 森北出版株式会社, 1998.

[5] 林果穂. 連続的なシミュレーション環境でのエージェントの学習と意思決定の実装について. 奈良女子大学理学部情報科学科 2013 年度卒業論文, 2014.