目次

第 1 章 はじめに

1.1 背景

現在多くの分野でロボットやコンピュータが活躍しており，より身近なものとなってきた[1]．身の回りには工業用ロボットから家庭用コンピュータまで様々な場面で様々な機械が見られるようになった．少し前までロボットやコンピュータは多くの場合，ある入力に対して決められた出力が用意されているような単純なものであり，特定の用途でのみ用いられることが多かった．しかし，最近ではロボットやコンピュータはより多様な用途が求められている（Fig.1.1）．それは社会が複雑な環境であるがゆえにロボットやコンピュータは様々な場面に対応する必要が出てきたためである．例えば家庭内で動く掃除ロボットを考える．家庭内は人間が思っている以上に複雑な環境である．テーブルや家電製品などのある程度固定されたものや，床に散らばった雑誌やテーブルの上の食器など置かれる場所がばらばらなものまである．こういった環境の中で毎日同じ動作をして掃除をすることは不可能である．掃除を行うときの状況に合わせた行動が必要になってくる．こういった多様さに対応するための方法の一つとしてロボットやコンピュータが自ら学習を行う機械学習[2]がある．

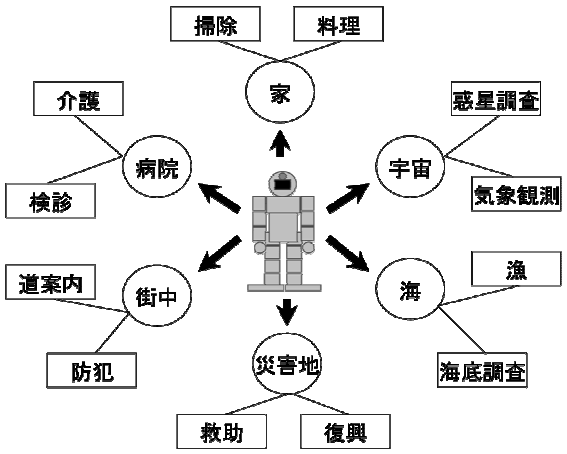


Fig.1.1：ロボットを取り巻く環境

1.2 機械学習

機械学習とは人間が環境に適応していくように，ロボットにも状況に合わせた行動を取れるようにロボットに知能を持たせる方法である．機械学習はこれからの社会において活

躍するであろうと期待される分野である．特に人間が活動できないような苛酷な環境や未知の環境に対して活躍するだろうと考えられる．

この機械学習に関してはすでに大まかな枠組みが出来上がっており，有名どころではニューラルネットワーク・遺伝的アルゴリズム[3]・強化学習[4]などが挙げられる[5]．中でも強化学習は実ロボットに用いられることが多い手法として注目されており，様々な研究がされている．実ロボットに対しての研究[6]だけでなく，ネットワークのルーティングに関する研究にも応用されている[7]．

1.3 強化学習

強化学習はある状態で取った行動の結果に注目し，このときの評価が良くなるように学習を行うものである．このときに利用するのが報酬と呼ばれるスカラ値の情報である．ロボットは行動を取ることでその行動に見合った報酬が得られる．この学習方法は人間にも当てはまることのある学習である．例えば子供は悪いことをすれば大人に怒られる．逆に良いことをすれば褒められる．当然子供は褒められたいし，怒られたくない．いろいろなことをしていくうちに子供は悪いことをしなくなっていき，良いことをするようになる．強化学習はこの過程と全く同じである．怒られるということをマイナスの報酬とし，ほめられるということをプラスの報酬として，ロボットに学習させるのである．

報酬は人間によって設定され，この報酬を元に試行錯誤から目的を達成するように学習が行われる．この報酬による学習の有用な点は変動のある環境を扱うことができる点となっている．

1.4 強化学習の問題点

ロボットが身近になり，複雑な環境でも学習できるようになってきたことで人間はロボットやコンピュータに多様な仕事を求めるようになった．ある特定の環境で１つの仕事をこなすだけではなく，変化のある環境の中で様々な仕事をこなすものを求めているのである．

しかし，強化学習を行う場合には何度も学習を行う必要があるので完全に学習するまでに時間がかかってしまうという欠点がある．これはコンピュータ上（シミュレーションなど）の話ならば問題はないが，実際のロボットなどの実機では学習に時間がかかるというのは好ましくない．また，強化学習はある目的に対して設定された報酬によって学習が行われており，学習の過程が報酬に依存している．そのため同じ環境下であったとしても学習途中や学習終了後に目的が変わってしまうと，それまでの目的に対しての学習の結果によって，効率的に学習ができないことが課題となっている．つまり，目的が頻繁に変わるような場合には強化学習ではうまく学習できないことがある．

1.5　目的

本論文では強化学習において，目的が変わった場合に素早く対応できるシステムの提案を行う．

1.6　アプローチ概要

本論文では強化学習が報酬による学習であることに注目し，報酬とは別の情報を利用して学習するアプローチを取る．報酬とは別の情報を知識化する．これによって報酬による学習だけでなく．報酬とは別のものに対して学習を行っていく．また，知識化した情報を強化学習に影響を与える形で利用することで本論文の目的を果たす．

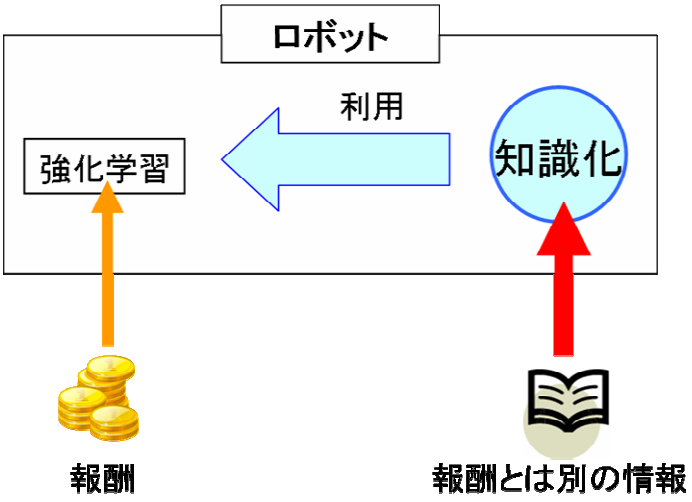


Fig.1.2：アプローチ

第 2 章 強化学習

2.1 強化学習概要

2.1.1 強化学習とは

強化学習はロボット（エージェント）などにおいて経験的に物事を学習していく学習方法となっている．ロボットは自分を含む周囲の環境を認識し，より良い行動をとるように学習を進めていく．

強化学習で扱う環境は基本的なものだとマルコフ決定過程[8]としてモデル化された環境である．この環境モデルに対しては有限回の試行で最適な解を見つけ出すことが可能であることが証明されている．ただし，マルコフ決定過程としてモデル化されていない環境でも強化学習を用いることは可能であるが，この場合最適な解が見つかることは保障されていない．

強化学習では報酬と呼ばれるスカラの値を用いて学習を行う（Fig.2.1）．ロボットは行動を選択するとその行動の結果に見合った報酬を受け取る．受け取った報酬を基にその行動が良かったのかどうかを判断する．この報酬は人間によって設定されるため，実際にロボット等に学習を行わせる際には目的に合わせて人間が報酬を設定してやる必要がある．近年ではこの報酬をロボット自ら自動的に設定するような研究も行われている[].

強化学習は大きく 2 つのパートに分けることができる．一つは学習部と呼ばれ，もう一つは行動選択部というパートである．学習部と行動選択部については次の 2.2 節・2.3 節で詳しく話をする．

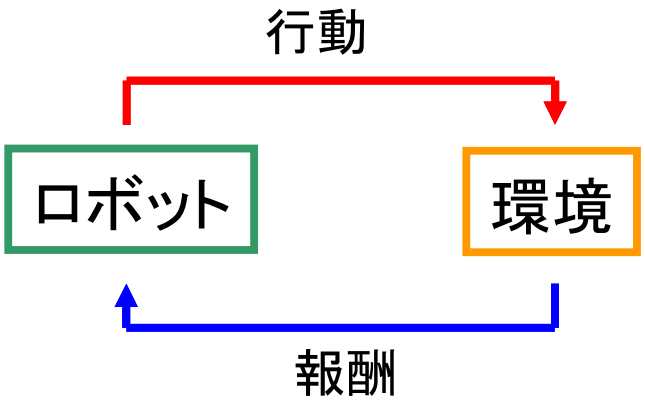


Fig.2.1： 強化学習における行動と報酬の関係

2.1.2 強化学習の構成要素

強化学習を構成する要素としては以下のものが挙げられる．これらから構成される例として Fig.2.2 を示す.

- ロボット（エージェント）

  実際に学習を行うもの．コンピュータ上で行うシミュレーションの場合にはエージェントと呼ばれる．ロボットやエージェントはセンサを有し，周りの環境を認識することが可能である．また，ロボットやエージェントは何らかの行動を取ることができる．ただし取れる行動はロボットやエージェントの身体構造に依存する.

- 環境

  ロボットを取り巻く周りの状態．例としてロボットが家の中に置かれているとすると環境は家の中となる．この家の中にテーブルや電気があり，これらは環境の要素である．ロボットが認識する状態はこの環境の要素の状態によって決められる．強化学習において環境を認識することは重要な要素である[9]．また，認識できる状態は所持しているセンサに依存する.

  例えばロボットがカメラをセンサとして持っていた場合,「目の前にテーブルがある」・「電気がついている」などを認識することができる．ロボットは「テーブルを押す」・「スイッチを押す」などの行動を取ることで「テーブルが移動する」・「電気が消える」といった環境の要素の状態変化が起きる．つまりロボットが行動することによって環境が変化する．この時ロボットが認識している状態は別の状態へと遷移するのである．また，ロボットが行動しなくても「時間が経つと電気が消える」といったような環境も存在する．このようにロボットの行動やそれ以外の要因で変化したりするような環境は動的環境と呼ばれる．これに対して変化が起きない環境は静的環境と呼ばれる.

- 報酬関数

  報酬関数は強化学習において目的（タスク）を表している．つまりこの関数はある状態で何が良くて何が悪いかを定義したものである．基本的にはロボットが認識したある状態に対してスカラの値が報酬として定義される．ロボットは行動を取ることで別の状態に遷移し，遷移した先の状態から報酬を貰うということになる．この報酬関数は目的に応じて人間が定義するのが一般的である.

- 学習部

  実際に学習を行う部分で，目的に対して学習が行われる．つまり受け取った報

酬を基に自分のとった行動の評価が行われる部分である．この学習部では価値関数と呼ばれるものを利用する．報酬関数が即時的な意味合いで何が良いかを表しているのに対して，価値関数は最終的に何が良いかを表す．報酬関数によって学習が行われた結果が価値関数であるとも言える．よってこの価値関数はロボットが変更を行うものである．詳しい内容は 2.2 節で話をする．

・ 行動選択部

ここでは学習部で行われた評価の結果から次に取る行動を選択する．詳しくは 2.3 節で説明する．

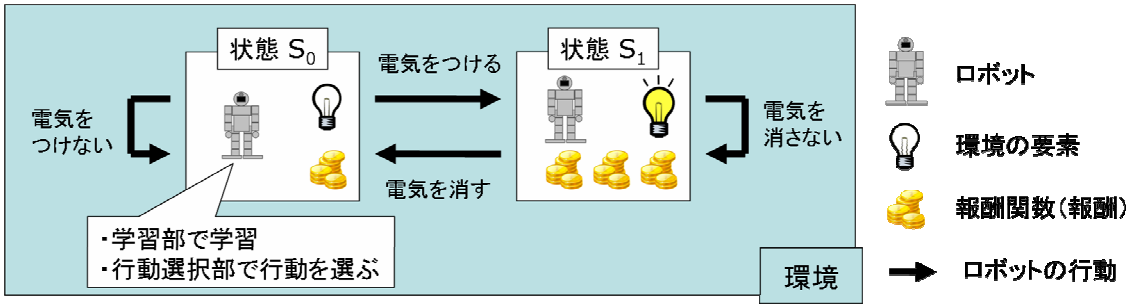


Fig.2.2： 強化学習の構成例

2.1.3 強化学習の流れ

強化学習の流れは Fig.2.3 のようになっている．

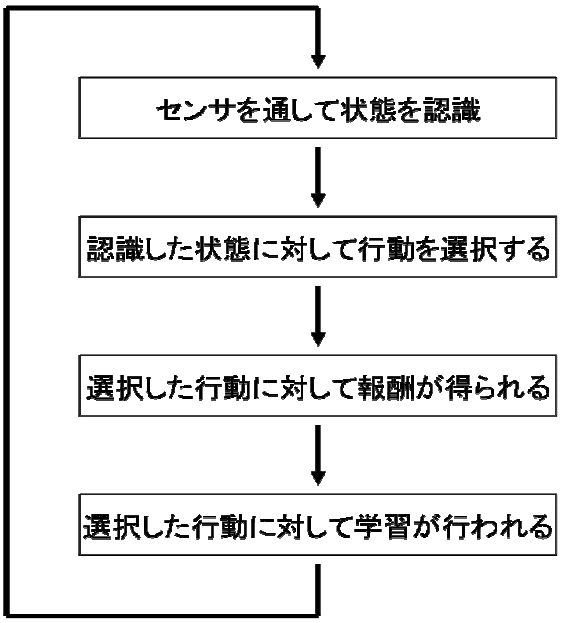


Fig.2.3： 強化学習の流れ

ロボットやエージェントはセンサを有しており，このセンサで自分が置かれている状況を認識する．そしてそれまでの学習の結果から認識した状態に対して行動を選択する．この時どの行動を選択するかは行動選択部の手法に依存する．この選択した行動に対して報酬を得ることで学習を行っていく．どのように学習するかは学習部の手法に依存している．ロボットやエージェントはこのサイクルを繰り返すことで目的に対してより最適な行動を学習していく．

2.1.4 強化学習の利点

強化学習には以下のような利点・特徴が存在する．

- 動的な環境・未知の環境を扱うことが可能

  報酬によって学習を行うことで動的な環境や未知の環境を取り扱うことが可能になっている．これは報酬構造と環境構造が別々のものだからである．このことから強化学習は動的な環境や未知の環境下でのロボットの行動獲得などの問題に用いられることが多い．

- 学習の際に教師を必要としない

  強化学習ではどのように学習していくかという学習過程まで人間が教えてやる必要は無い（教師を必要としない）．最終的な目的を報酬という形で与えさえすれば，目的を達成するまでの過程というのは自動的に学習する流れになっている．

- 試行錯誤による探索

  強化学習では試行錯誤による行動からどのように行動すれば良いかを学習する．試行錯誤によって様々な経験をすることでより最適な行動を見つけ出すことができる仕組みになっている．

- 遅延報酬

  報酬の与え方は様々な方法があるが，強化学習では遅延報酬と呼ばれる形で報酬を与える場合がある．遅延報酬では最終的な目的の状態に対してのみ報酬を設定するなど，すべての状態に対して報酬を設定するわけではない．このためロボットは自分の取った行動の良し悪しを即座に決めることができないことがあり，この時行動の評価に遅れが生じる．この遅延報酬による問題設定はより現実的な問題に対して用いられることが多い．これは実社会においてすべての状態を定義できないことが多いためである．このような場合は最終的な目的の状態に報酬を設定し，あとはロボットの試行錯誤による学習に任せるといった形になる．

2.1.5　マルコフ決定過程

強化学習における最も基本的な環境ではマルコフ決定過程（MDP）としてモデル化された環境である．この環境は以下のような特徴を持つ

- 環境は状態を持ち，その状態を完全に認識可能である
- ロボットが行動を行うと環境が確率的に遷移し，環境から報酬を確率的に得られる
- 状態 s'への遷移が，そのときの状態 s と行動 a にのみ依存し，それ以前の状態や行動には関係ない
- どの状態からスタートしたとしても，無限時間経過した後の状態分布確率（どの状態にいるかを表した確率分布）は最初の状態とは無関係である

環境の状態を完全には認識できない場合は部分観測マルコフ決定過程（POMDP）[]という．マルコフ決定過程の環境では，ロボットによる環境の状態認識は完全であることが仮定されている．しかし，現実ではセンサの能力やノイズによって認識した状態に不確実性・不完全性がある場合が多い（Fig2.4）．部分観測マルコフ決定過程(POMDP)は，マルコフ決定過程のモデルを拡張し，ロボットの状態認識に不確実性を付加したモデルである．

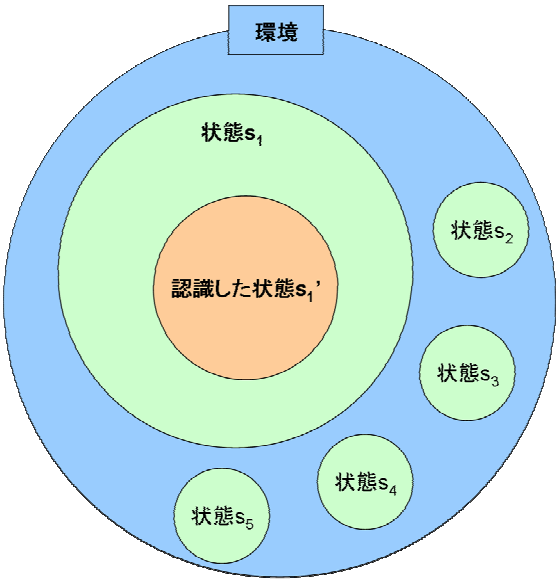


Fig.2.4：環境の状態と認識した状態の関係

2.2　強化学習・学習部

ここでは強化学習における学習部の話をする．学習部では手に入れた報酬に基づいて実際に学習を行う部分である．強化学習では報酬から自分の取った行動の良し悪しを判定して学習を行っている．一言で学習と言っても，どのように行動の良し悪しを判定するか，判定した結果をどのように保持するかなどの点から様々なアルゴリズムが存在している．中でも有名な手法に Q 学習と呼ばれるものがある．本論文では実験で Q 学習を用いるので，ここからはこの Q 学習について詳しく話をする．

2.2.1　Q 学習

Q 学習ではロボットやエージェントは学習の結果又は途中経過を示す Q 値を持つ．この Q 値は価値関数であり，将来的にどの程度報酬を貰えるかを表している．そのため Q 値は期待報酬または行動の評価値とも呼ばれている．Q 値はロボットやエージェントが認識する状態とその時に取ることができる行動の一つをペアにしたものに対して与えられることが多い（Fig.2.5）．例えばロボットの現在の状態を s_t とし、この状態で可能な行動が a_1・a_2・a_3・a_4 の 4 通りあるとする．この時ロボットは 4 つの Q 値 $Q(s,a_1)$・$Q(s,a_2)$・$Q(s,a_3)$・$Q(s,a_4)$を元に取る行動を決定する．行動を決定する方法は行動選択部の手法に依存する．行動選択部については 2.3 節で詳しく説明する．

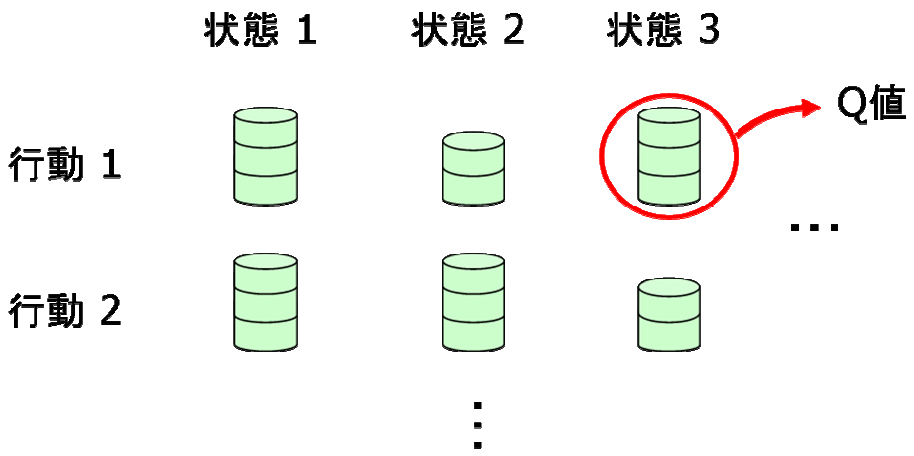


Fig.2.5：Q 値の概念

この Q 学習では報酬を得られたかどうかにかかわらず 1 つの行動ごとに学習を行う．このとき式（2.1）を用いて学習を行う．式（2.1）では報酬の他に，行動した先の状態が持つ Q 値を用いて行動の良し悪しを決めている（Fig.2.6）．これによって Q 学習は以下のような特徴を持つ．

・ 最適な行動を見つけることが可能

Q 学習で用いられる Q 値は有限回の更新によって，ある最適な値に収束することが証明されている．よって目的を達成するための最適な行動を見つけることが可能になっている．ただし，Q 値が収束するまでの更新回数は行動選択部の手法（どのように行動を選択したか）や式（2.1）で用いられているパラメータ α ・ γ の影響を受ける．

・ 遅延報酬問題を扱える

Q 学習では行動毎にその行動の評価を行う．この時報酬のみで行動を評価しようとすると，報酬を得られなかった場合には正しい評価を行えない．そのため遷移先の状態が持つ最大の Q 値を参考にすることで評価を行う．この「遷移先の状態が持つ最大の Q 値を参考にする」ということを繰り返すことで報酬の情報が伝播する．これによって遅延報酬問題を扱うことができるようになっている．

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right] \qquad (2.1)$$

$Q(s_t,a)$：更新対象のQ値
$\max_a Q(s_{t+1},a)$：遷移先の状態が持つ最大のQ値
s_t：遷移する前の状態
s_{t+1}：行動後の遷移先状態
a_t：行動
r_{t+1}：報酬
α：学習率$(0.0 \leqq \alpha \leqq 1.0)$
γ：割引率$(0.0 \leqq \gamma \leqq 1.0)$

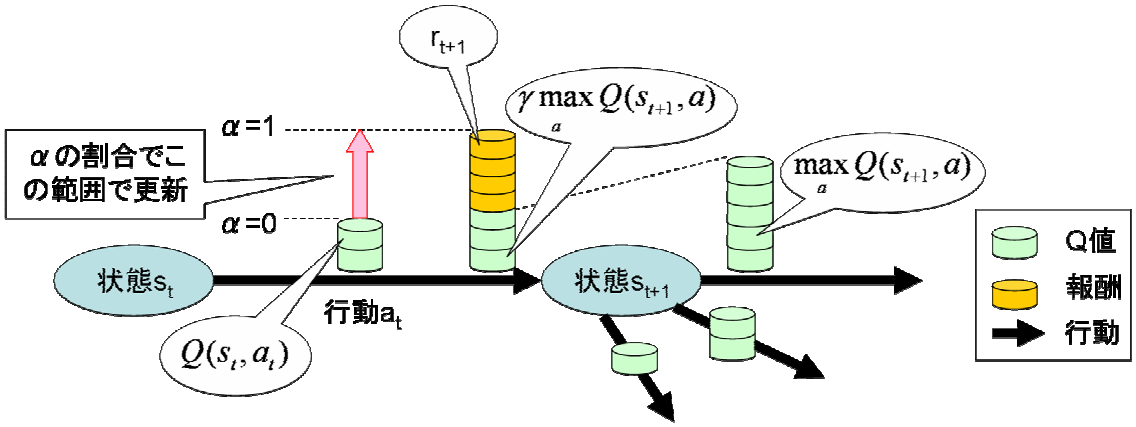


Fig.2.6：Q 学習による Q 値更新

2.3　強化学習・行動選択部

ここでは強化学習における行動選択部の話をする．行動選択部ではロボットやエージェントが行動を選ぶ部分となっている．この時ロボットは学習部で行っている学習の結果（価値関数）を基に行動を選択する．

行動選択部のアルゴリズムは，どのように行動を選択するかという点からいくつかの手法がある．今回は greedy 法，ε-greedy 法，Pursuit 法について説明をする．

A)　greedy

greedy とは「強欲な」という意味であり，その名のとおり greedy 手法では最も評価の高い行動を取る．つまり，ある状態で取れる行動の中で最も高い報酬が得られると思われる行動を選択する（Fig.2.7）．

この手法の特徴は行動の評価に忠実に従うことである．これによってある状態において各行動の評価に差が生まれると，その状態においては行動が一意的に決まる．しかしこれでは，その他の行動がもっと良い行動である可能性については考慮しない．そのため，状態数が多い場合や行動の選択肢が多い場合には向かない．

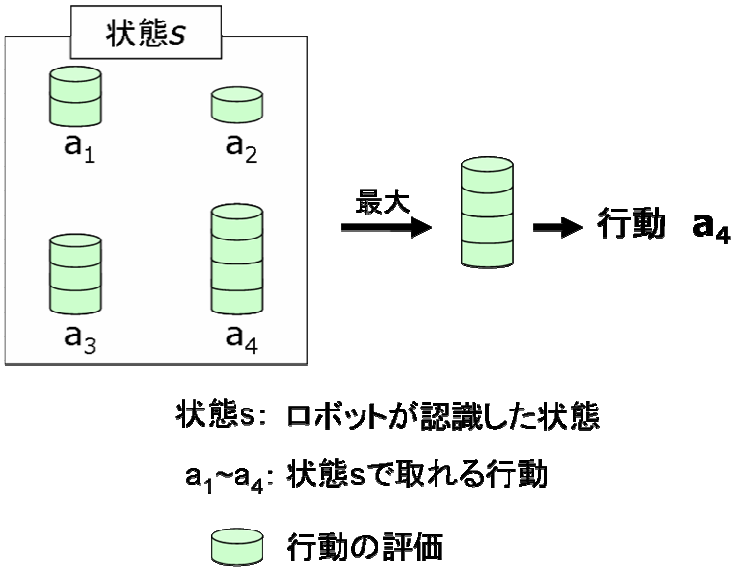


Fig.2.7：greedy 法

B)　ε-greedy

ε-greedy は greedy 手法にεの確立でランダムな行動を取るようにしたものである．この時εは 0 以上 1 以下の値であるので(1-ε)の確率で greedy 法と同様に最も高い報酬が得られるだろう行動を取ることになる（Fig2.8).

この手法の特徴はεの設定を任意で決めることができるところにある．εの値を大きく

設定してやれば様々な状態を経験することが可能である．これはどの行動がよりよい選択なのかを探索するということである．また，ε を小さく設定してやれば greedy な行動を取りやすくなる．つまりより多く経験をさせて最適な行動を見つけさせるのか，少ない経験でも良いのでより早く目的を達成させるのかを ε で調節できる．

このような特徴から行動選択の手法としてよく用いられる手法の一つとなっている．

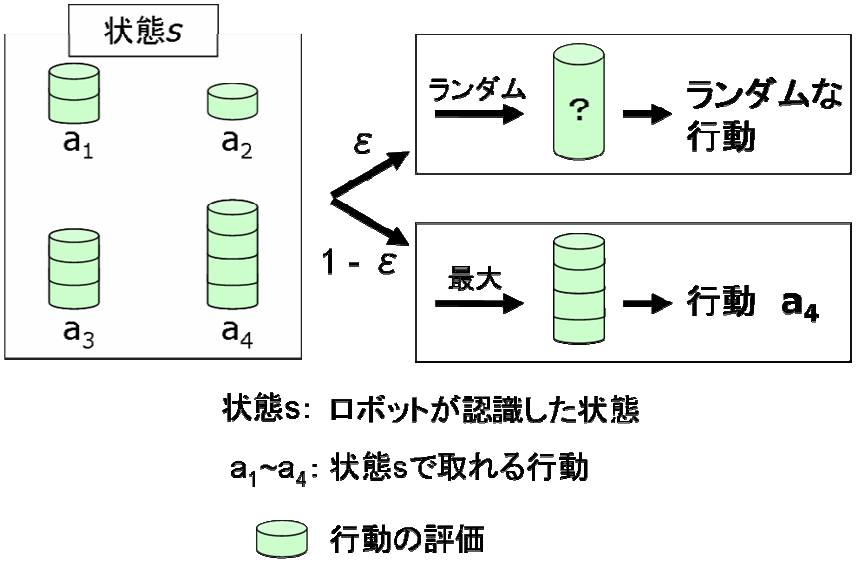


Fig.2.8： ε -greedy 法

C) Pursuit

Pursuit ではある状態 s での行動 a を選択する確率を表す $P_s(a)$を用いる．この確立 $P_s(a)$は行動毎に更新される．最も行動の評価が大きいものに対しては式(2.2)によって確率が 1 に近づくように更新される．それ以外に対しては式(2.3)で更新され，確率が 0 に近づくように更新される．更新される例として Fig.2.9 を示す．Pursuit ではこの確率 $P_s(a)$に基づいて行動を決める．

$$P_s(a_{\max}) \leftarrow P_s(a_{\max}) + \beta[1.0 - P_s(a_{\max})] \tag{2.2}$$

$$P_s(a_{another}) \leftarrow P_s(a_{another}) + \beta[1.0 - P_s(a_{another})] \tag{2.3}$$

ここで a_{max} は選択できる行動の中で最も Q 値が大きい行動で，$a_{another}$ はそれ以外の行動である．また，β は確立の変動幅を決めるパラメータであり，0 以上 1 以下の値をとる．この β の値が大きいとより早く確率を収束させることになる．逆に β が小さい場合には様々な状態を経験しやすくなる．

この手法の特徴は確立を用いて行動を決定することと，その確率が行動の評価によって更新されていくことである．この手法では学習初期段階では様々な行動を取るようになっており，学習が進むほど greedy 法に近い行動の取り方になる．また，β を調整することでより多く経験をさせて最適な行動を見つけさせるのか，少ない経験で目的を達成させるのかを決めることができる．

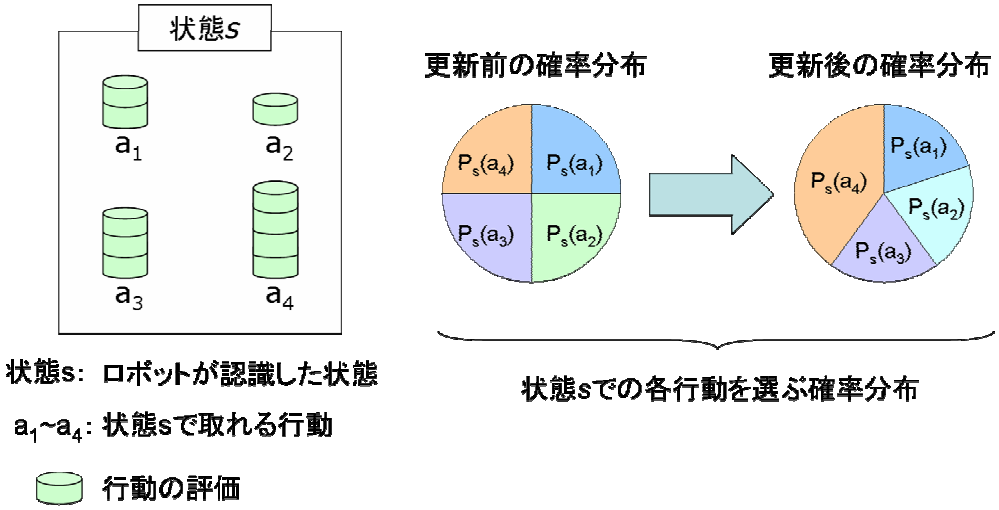


Fig.2. 9: Pursuit 法における確率 Ps(a)の更新例

2.4　強化学習の問題点

強化学習は報酬による学習であり，それによって様々な特徴・利点がある．しかし，報酬による学習は万能ではなく，以下のような課題も抱えている．

- ある目的に対してのみの学習

  強化学習では報酬を設定することである目的に対して試行錯誤を通して学習を行う．しかし，試行錯誤している中で経験したとしても，目的とは異なることについては学習を行わない．例えば街のある場所から駅までの道をロボットが学習する場合（Fig.2.10），ロボットは駅につくと報酬を貰うことができる．この時試行錯誤から駅だけでなくコンビニや郵便局などの様々な場所を通過しているはずである．しかし，強化学習ではコンビニなどの通過した場所への道を学習することは無い．最終的に学習するのは駅に向かうための道のみである．このことは後述する「果たすべき目的が変化すると対応が遅れることがある」とも関係してくる．

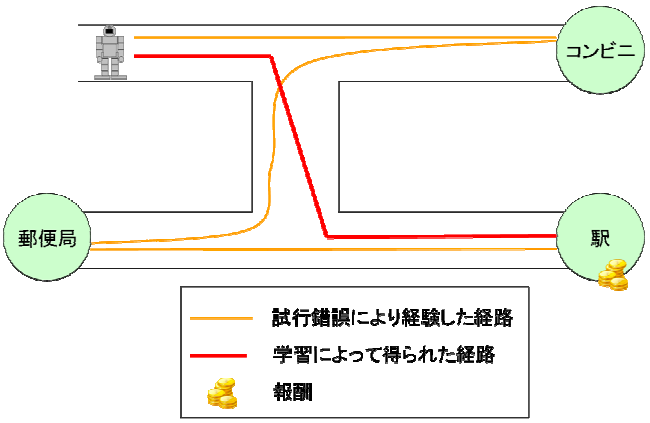


Fig.2.10：経験した経路と学習によって得られた経路

・ 果たすべき目的が変化すると対応が遅れることがある

「ある目的に対してのみの学習」で述べたとおり，強化学習では報酬で設定していない目的に対しての学習は行わない．そのため目的が変わった場合（報酬設定を変えた場合），それまでの学習で経験していたとしても 1 からの学習になってしまう．場合によっては，変わる前の目的に対する学習の結果から影響を受けて新たな目的に対する学習がスムーズに行われないことがある．上で記述した例の続きになるが，例えば駅への道を学習した跡にコンビニへの道を学習させようとする場合を考える．つまり駅に行っても報酬を貰えず，コンビニへ行けば報酬を貰えるようにした場合である．この時ロボットはそれまでの学習から「駅へ行けば報酬がもらえる」と認識しているので駅へ向かう．しかし，駅へ行っても報酬が貰えないので結局 1 からコンビニへの道を学習しなくてはならない．さらにコンビニへの道を学習した後で，再度駅への道を学習させる場合にも同様のことが起こる．このように目的が変化するような場合においては過去に経験しているはずの情報を有効に活用できていないと言える．

・ 学習にかかる手間（行動する回数）が大きい

強化学習では試行錯誤をすることで目的の状態へと学習を進める．この時ロボットが直面している環境が大きいほど経験する状態は多くなる．経験する状態が増えてくると，試行錯誤する回数は増えてしまう．また，各状態において取れる行動が多くなっても試行錯誤する回数は必然的に増えてくる．このように環境が大きくなったり，行動の選択肢が増えたりするほど学習にかかる手間が大きくなってしまう傾向にある（Fig.2.11）．

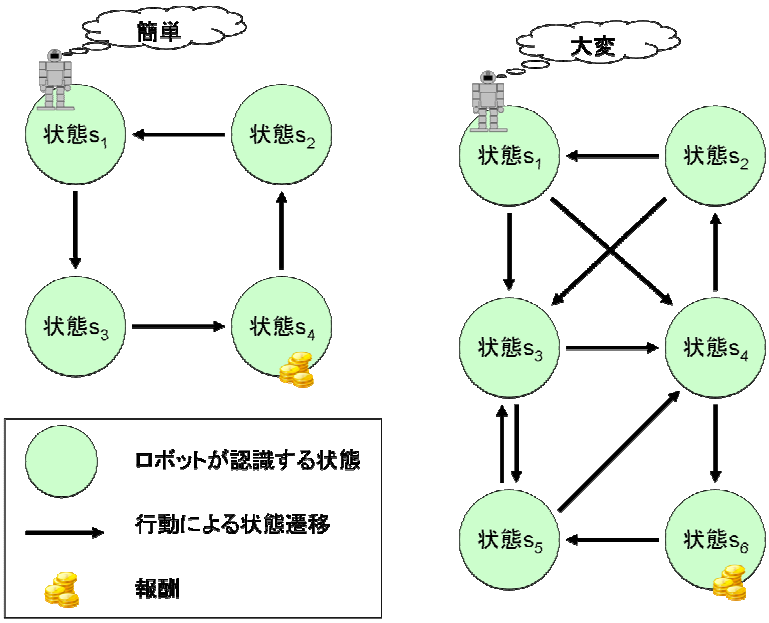


Fig.2.11： 認識する状態・行動の選択肢の多さと学習の手間

第 3 章 報酬非依存型知識の提案

2 章では強化学習について述べ，強化学習は報酬によって学習を行うということを述べた．しかし，報酬によって学習を行うため，環境が変わらなくても目的が頻繁に変わるような問題には対応が追いつかないことがあるなどの問題点があることも述べた．これらの問題点は報酬による学習によって引き起こされる問題である．本論文ではこれらの問題点を解決するために報酬に依存しない学習を提案する．そこで本章では報酬に依存しない学習で扱う情報である報酬非依存型知識を提案し，報酬非依存型知識の利用の方法を提案する．

3.1 報酬非依存型知識の概要

強化学習における問題点である「目的が頻繁に変わる場合」への対応を素早く行うために扱う情報として報酬非依存型知識を導入する．強化学習の問題点は報酬による学習によって引き起こされている．そのため報酬（目的）に依存しない情報を知識化し，報酬非依存型知識と定義した．報酬非依存型知識とはロボットが認識する状態とロボットの行動による状態遷移に注目した情報である．この報酬非依存型知識を獲得することで環境に対する学習を強化学習とは別に行う．そして問題点の解決のために獲得した報酬非依存型知識を用いることになる．概念図を Fig3.1 に示す．Fig.3.1 ではロボットは環境から行動の結果として報酬を受け取り，自分の状態をセンサによって読み取る．この時ロボットはセンサで読み取った情報から目的に依存しない情報を報酬非依存型知識として蓄える．蓄えた報酬非依存型知識はロボットが目的に対してどのように学習していけばよいかを予測するために利用する．

報酬非依存型知識の詳しい定義を 3.2 節で述べる．どのように報酬非依存型知識を利用するかのアプローチを 3.3 節で述べる．3.4 節では報酬非依存型知識の具体的な獲得方法について述べる．また，具体的な報酬非依存型知識の利用方法は 3.5 節で述べる．3.6 節では 3.5 節で述べる方法以外での報酬非依存型知識の利用について述べる．

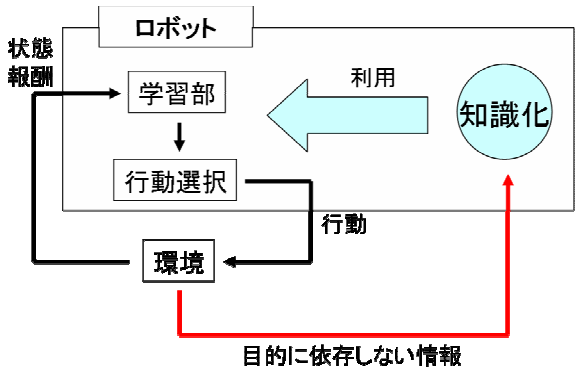


Fig.3.1： 提案システムの概念図

3.2　報酬非依存型知識の定義

この節では提案するシステムで用いる報酬非依存型知識を定義する．この報酬非依存型知識はある状態行動対に対して一つ存在する．そのため，いくつもある報酬非依存型知識をロボットが保持するためのテーブルが必要になる．よって報酬非依存型知識を保持するための知識テーブルもこの節で定義する．

3.2.1　報酬非依存型知識として扱う情報

報酬非依存型知識は 3.1 節で述べたように目的に無関係な情報を取り扱う．そのため報酬非依存型知識として注目したのはロボットが置かれている環境についての情報である．環境の中でもロボットが認識する状態についての情報に注目した．しかし，環境として重要なことはある状態と別の状態がどのようにつながっているかである．そのため，ロボットが認識する状態と，ロボットが取る行動の組み合わせからどの状態へ移り変わるかを報酬非依存型知識として扱う．例として Fig.3.を挙げると，ロボットが認識する環境の状態は「電気が付いている」となる．このようにロボットが認識する状態と行動についての情報を報酬非依存型知識として扱うことで環境への学習を行っていく．

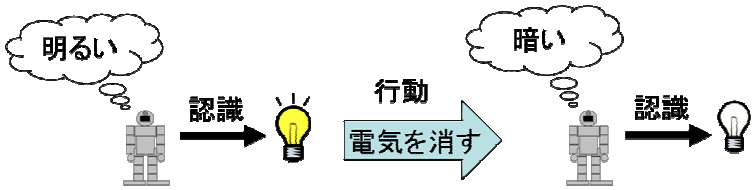


Fig.3.： 報酬非依存型知識として扱う情報

3.2.2　定義：報酬非依存型知識

本論文では報酬非依存型知識を以下のように定義する．

（状態 S，行動 a）→（次状態 S’）

ここで，「状態 S」とはロボットが認識した状態のことを示し，「行動 a」は「状態 S」の時にロボットが取ることのできる行動のひとつを示している．また，「次状態 S’」は「状態 S」の時に「行動 a」を取ることで遷移する先の状態である．この報酬非依存型知識はロボット自身が認識する状態と，その時に選択可能な行動の一つとの状態行動対によってどの状態に遷移するのかを表している．ここでは報酬非依存型知識はある状態行動対がどの状態に遷移するのかを確定情報として定義している．

Fig3.を例として考えると，以下に挙げる２つの情報が報酬非依存型知識となる．

- （状態 S_0，電気をつける）→（状態 S_1）
- （状態 S_1，電気を消す）→（状態 S_0）

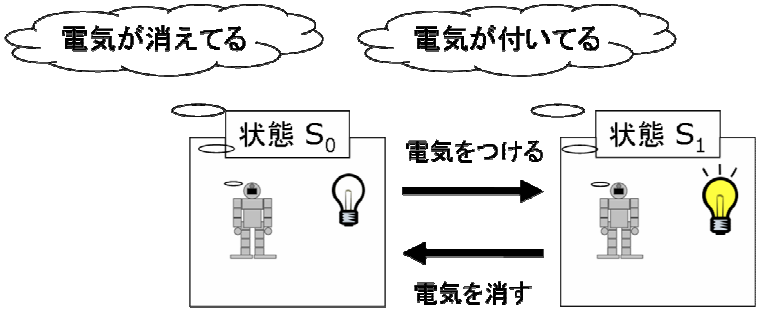
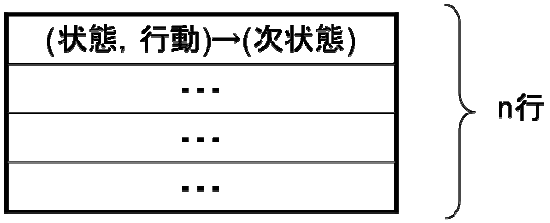


Fig.3.： 電気の点灯を認識できるロボットの状態の認識と遷移の例

報酬非依存型知識は目的とは関係無く，どの状態へ遷移するかを確定的に現している．そのため環境が別の環境に変わらない限り永続的に知識として扱うことが可能である．一方で，強化学習で用いている価値関数（Q 学習の場合は Q 値）も状態行動対に対して意味を与えているが，価値関数の場合は状態行動対が目的に対してどの程度価値があるのかを表している．そのため価値関数は目的が変わると新たな目的に対して利用できない．

3.2.3　定義：知識テーブル

ここでは報酬非依存型知識を保持するための知識テーブルについて定義する．知識テーブルは以下のように定義する．



定義した知識テーブルについての特徴を以下に挙げる．

- 知識テーブルは保持する報酬非依存型知識の数分の行を持つ

  知識テーブルは各行に報酬非依存型知識を１つずつ保持する．そのため保持する報酬非依存型知識が n 個ある場合には n 行の知識テーブルになる．

・ 知識テーブルの行数はいつでも拡張可能である

ここで定義した知識テーブルの行数は知識テーブルを所有するロボット自身がいつでも拡張可能なものとする．

・ 重複する報酬非依存型知識を持たない

知識テーブル内にある報酬非依存型知識の重複は存在しない．この重複する報酬非依存型知識とは報酬非依存型知識の定義において「状態」・「行動」・「次状態」の３つ全てが同じものを指す．「状態」・「行動」・「次状態」の内１つ又は２つ同じ場合は別の報酬非依存型知識とする．

3.3 強化学習における報酬非依存型知識の利用方法

本論文では強化学習において報酬非依存型知識を利用するシステムを提案する．そのためこの節では強化学習を用いたシステムにおいて報酬非依存型知識を利用する場合，どのように利用していくのかについて記述する．

3.3.1 強化学習における報酬非依存型知識の利用：アプローチ

報酬非依存型知識を利用する目的は強化学習において様々なタスクに素早く対応させることである．

学習初期段階において強化学習では過去に行った行動を再度行ったり，過去に経験した状態を何度も経験したりすることが多くなってしまう．このような行動や状態の経験は無駄であることが多い．特にロボットにとってタスクが変わった場合，前のタスクで経験した行動や状態を新たなタスクのために再度経験しなければならない．そこで本論文では，これらの無駄な試行錯誤を減らすことを目的として報酬非依存型知識を利用する．無駄な試行錯誤を減らすためには，ロボットが学習の最終的な目的に対してどのように学習していけば良いかを知れば良い．そこで Fig.3.5 のように報酬非依存型知識を用いて学習の最終的な目的に対する状態遷移の予測を行う．つまり，ロボットがどのように学習していけばよいかロボット自身が報酬非依存型知識から予測を行う．本論文ではロボットがどのように学習を行っていけばよいかの予測を強化学習で用いる価値関数を更新することで行う．予測した行動を取りやすいように価値関数を更新することで学習の効率化を図る．

また，提案するシステムの構成図を Fig.3.6 に示す．提案するシステムではまず報酬非依存型知識を蓄える．そして蓄えた報酬非依存型知識を利用することで価値関数を更新する．

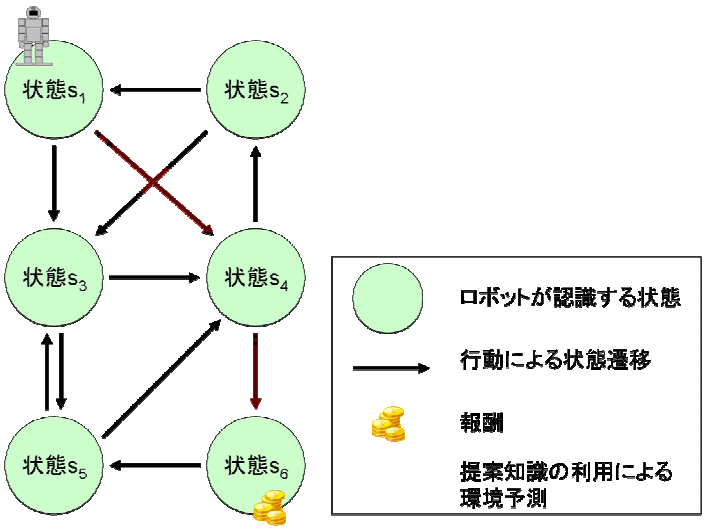


Fig.3.5： 報酬非依存型知識利用のアプローチ

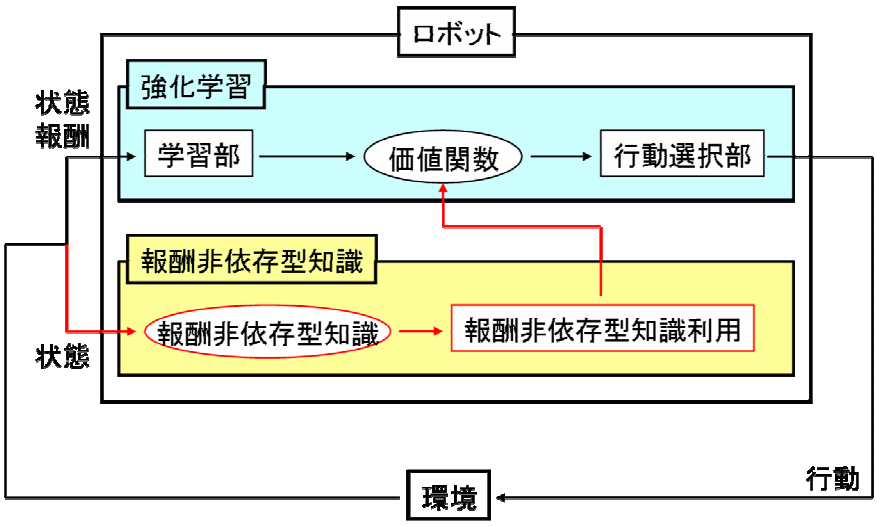


Fig.3.6： 提案システム構成図

3.3.2　強化学習における報酬非依存型知識の利用：流れ

本論文で提案するシステムの流れを fig.3.7 に示す．Fig.3.7 において強化学習を表す部分は「認識」→「学習」→「行動」である．これに対して報酬非依存型知識を用いた部分は「報酬非依存型知識の獲得」→「報酬非依存型知識の利用」となる．

報酬非依存型知識を利用するには，報酬非依存型知識を所持している必要がある．そのために報酬非依存型知識を用いる場合には大きく分けて２つの部分に分かれる．一つは「報酬非依存型知識の獲得」，もう一つは「報酬非依存型知識の利用」となる．「報酬非依存型知識の獲得」については 3.4 節でロボットが実際にどのようにして報酬非依存型知識を獲得

するのかを詳しく述べる．「報酬非依存型知識の利用」は 3.5 節で具体的な利用の方法について述べる．

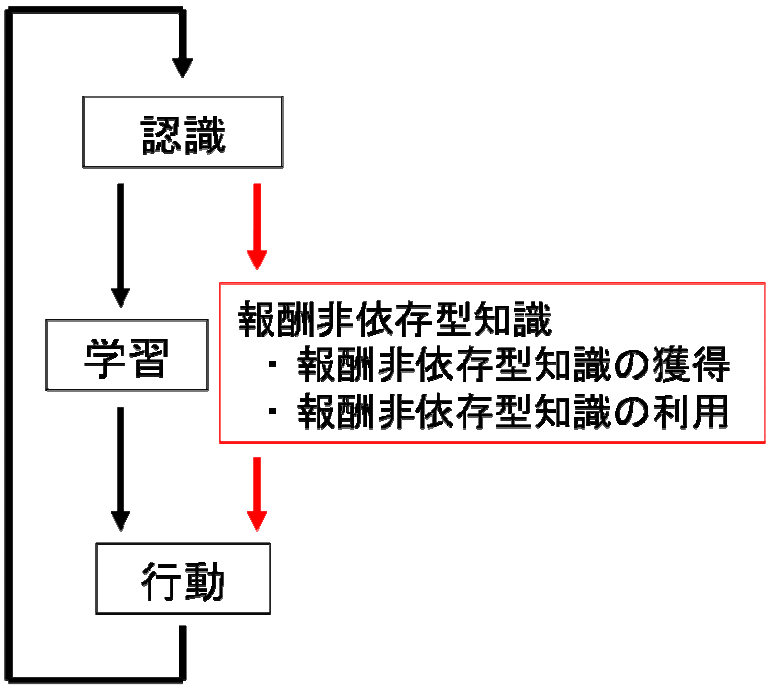


Fig.3.7：強化学習における報酬非依存型知識利用の流れ

3.4　報酬非依存型知識の獲得

この節では 3.2 節で定義した報酬非依存型知識の獲得方法について記述する．3.2 節でも記述したとおり，報酬非依存型知識は状態行動対によってどのように状態遷移が起こるかを表した情報となっている．そのため報酬非依存型知識を得るためにはロボットは実際に行動する必要がある．また，ロボットはセンサを通じて行動の結果どのように状態が遷移したかを認識する必要がある．つまり，ロボットは実際に行動し，状態を認識することで報酬非依存型知識を獲得する．本論文ではロボットは行動毎に自分自身が持つ知識テーブル内にその行動による状態の遷移情報を報酬非依存型知識として追加することとする．

ロボットが報酬非依存型知識を獲得する際に，同じ報酬非依存型知識が知識テーブル内に存在しないかを検索する．もし同じものがあった場合は，その報酬非依存型知識を再度知識テーブルに追加することは無い．

報酬非依存型知識を知識テーブルに追加する場合は，知識テーブルは報酬非依存型知識を保持するために新たに 1 行増える．本論文では知識テーブルの大きさについては定義し

ていない．そのため本論文ではロボットは報酬非依存型知識を無限の数だけ持つことが可能となっている．

例として Fig.3.8 を示す．Fig.3.8 ではロボットはまず「状態 S」と認識する状態にある．そこでロボットは「行動 a」を取ることで「状態 S'」へと状態が遷移する．この時ロボットは自分自身が持つ知識テーブル内にこの報酬非依存型知識が無いかを調べる．無かった場合には知識テーブルに追加する．

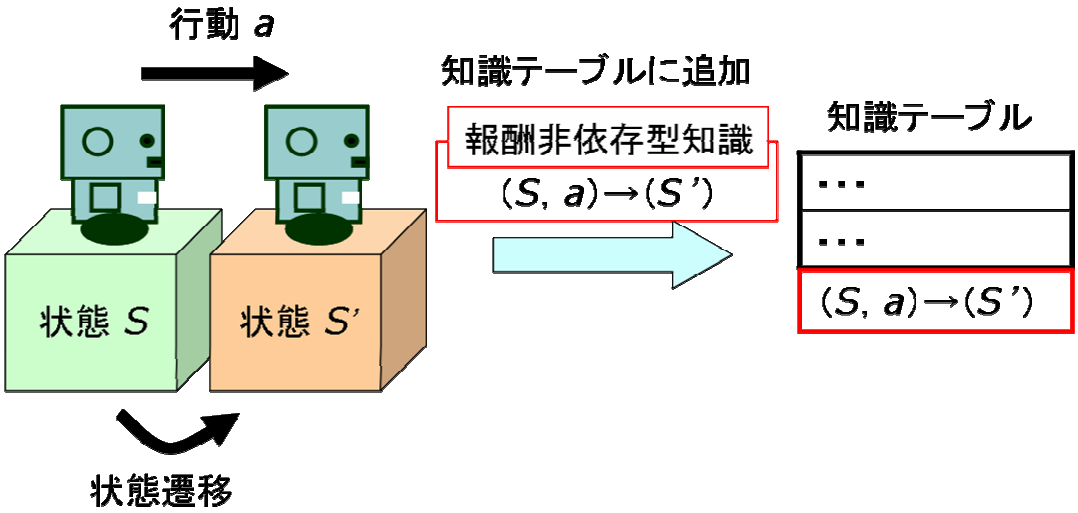


Fig.3.8：報酬非依存型知識の獲得例

3.5　報酬非依存型知識の利用

本節では報酬非依存型知識の具体的な利用方法について記述する．さらに，一般的な迷路問題を例にどのように利用されているか記述する．

3.5.1　報酬非依存型知識の利用の流れ

まず，報酬非依存型知識の利用の流れについて記述する．報酬非依存型知識の利用の流れは以下の 3 ステップに分かれる．また，Fig.3.9 に報酬非依存型知識の利用の流れの例を示す．

Step1：報酬を獲得する
Step2：知識テーブル内で報酬を得た状態までの遷移ルートを探す
Step3：Step2 で見つけたルートに対して価値関数の対応する部分を更新する

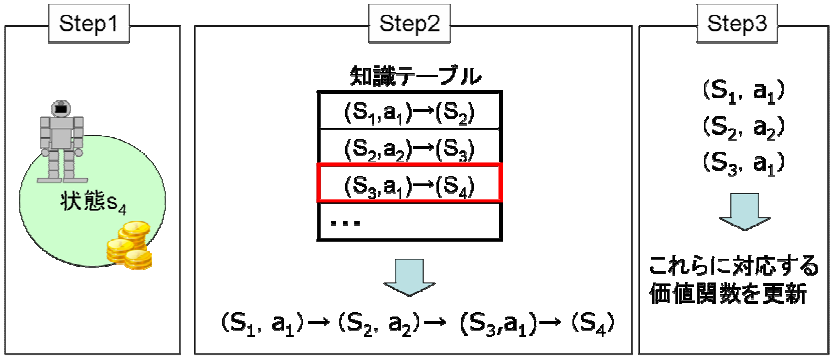


Fig.3.9：報酬非依存型知識の利用の例

報酬を獲得することが報酬非依存型知識を利用する際のトリガーとなる．報酬を獲得したら Step2 へ移る．

Step2 では知識テーブルの中において全ての状態から報酬を得られた状態までどのように遷移していけば良いかを検索する．この Step2 では報酬を得られた状態までの遷移ルートを探すが，見つかるルートは一つとは限らない．その例を Fig.3.10 に示す．Fig3.10 の左側が環境を表している．この環境下において状態 S_4 で報酬を獲得できるとすると，状態 S_4 までの遷移ルートは複数考えられる．実際に知識テーブルに複数のルートを見つけ出せるだけの報酬非依存型知識がある場合には Step2 で見つかるルートは複数になる．

Step3 では Step2 で見つけたルートに対して各状態行動対に対応する価値関数を更新する．ただし，同じ目的（同タスク）の場合，過去に見つけたルートに対して更新は行わない．つまり同じ目的を複数回達成した場合，過去の目的達成時に更新したルートに対しては更新を行わない．そして目的が変わった場合には新たな目的に対する新たなルートが見つかることになり，このルートに対しては更新を行うことになる．具体的にどのように更新されるかは 3.5.2 句で述べる．

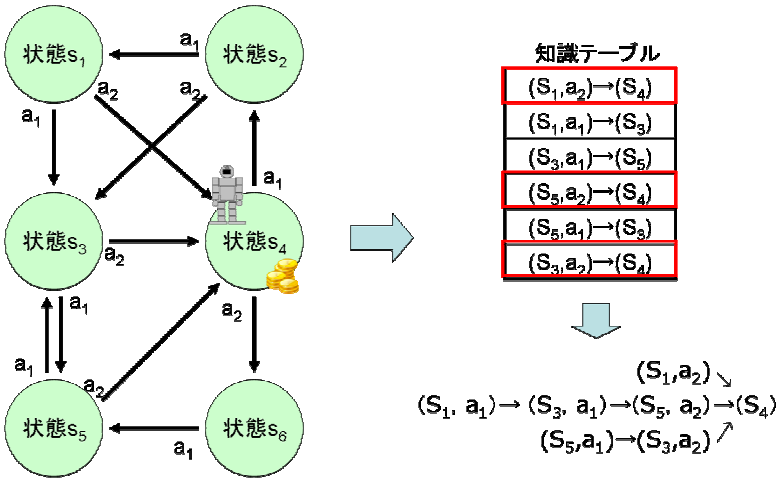


Fig.3.10：Step2 における複数ルートの発見とそのときの環境

3.5.2　報酬非依存型知識の利用における価値関数の更新

3.5.1 句で述べたように報酬非依存型知識の利用は３つのステップに分かれ，Step3 において価値関数が更新されることになる．本論文で提案するシステムでは強化学習の学習部に Q 学習を用いているので，価値関数は Q 値を表す．そのためここでは Q 値を具体的にどのように更新するかについて述べる．

Q 値を更新するための式は式（3.1），式（3.2）とした．これらの式は Q 学習に基づいた式 となっている．そのため Q 学習で用いるパラメータγを式（3.1）に含んでいる．

$$Q(s_d,a) \leftarrow Q(s_d,a) + (f \times \gamma)^{d-1} \times r \times SG(s_d) \tag{3.1}$$

$$SG(s_d) = \begin{cases} 1.0 & (s_d \text{が初期状態から目的状態のルート上}) \\ 0.5 & (\text{それ以外}) \end{cases} \tag{3.2}$$

式（3.1）において Q(sd,a)は更新対象の Q 値を表す．また，d は状態 sd から報酬をもらえた状態（目的状態）までの最短距離（最も少ない状態遷移の数）を表す．ここでの最短距離は知識テーブル内での最短となる．そのため環境における最短距離と一致するとは限らない．f は報酬非依存型知識の利用度を表すパラメータである．f は 0 以上１以下の値を取り，1 に近いほど大きな値で更新することになる．価値関数を大きな値で更新するほど強化学習における探索のための試行錯誤が減ることになる．つまり，f が 1 に近いほど目的に対する学習の完了時の状態に近くなるように価値関数を更新する．また，f が 0 に近いほど更新する値が小さくなる．f が 0 の場合は報酬非依存型知識の利用による価値関数の更新は無くなり普通の強化学習となる．よってロボットは f が小さいほど強化学習による試行錯誤を重要視し，f が大きいほど報酬非依存型知識を重要視するようになる．また，r は獲得した報酬を表している．

式（3.2）で表される SG(sd)は更新対象となる状態行動対の内の状態 sd がロボットの初期状態から目的状態までのルートの中にあるかどうかで値が変わる．このようにすることで初期状態から目的状態までのルートは重要であるとみなし，それ以外のルートに対しては探索する必要性も考慮した．

式（3.1），式（3.2）によって目的の状態に近い状態行動対ほど更新される値が大きくなるようになっている．この理由は目的の状態に近い状態行動対ほど重要であると考えられるからである．Fig3.11 に更新の例を示す．この例では S_1 を初期状態として，S_4 を目的状態とした場合である．S4 に近い状態行動対ほど大きな値で更新される．また S1 から S4 へのルートに対しては SG(sd)が 1.0 となっており，その他のルートに対しては SG(sd)が 0.5 となっている．

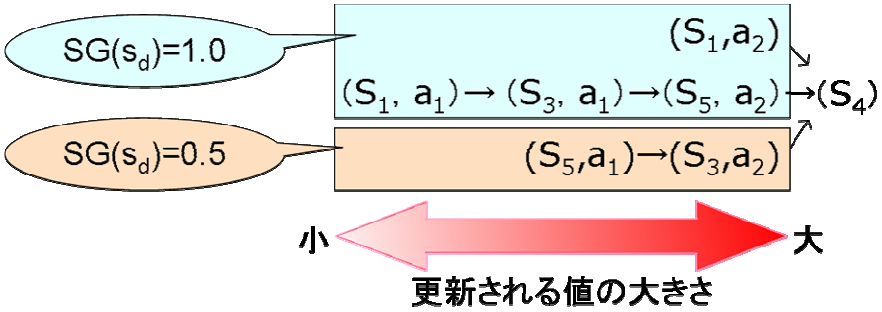


Fig.3.11： 更新の例

3.5.3　迷路問題における報酬非依存型知識の利用例

ここでは迷路問題を例に，報酬非依存型知識の利用がどのように行われているかを記述する．Fig.3.12 に迷路問題で用いられる迷路の例を示す．迷路問題ではスタートからゴールまでの道を学習することが目的となる．今回の例の場合，各マスをそれぞれ 1 つの状態とし，ゴールで報酬を得られるようになっている．

Fig.3.12 の環境下でロボットがゴールした場合の報酬非依存型知識利用による更新例を Fig.3.13 と Fig.3.14 に示す．ロボットがゴールしたときに所持している報酬非依存型知識の量によって更新される範囲が異なる．少ない場合が Fig.3.13 であり，多い場合が Fig.3.14 になる．この更新例では Q 値の初期値を 0 として，r=10, γ =0.9,f=1.0 の時の更新の例となっている．それぞれの矢印が状態行動対を表している．矢印の横に書かれている数値が式（3.1）と式（3.2）によって更新される値となる．数値が書かれていない矢印は更新されないことを表している．

更新される値に注目すると，ゴールに近い状態行動対ほど大きな値で更新されている．さらにスタートとゴールを結ぶルート以外の状態行動対に対する更新値が小さくなっている．また，報酬非依存型知識を持たない部分は更新されない．

Fig.3.12 :迷路の例

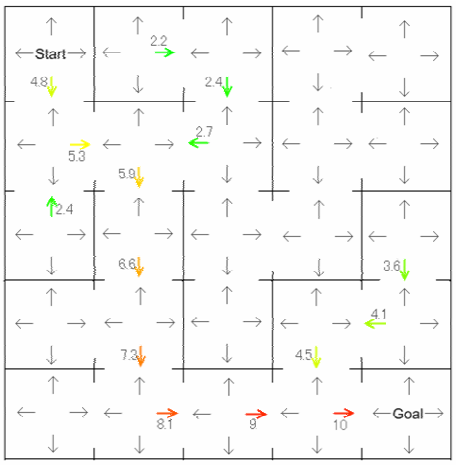


Fig.3.13：所持している報酬非依存型知識が少ない場合の更新例（r=10, γ=0.9,f=1.0）

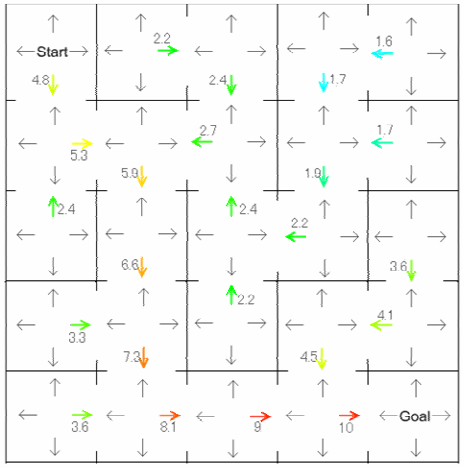


Fig.3.14：所持している報酬非依存型知識が多い場合の更新例（r=10, γ=0.9,f=1.0）

3.6　環境変化への対応のための報酬非依存型知識の利用

報酬非依存型知識は環境に対する学習であることから，報酬非依存型知識を環境変化への対応のために使える可能性がある．そのため本節では報酬非依存型知識を環境変化への対応のために利用することについて記述する．また，本論文で取り扱う環境変化についても記述する．

3.6.1　環境変化とは

まず，本論文で取り上げる環境変化について記述する．本論文では環境変化は環境構造が変化することを表す．具体的には環境が取りえる各状態の遷移のルールが変化することを環境変化とする．つまり各状態行動対による遷移先が変わることが本論文における環境変化である．環境変化の例として Fig.3.15 を示す．

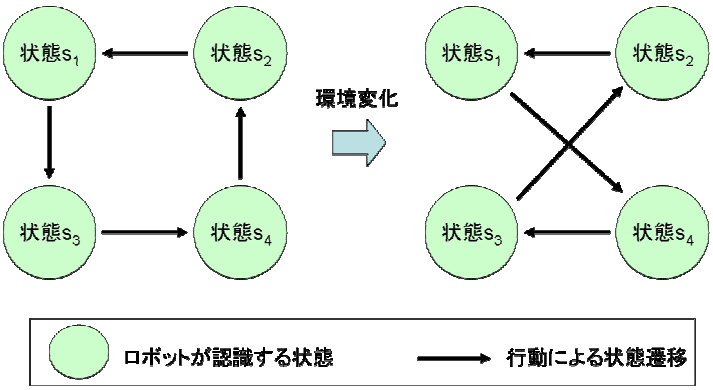


Fig.3.15：環境変化の例

3.6.2　報酬非依存型知識の利用による環境変化の認識と対応

A)　認識

報酬非依存型知識は環境についての情報であり，ある状態行動対がどのように遷移するかを表している．そのため環境変化に対して報酬非依存型知識を用いることで普通の強化学習よりも対応が早くなるのではないかと考える．

報酬非依存型知識を持っていない普通の強化学習の場合，環境変化に対応することができる仕組みになっている．しかし普通の強化学習では、ロボットは環境の変化を認識することはできない．そのため環境が変化したことに対して特別な対応を行うことは無く，環境変化への対応に時間がかかってしまう．

報酬非依存型知識を有する場合，所持している報酬非依存型知識と実際の行動の結果を照らし合わせることで環境が変化したかどうかを知ることができる．Fig.3.16 を例に詳しく説明をする．Fig3.16 で環境が変化する前に状態 S_1 で行動 a を取った場合，ロボットは報酬非依存型知識として「$(S_1,a)\rightarrow(S_3)$」を得る．その上で環境変化後に状態 S_1 で行動 a を取った場合，実際の行動で得られた結果は「$(S_1,a)\rightarrow(S_4)$」となる．この結果と同じ状態行動対(S_1,a)についての報酬非依存型知識を知識テーブルの中から探し，見つかった場合は比較を行う．今回の例では知識テーブルに「$(S_1,a)\rightarrow(S3)$」があるのでこれと実際の行動の結果と比較を行う．ロボットは比較の際に報酬非依存型知識と実際の行動の結果が異なれば環境が変化したと認識する．この時環境が変化したと認識する部分は状態行動対(S_1,a)に対してのみである．

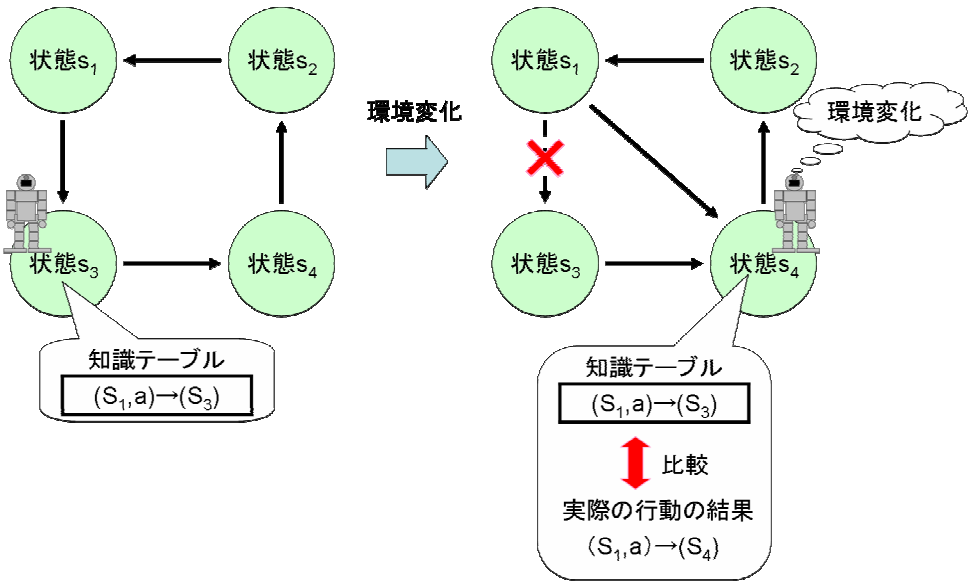


Fig.3.16：環境変化の認識の例

B)　対応

報酬非依存型知識を用いることで環境変化を認識することが可能であることは前述したとおりである．そこでここからは環境変化を認識した際にどのような対応を取るかを述べる．

環境が変化した場合は変化した状態行動対に対して目的を達成するための学習をやり直す必要がある．具体的な方法として価値関数を初期値に戻すようにした．価値関数をリセットすることでその部分について再学習がスムーズに始まるようにするためである．

また環境が変わったので，知識テーブル内にある変化があった部分の報酬非依存型知識を新しい環境で認識した情報に更新する必要もある．古いほうの報酬非依存型知識を新しい報酬非依存型知識で上書きする形になる．

Fig.3.17 に環境変化を認識した場合の対応の例を示す. Fig.3.17 では環境変化を認識したため，ロボットは知識テーブル内の報酬非依存型知識を更新し，価値関数の対応する部分を初期化する．

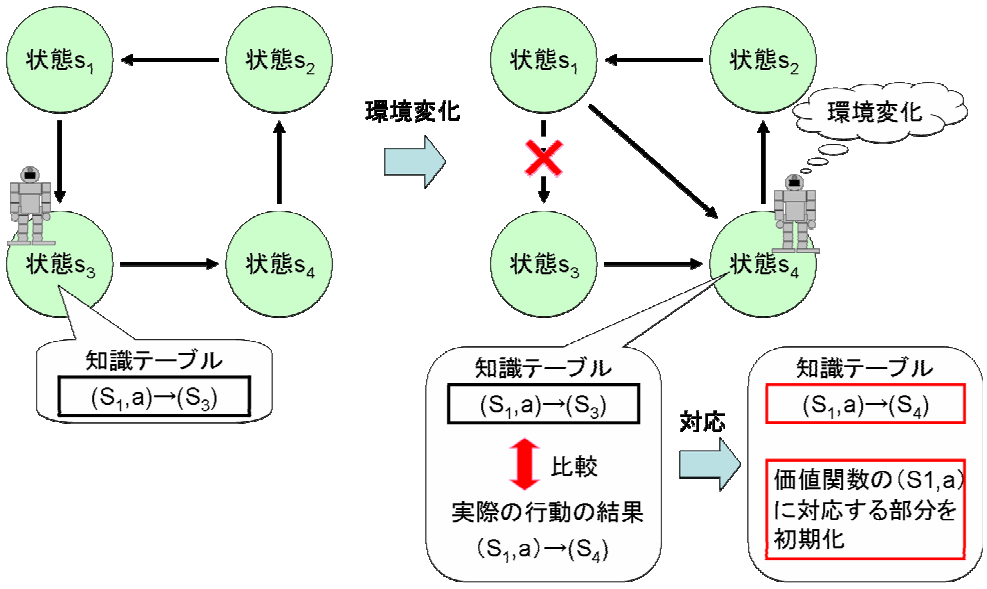


Fig.3.17 :環境認識した際の対応例

第 4 章 提案手法を用いた迷路問題での実験

4.1 実験概要

4.1.1 概要

提案システムの有用性を示すためにシミュレーション実験を行った．本実験ではタスクとしてゴール変化を伴う迷路問題を用いる．このタスクではロボットはスタートからゴールまでより少ない行動数で行けるように学習する．この迷路問題に報酬非依存型知識を有する提案システムと強化学習のみのロボットを適用する．本実験ではロボットがゴールするまでを 1 試行とし，1 試行ごとの行動数に注目して結果を比較していくことで提案システムの有効性を示す．概要の図を Fig.4.1 に示す．

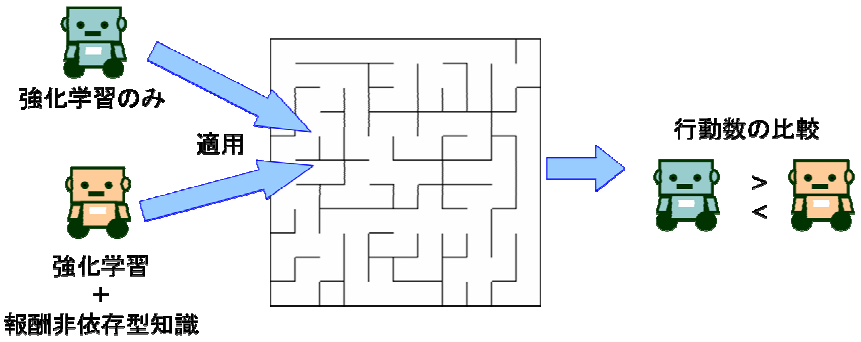


Fig.4.1： 実験概要

4.1.2 タスク

本実験ではタスクとして「ゴール変化を伴う迷路問題」を用いた．このタスクではスタートは固定でゴールの場所は 1 箇所となる．ただし，このタスクでは一定試行ごとにゴールの位置が変わるような迷路問題となっている．この設定は目的が変わっても素早く対応できるということを示すための設定である．Fig.4.2 に例を示す．

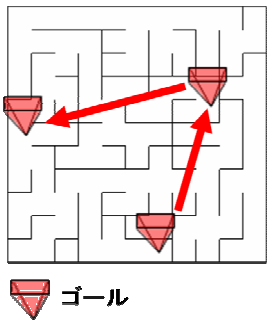


Fig.4.2： ゴール変化を伴う迷路問題

ゴールが変化する際にゴールの位置はランダムに決められる．ただし，本実験ではゴールが変化するパターンは強化学習のみのロボットと報酬非依存型知識を有するロボットで同じものを用いる．これはゴールの位置が異なっていると正しく結果を比較できないためである．また，ゴールが変化するまでの試行回数は実験ごとに設定するものとする．

また本論文ではロボットが認識する状態を迷路の各マスに対応させる．Fig.4.3 に 3×3 マスの迷路の各マスと各状態(S0～S8)の対応の例を示す．

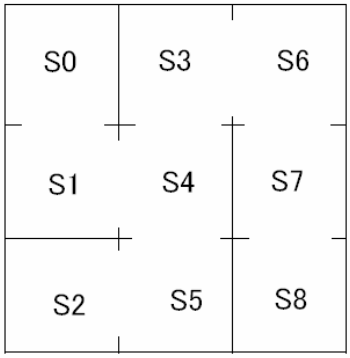


Fig.4.3：迷路問題における各マスと各状態の対応例

4.1.3　ロボットの設定

本実験で用いるロボットの設定を記述する．本実験はシミュレーション実験なので実ロボットは持ちいらない．今回ロボットは迷路のどのマスにいるかを状態として認識することができる．ロボットは各状態において「上へ移動」・「下へ移動」・「左へ移動」・「右へ移動」の４つの行動を取ることができる．本実験ではこの４つのうちのどれかの行動を取ることを１行動とする．壁にぶつかる方向への行動の場合は，行動する前の状態と行動したあとの状態は同じ状態になる．ただし，この行動も１行動とする．

また，強化学習の学習部には最もよく使われる Q 学習を用いる．行動選択部は代表的なものとして ε -greedy と Pursuit の２つを用いた． ε -greedy と Pursuit でそれぞれ結果の比較のため，強化学習のみのロボットと報酬非依存型知識を有するロボットを用意する．報酬非依存型知識を有するロボットについてはパラメータ f の違いによる結果の違いを見るため２種類用意した．以下に比較するロボットについてまとめる．

・ 行動選択部に ε -greedy を用いるロボット

Robot A. 強化学習のみ

Robot B. 強化学習＋報酬非依存型知識（f=0.5）

Robot C. 強化学習＋報酬非依存型知識（f=1.0）

・ 行動選択部に Pursuit を用いるロボット

Robot D. 強化学習のみ

Robot E. 強化学習＋報酬非依存型知識（f=0.5）

Robot F. 強化学習＋報酬非依存型知識（f=1.0）

本論文では結果を出力する際には ε-greedy を用いた場合の Robot A・B・C について比較したものと，Pursuit を用いた場合の Robot D・E・F で比較したもので分けている．

4.1.4 実験の種類

これまでの設定を共通の設定とした上で本論文では静的環境と動的環境についてそれぞれ実験を行う．さらに静的環境については環境の構造の違いに注目して２種類の実験を行う．動的環境では１種類の実験となる．静的環境での実験については迷路のサイズが「16×16」・「32×32」・「64×64」で実験を行った．動的環境下での実験については「16×16」の迷路サイズのみで実験を行った．以下に実験の種類についてまとめる．

・ 静的環境

・実験１：迷路構造が複雑な環境

・実験２：迷路構造が単純な環境

→それぞれ「16×16」・「32×32」・「64×64」の迷路サイズで実験

・ 動的環境

・実験３：迷路構造が一定周期で変化する環境

→「16×16」の迷路サイズで実験

以降は静的環境下における実験（実験１・実験２）と動的環境下における実験（実験 3）に分けて，環境（迷路構造）の説明・パラメータ設定の説明・実験結果の表示・実験結果の考察を行う．静的環境下での実験の話を 4.2 節，動的環境下での実験の話を 4.3 節で記述する．

4.2　静的環境下での実験

本節では静的環境下での実験について詳しく述べる．静的環境下での実験は迷路構造の違いから２種類の実験に分かれている．迷路構造が複雑な環境下での実験を実験１とし，迷路構造が単純な環境下での実験を実験２としている．そのため 4.2.1 句で実験１について述べ，4.2.2 句で実験 2 について述べる．

4.2.1　実験１

A)　目的

本実験では報酬非依存型知識を利用することで目的が変わっても素早く対応できるということを示す．

B)　環境（迷路構造）

ここでは実験１で用いる迷路構造の特徴について述べる．今回実験１で用いる迷路の構造の特徴としては迷路構造が複雑なものとなっている．具体的に述べると，任意の１マスから同じマスを１度しか通らないで別のマスまで行くルートは１通りとなっている．Fig.4.4 にこの例を示す．Fig.4.4 では左上のマスから右下までのマスまでのルートは１通りとなっている．全てのマスにおいてこのように別のマスまでのルートが１通りとなった迷路が実験１で用いる迷路になっている．

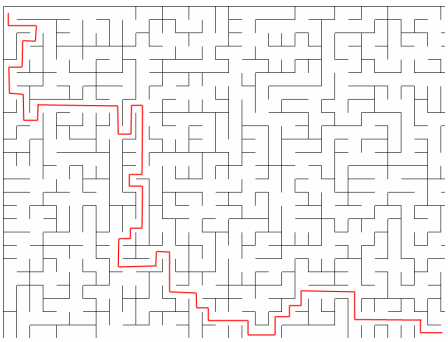


Fig.4.4：任意の１マスから別のマスまでのルートが１通りの例

このような迷路では強化学習を用いたロボットにとってゴールすることが難しくなっている．つまりゴールまでの道のりを学習しきるまでに時間がかかってしまう迷路構造となっている．しかし，ロボットにとってこのような迷路構造だとゴールまでの最短ルートを見つけやすくなっている．これらの理由はスタートからゴールまでのルートは１通りだからである．

また，実験１では迷路のサイズとして「16×16」・「32×32」・「64×64」の迷路路でそれぞれ実験を行った．Fig.4.5(a)～(ｃ)がそれぞれ実際に使用した迷路である．各迷路におい

て色つきのマスはゴールとなる位置を現している．色つきのマスに書かれている数字は何番目のゴールかを表している．各迷路におけるゴールが変化するまでの試行回数についてはパラメータ設定で記述する．

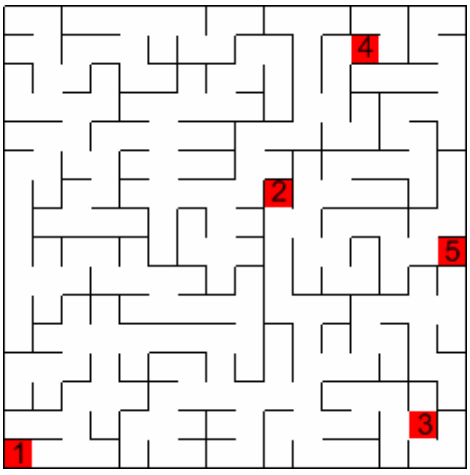


Fig.4.5(a)：16×16

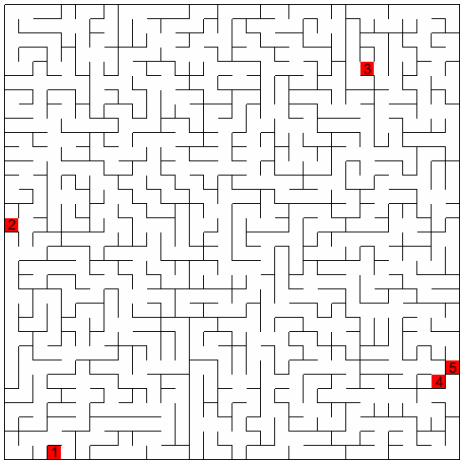


Fig.4.5(b)：32×32

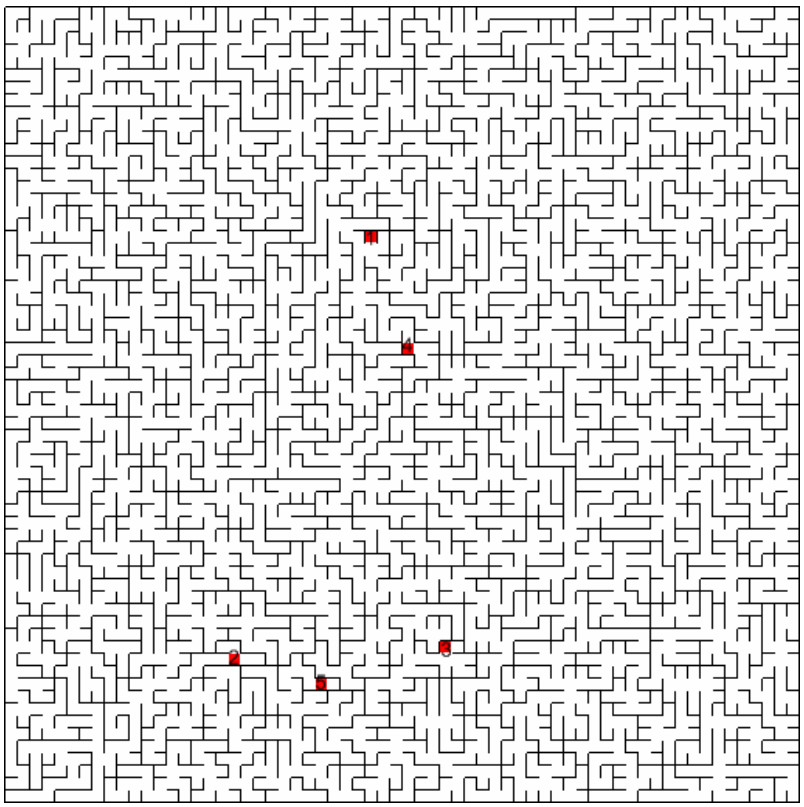


Fig.4.5(c)：64×64

Fig.4.5： 実験で用いた迷路

C) 実験パラメータ設定

実験 1 で用いたパラメータ設定を表 4.1～表 4.4 に記す．実験 1 では迷路サイズの異なる迷路を用いて 3 つの実験を行う．そのため 3 つの実験について共通する設定を表 4.1 に記し，各迷路固有の設定を表 4.2～表 4.4 に記す.

表 4.1：全てのサイズの迷路に共通な設定

報酬（ゴールのみ）	100
実験終了までの試行数	1000
Q 値の初期値	0.001
ε (ε -greedy)	0.05
β (Pursuit)	0.7

表 4.2：迷路サイズが 16×16 の迷路での実験パラメータ設定

迷路サイズ	16×16
ゴールが変化するまでの試行数	200
α	0.5
γ	0.5

表 4.3：迷路サイズが 32×32 の迷路での実験パラメータ設定

迷路サイズ	32×32
ゴールが変化するまでの試行数	200
α	0.5
γ	0.8

表 4.4：迷路サイズが 64×64 の迷路での実験パラメータ設定

迷路サイズ	64×64
ゴールが変化するまでの試行数	200
α	0.5
γ	0.9

D) 実験結果

以下に実験結果を迷路のサイズごとに載せ，簡単に考察を行う．

（1）迷路サイズ 16×16

まずは行動選択部の手法にε-greedy を用いた実験結果を Fig.4.6・Fig.4.7 に記す．各グラフで比較している対象は強化学習のみを用いた Robot A ，強化学習に加えて報酬非依存型知識を有する Robot B（f=0.5）と Robot C（f=1.0）である．

Fig.4.6 は各試行ごとの行動数の比較を行ったグラフである X 軸が試行回数を表し，Y 軸がゴールまでにかかった行動数である．このようなグラフにおいて試行数が増えるごとに行動数が減っているということはロボットが学習をしていることを表している．Fig.4.6 ではグラフを見やすくするため Y 軸の上限を 5000 にしてグラフへ出力している．

Fig.4.6 では試行回数が 200 増えるごとに各ロボットの行動数が急激に増大している．これはゴールが別の場所へ移った影響を受けていることを示している．さらに，ゴールが変化したことによって増えたときの行動数はまったく学習をしていない状態（試行数 0 付近）からゴールを探す場合よりも多くなっていることがわかる．これは前のゴールについての学習の結果が新たなゴールに対しての学習に悪影響を与えているためである．どのロボットもゴールが変化した影響を受けてはいるが，強化学習のみのロボットより報酬非依存型を有したロボットのほうが収束が早くなり，ゴールが変化した影響を小さく抑えた結果となった．

特に Robot C についてはゴールが変わった次の試行から学習が終了した状態と変わらない行動数になることが多かった．このようになった理由としては，過去の様々なゴールに対する学習をしていくうちに報酬非依存型知識が蓄えられたためこのような結果となったと考えられる．また，迷路サイズが小さかったため学習が簡単だったことも理由の一つである．

最初のゴールについての試行である 0～199 試行目までは各ロボットで大きな差が生まれていない．初期段階では迷路を十分に探索していないためにロボットが持っている報酬非依存型知識の量が少ないためである．

4 つ目のゴールであるゴール 4（試行回数が 600～799）に注目すると Robot B と Robot C の行動数が途中で増えている．ここで Fig.4.5(a)を見ると，ゴール 4 の位置周辺はそれまでのゴールの位置とは異なっている．そのためゴール 4 付近の報酬非依存型知識が十分な量を保持していなかったと考えられる．よって Robot B と Robot C はゴール 4 周辺を学習するために行動数が増えてしまったと思われる．

また，Fig.4.7 は実験終了までの行動数の合計を比較したグラフになっている．このグラフから報酬非依存型知識を用いたロボットのほうが行動数が少なくなっていると見て取れる．

同様に行動選択部の手法に Pursuit を用いた実験結果を Fig4.8・Fig4.9 に記す．各グ

ラフで比較している対象は強化学習のみを用いた Robot D ， 強化学習に加えて報酬非依存型知識を有する Robot E（f=0.5）と Robot F（f=1.0）である．

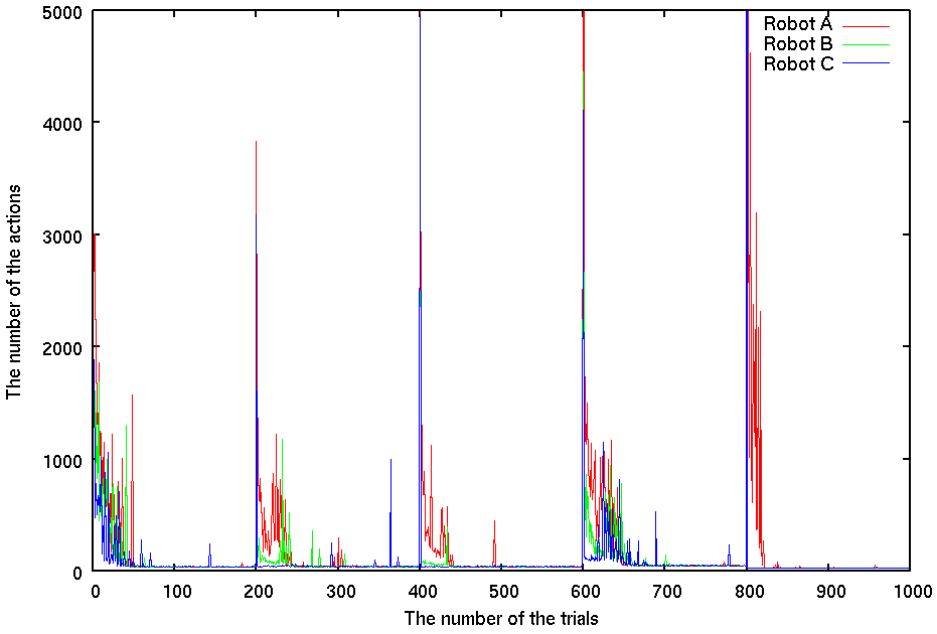


Fig.4.6：ε -greedy を用いた場合の各試行における行動数の比較

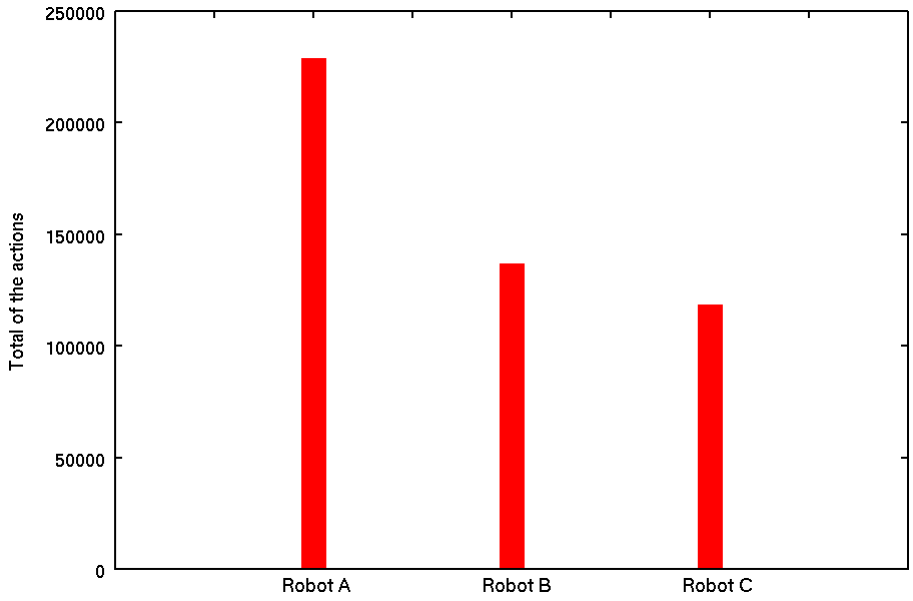


Fig.4.7： ε -greedy を用いた場合の 1000 回試行での行動数の総和の比較

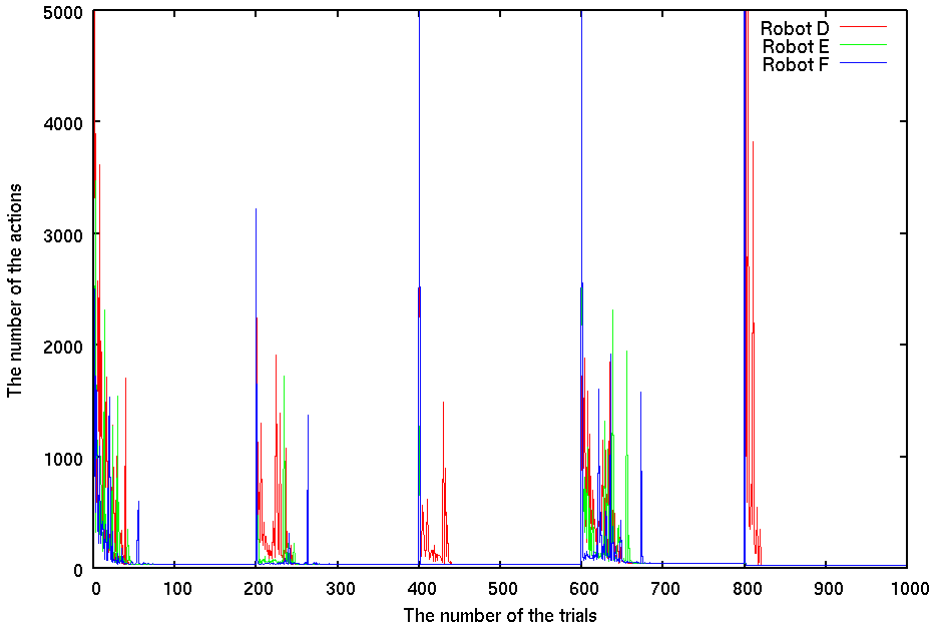


Fig.4.8：Pursuit を用いた場合の各試行における行動数の比較

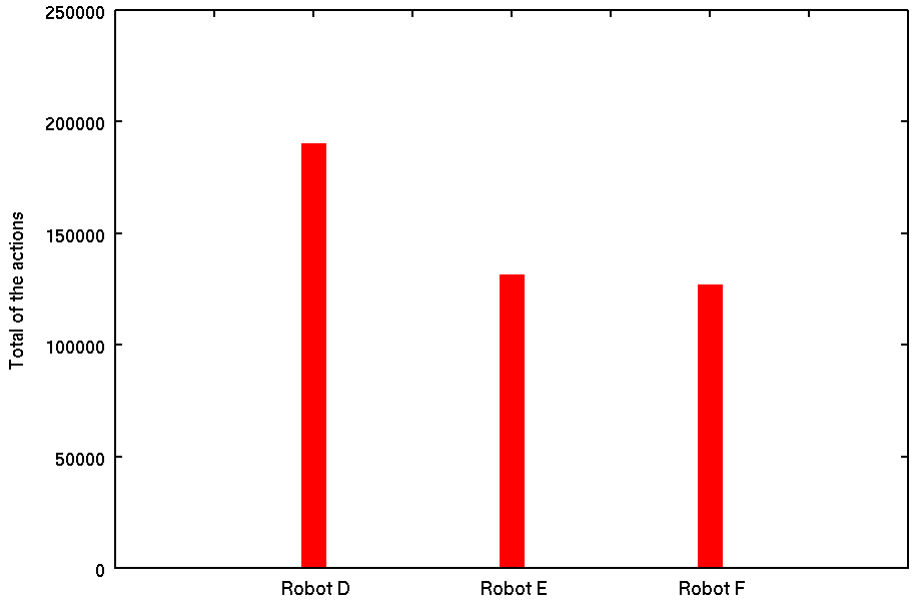


Fig.4.9：Pursuit を用いた場合の 1000 回試行での行動数の総和の比較

（2）迷路サイズ 32×32

同様に行動選択部の手法にε-greedy を用いた実験結果を Fig.4.10・Fig.4.11 に，Pursuit を用いた実験結果を Fig4.12・Fig4.13 に記す．

Fig.4.10・Fig.4.12 ではグラフを見やすくするため Y 軸の上限を 40000 にしてグラフへ出力している．

Fig.4.10 を見ると，迷路サイズが大きくなったこともあり報酬非依存型知識を有するロボットと強化学習のみのロボットとでの差が「迷路サイズが 16×16」のときと比べて大きくなった．

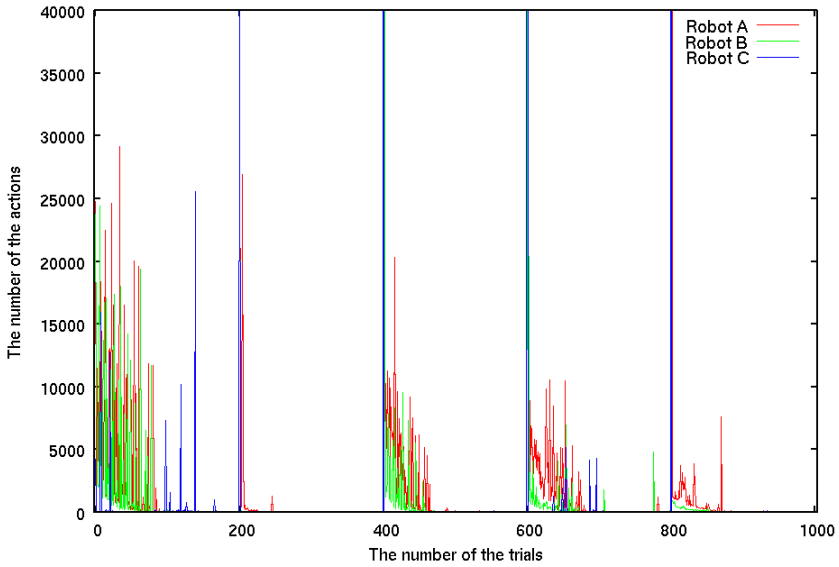


Fig.4.10：ε-greedy を用いた場合の各試行における行動数の比較

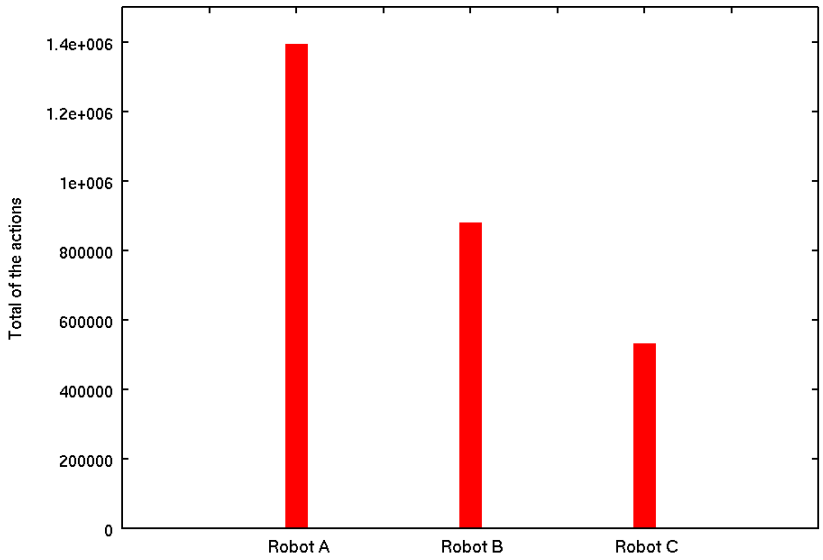


Fig.4.11： ε-greedy を用いた場合の 1000 回試行での行動数の総和の比較

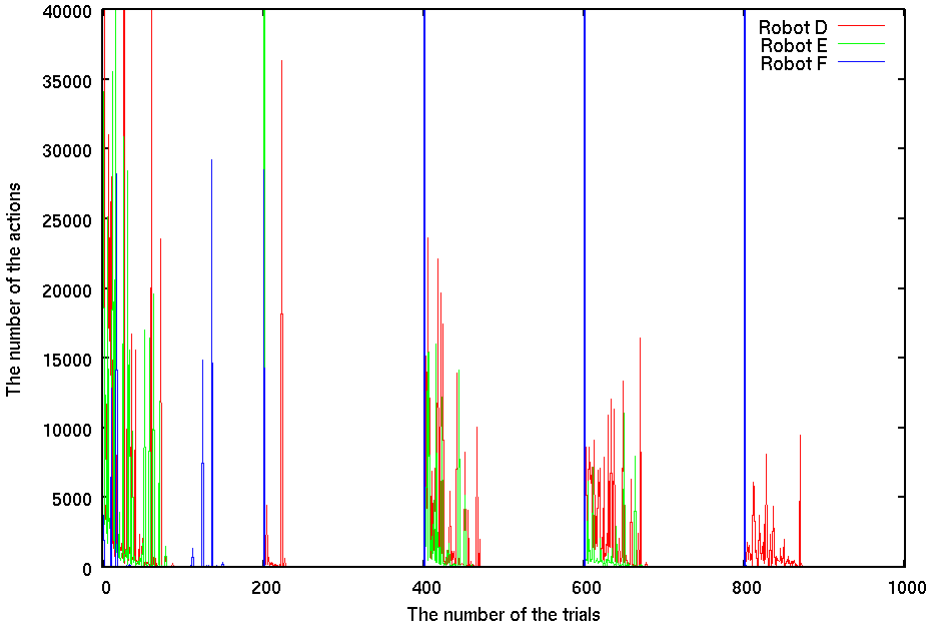


Fig.4.12：Pursuit を用いた場合の各試行における行動数の比較

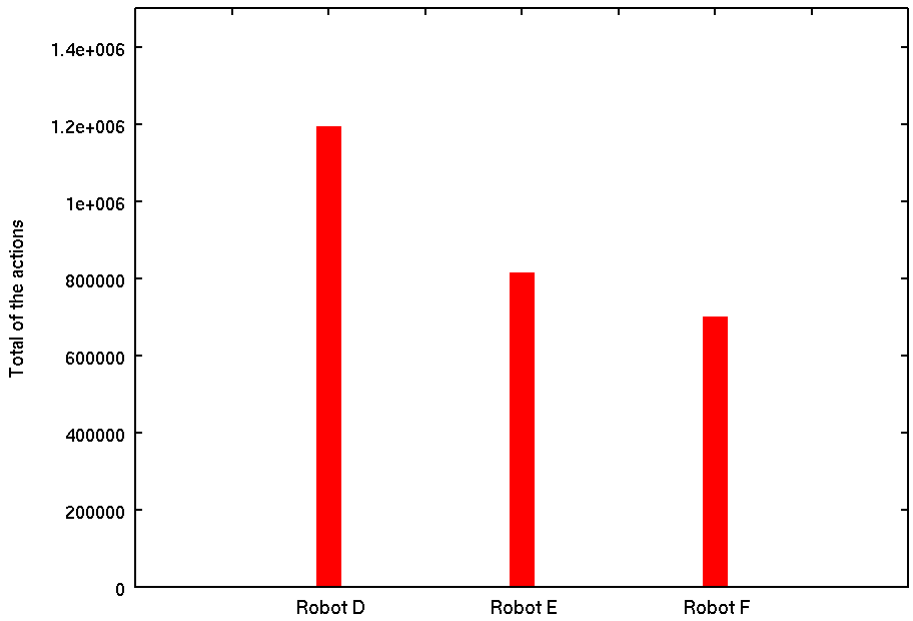


Fig.4.13：Pursuit を用いた場合の 1000 回試行での行動数の総和の比較

（3）迷路サイズ 64×64

これまでと同様に行動選択部の手法にε-greedy を用いた実験結果を Fig.4.14・Fig.4.15 に，Pursuit を用いた実験結果を Fig4.16・Fig4.17 に記す.

Fig.4.14・Fig.4.16 ではグラフを見やすくするため Y 軸の上限を 80000 にしてグラフへ出力している.

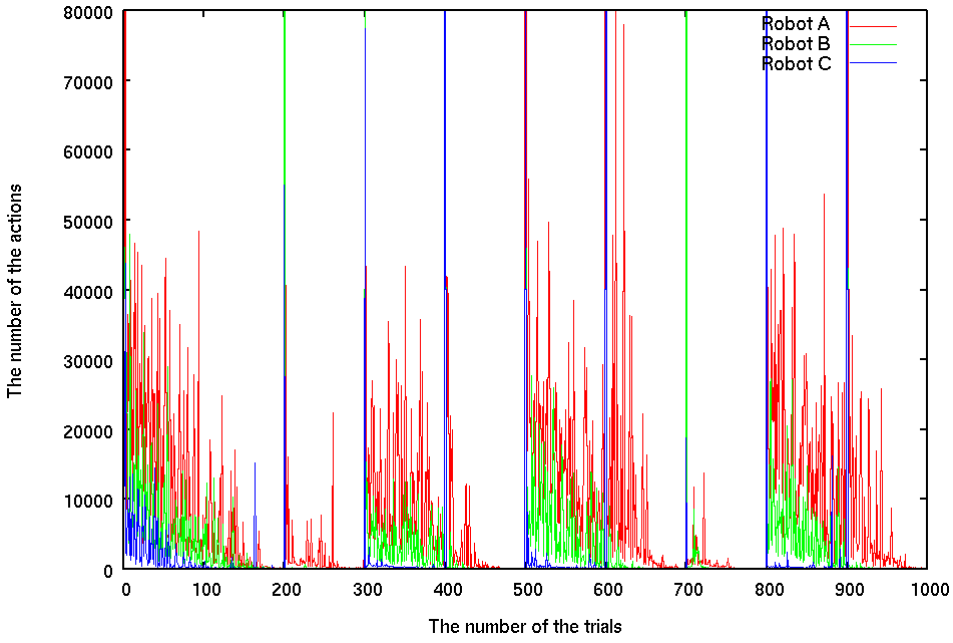


Fig.4.14：ε-greedy を用いた場合の各試行における行動数の比較

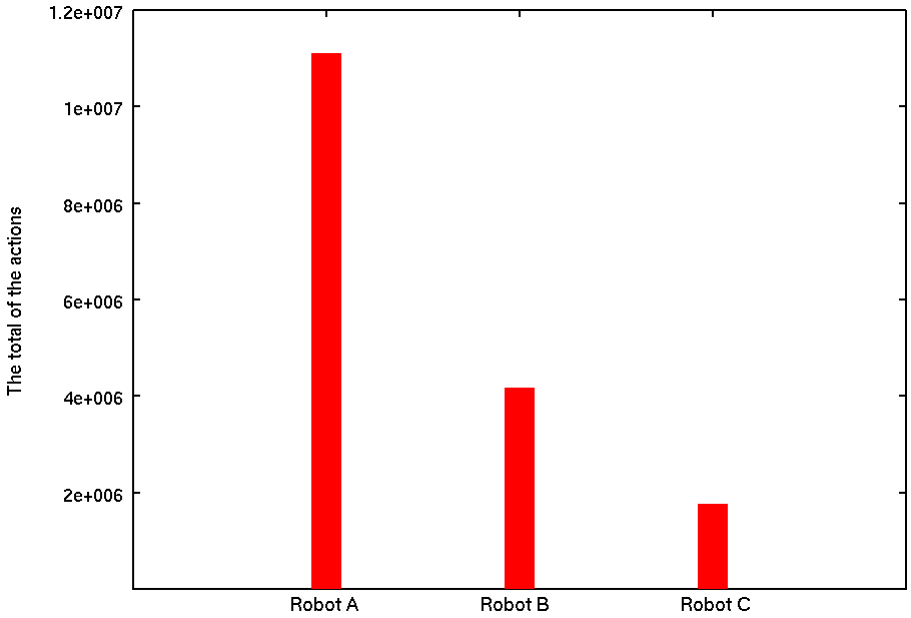


Fig.4.15： ε-greedy を用いた場合の 1000 回試行での行動数の総和の比較

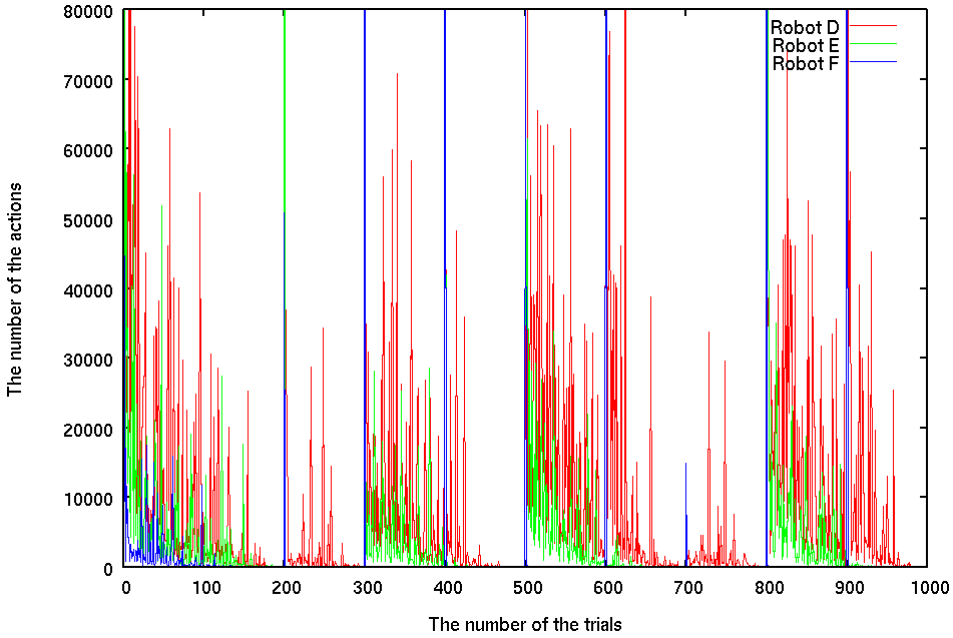


Fig.4.16：Pursuit を用いた場合の各試行における行動数の比較

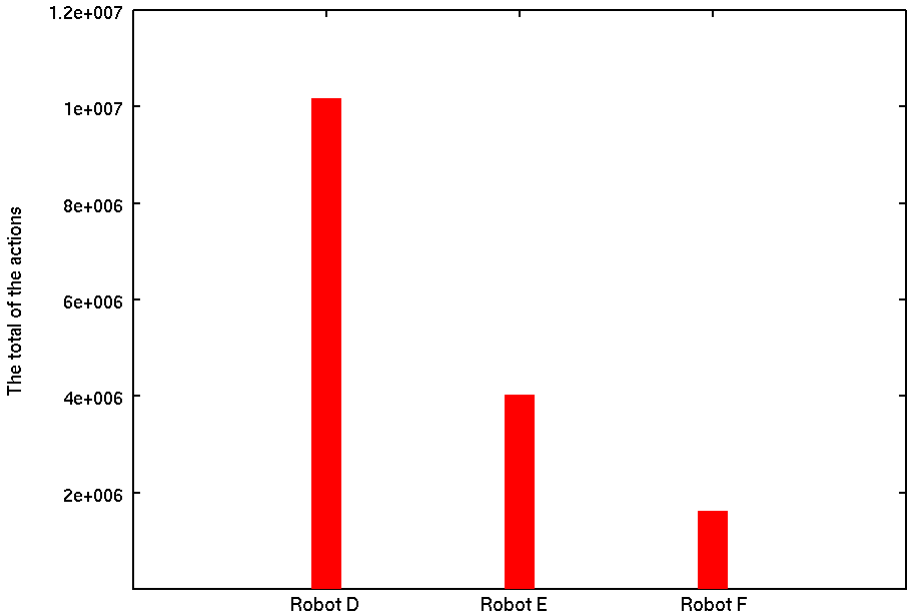


Fig.4.17：Pursuit を用いた場合の 1000 回試行での行動数の総和の比較

E)　実験結果の考察

報酬非依存型知識を用いた場合，どのサイズの迷路においても行動数の収束が早くなるということが結果として見ることができた．また学習初期段階又は最初のゴールに注目した場合，どのサイズの迷路においても各ロボット間での大きな差は見られなかった．これはこの段階では迷路を十分探索していないので，ロボットが持つ報酬非依存型知識が少ないためである．迷路のサイズが大きくなるほどこの特徴は顕著に現れていた．しかし，報酬非依存型知識を持つ場合のロボットは試行が進むほどゴール変化に強くなる傾向が見られた．

行動数の総和のグラフを見ても報酬非依存型知識を有するロボットのほうが少なくなっている．これも迷路のサイズが大きくなるほど差が大きくなっている．

これらによって報酬非依存型知識をもつロボットは報酬非依存型知識を獲得し，変化するゴールに対して有効に活用できていると言える．実験 1 の結果により報酬非依存型知識を有するロボットは静的環境下において，報酬非依存型知識を獲得することで，環境に対する学習をうまく行えていたと言える．

4.2.2　実験 2

A)　目的

実験 1 での迷路構造はゴールまでのルートが一つであった．そのためゴールまでのルートが複数ある迷路を用意し，より良いルートが見つけ出せているかを検証する．

B)　環境（迷路構造）

ここでは実験 2 で用いる迷路構造の特徴について述べる．今回実験 2 で用いる迷路の構造の特徴としては迷路構造が単純なものとなっている．具体的に述べると，任意の 1 マスから別のマスまで行くルートは複数ある構造となっている．Fig.4.12 にこの例を示す．

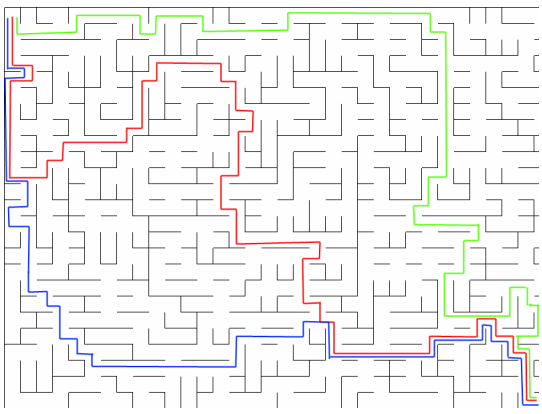


Fig.4.18：あるマスから別のマスまで複数ルートがある例

このような迷路では強化学習を用いたロボットにとってゴールすることが簡単になっている．しかし，ロボットにとってこのような迷路構造だとゴールまでの最短ルートを見つけにくくなっている．この理由はスタートからゴールまでのルート複数あるからである．

また，実験 2 では迷路のサイズとして「16×16」・「32×32」・「64×64」の迷路路でそれぞれ実験を行った．Fig.4.19(a)～(ｃ)がそれぞれ実際に使用した迷路である．色つきのマスがゴールとなる場所を指している．書かれている数字は何番目のゴールになるかを示している．

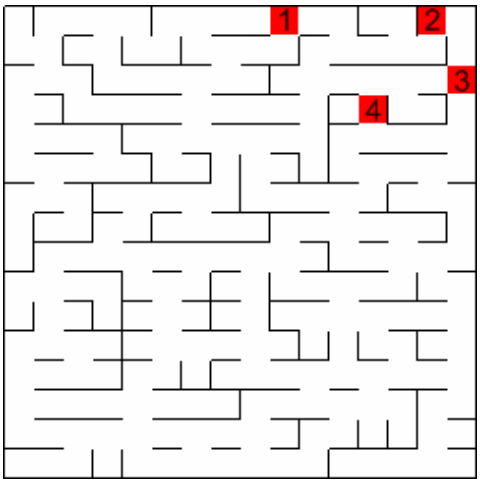


Fig.4.19(a): 16×16

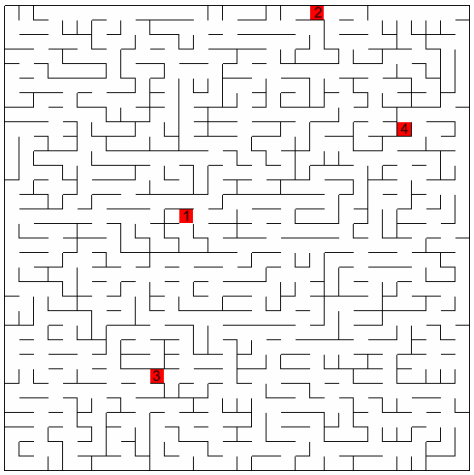


Fig.4.19(b)：32×32

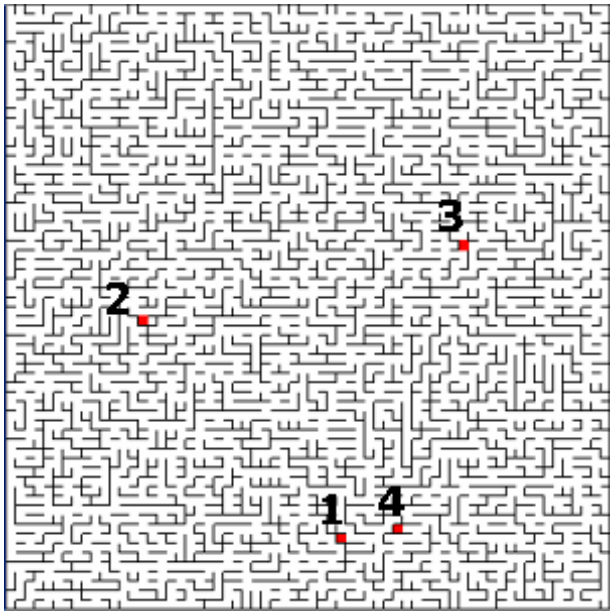


Fig.4.19(c)：64×64

Fig.4.19： 実験 2 で用いた迷路

C) 実験パラメータ設定

実験 2 で用いたパラメータ設定を表 4.5～表 4.8 に記す．実験 2 では迷路サイズの異なる迷路を用いて 3 つの実験を行う．そのため 3 つの実験について共通する設定を表 4.5 に記し，各迷路固有の設定を表 4.6～表 4.8 に記す.

表 4.5：全てのサイズの迷路に共通な設定

報酬（ゴールのみ）	100
実験終了までの試行数	1000
Q 値の初期値	0.001
ε (ε -greedy)	0.05
β (Pursuit)	0.7

表 4.6：迷路サイズが 16×16 の迷路での実験パラメータ設定

迷路サイズ	16×16
ゴールが変化するまでの試行数	250
α	0.5
γ	0.5

表 4.7：迷路サイズが 32×32 の迷路での実験パラメータ設定

迷路サイズ	32×32
ゴールが変化するまでの試行数	250
α	0.5
γ	0.8

表 4.8：迷路サイズが 64×64 の迷路での実験パラメータ設定

迷路サイズ	64×64
ゴールが変化するまでの試行数	250
α	0.5
γ	0.8

D)　実験結果

以下に実験結果を迷路のサイズごとに載せ，簡単に考察を行う．

（1）迷路サイズ 16×16

まずは行動選択部の手法にε-greedy を用いた実験結果を Fig.4.20～Fig.4.22 に記す．各グラフで比較している対象は強化学習のみを用いた Robot A ，強化学習に加えて報酬非依存型知識を有する Robot B（f=0.5）と Robot C（f=1.0）である．

Fig.4.20・Fig.4.21 については実験 1 と同じ形式の結果である．Fig.4.20 はグラフを見やすくするため Y 軸の上限を 500 にしてグラフへ出力している．

Fig.4.22 は各ゴールについて各ロボットが見つけた最短ルートの比較になっている．実験 2 ではスタートからゴールまでのルートが複数あるような環境であるのでこのような結果を出した．限られた試行回数の中でより良いルートを見つけ出せているかを各ロボットで比較している．Minimum で表されるグラフは本来の最短距離を示している．Fig.4.22 ではほぼどのロボットも最短距離を見つけることができている．ただ，ゴール 3 についてはRobot A（強化学習のみ）は最短ルートを見つけ出せていない．ゴール 3 の各ロボットの行動数を Fig4.20 で見てみると，ゴール 3 に当てはまる部分は試行回数が 500～749 までとなっている．この部分では Robot A は他のロボットよりも少し多めの行動数に収束してしまっている．これは本来の最短ルートを見つけずに別のルートに落ち着いたことを示している．こういったことは報酬非依存型知識を用いた場合でも起こることがありえる．このゴール 3 についてさらに試行を行えばいつかは最短ルートを見つけることができるはずである．そのため学習がうまくいっていなかったわけではないということになる．

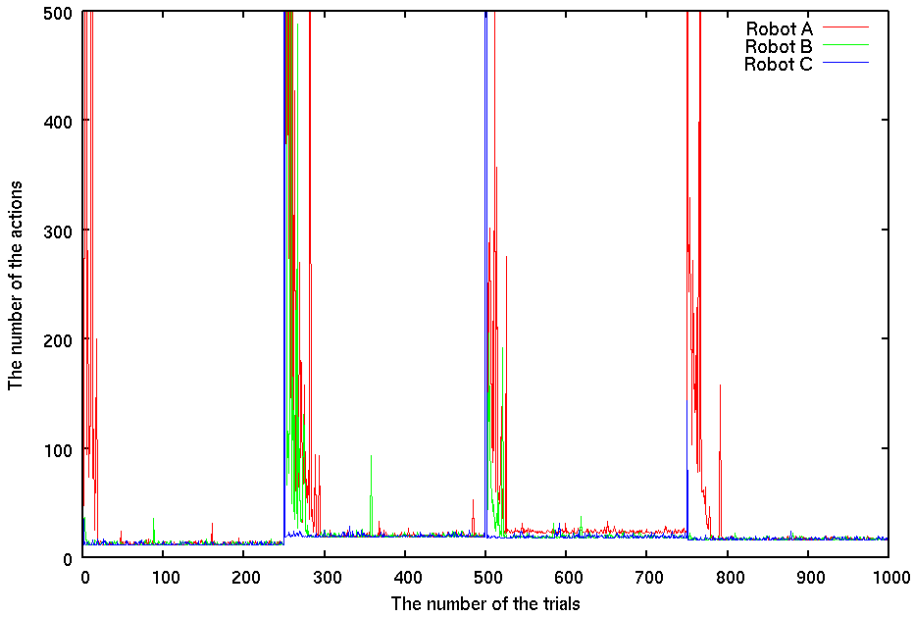


Fig.4.20：ε-greedy を用いた場合の各試行における行動数の比較

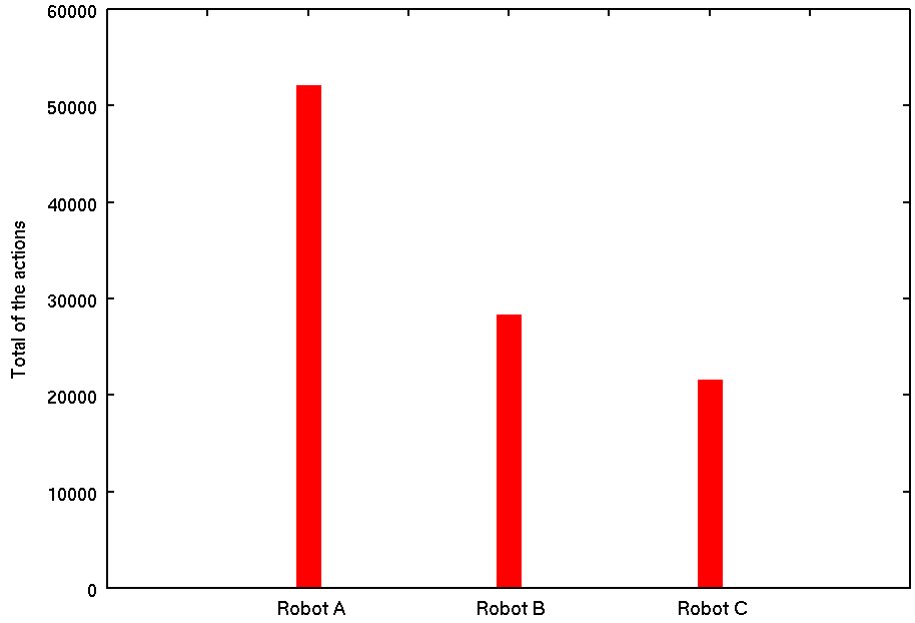


Fig.4.21： ε -greedy を用いた場合の 1000 回試行での行動数の総和の比較

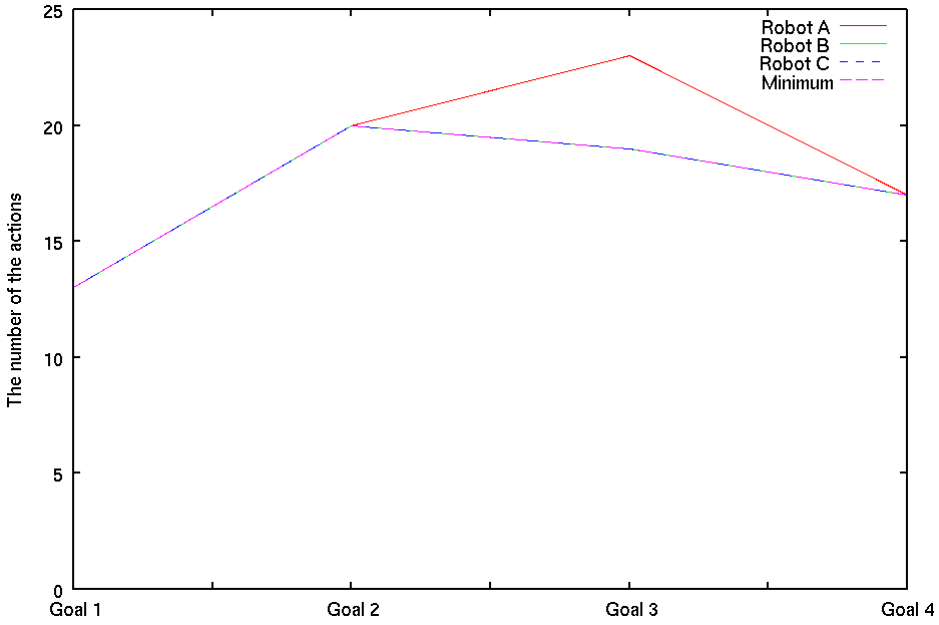


Fig.4.22： ε -greedy を用いた場合の各試行ごとの最短行動数の比較

同様に行動選択部の手法に Pursuit を用いた実験結果を Fig4.23～Fig4.25 に記す．各グラフで比較している対象は強化学習のみを用いた Robot D ， 強化学習に加えて報酬非依存型知識を有する Robot E（f=0.5）と Robot F（f=1.0）である．

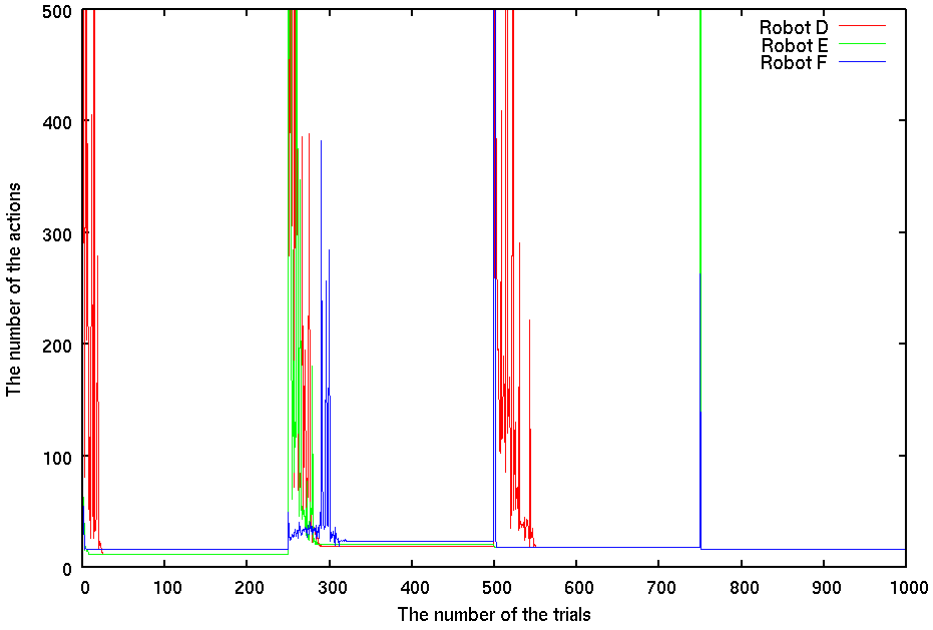


Fig.4.23：Pursuit を用いた場合の各試行における行動数の比較

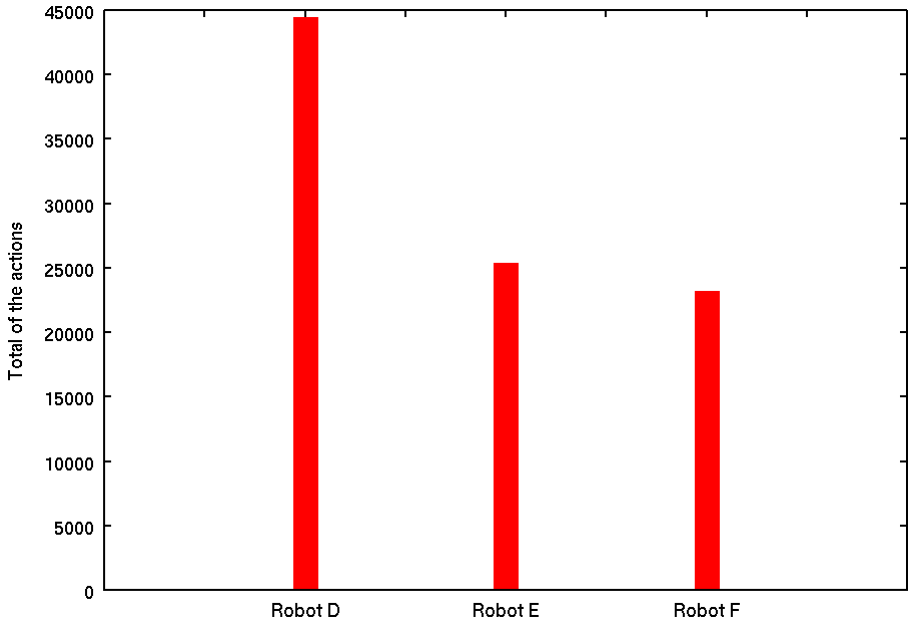


Fig.4.24：Pursuit を用いた場合の 1000 回試行での行動数の総和の比較

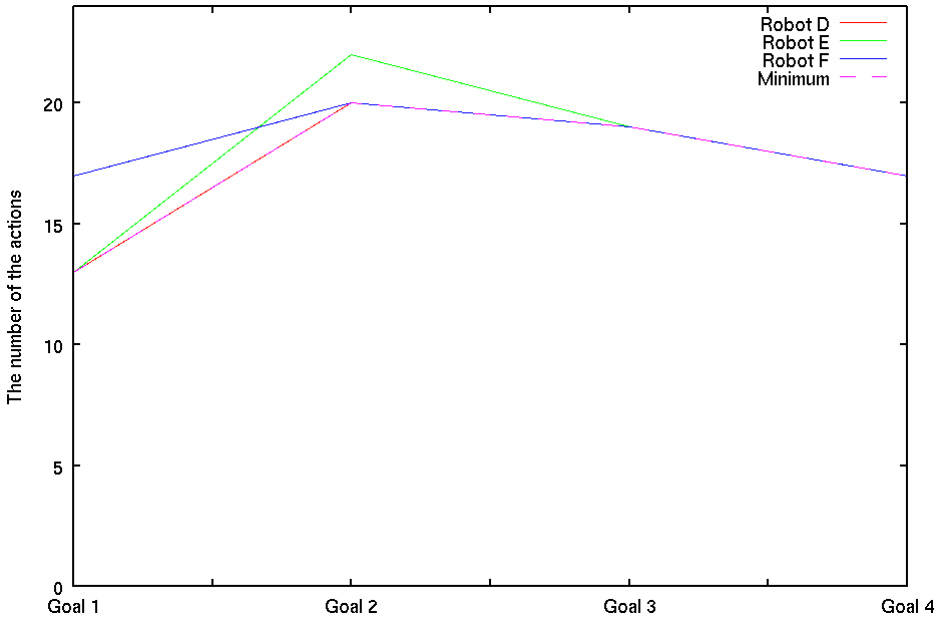


Fig.4.25 :Pursuit を用いた場合の各試行ごとの最短行動数の比較

（2）迷路サイズ 32×32

同様に行動選択部の手法に ε -greedy を用いた実験結果を Fig.4.26～Fig.4.28 に, Pursuit を用いた実験結果を Fig4.29～Fig4.31 に記す.

Fig.4.26・Fig.4.29 ではグラフを見やすくするため Y 軸の上限を 10000 にしてグラフへ出力している.

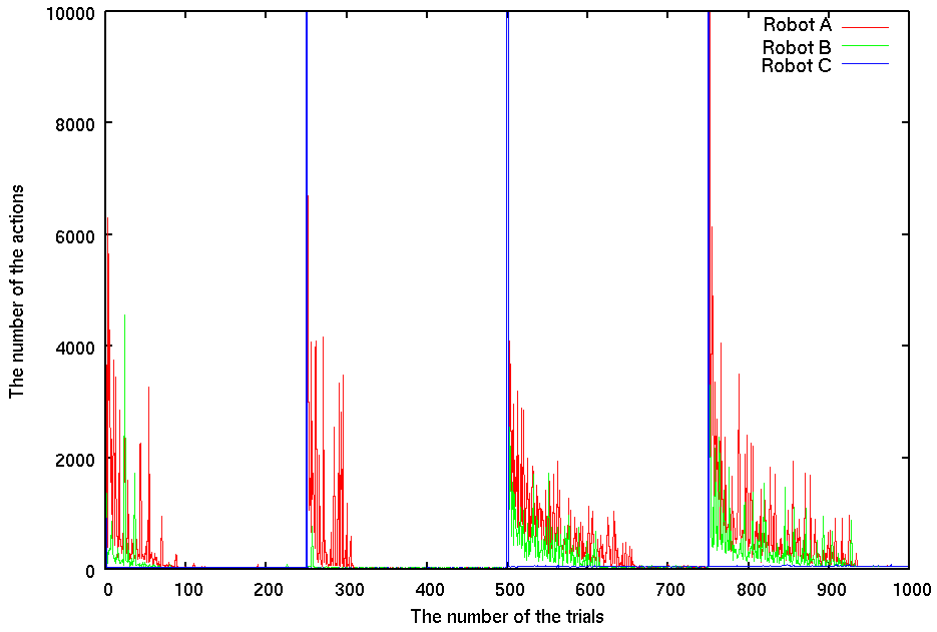


Fig.4.26 : ε -greedy を用いた場合の各試行における行動数の比較

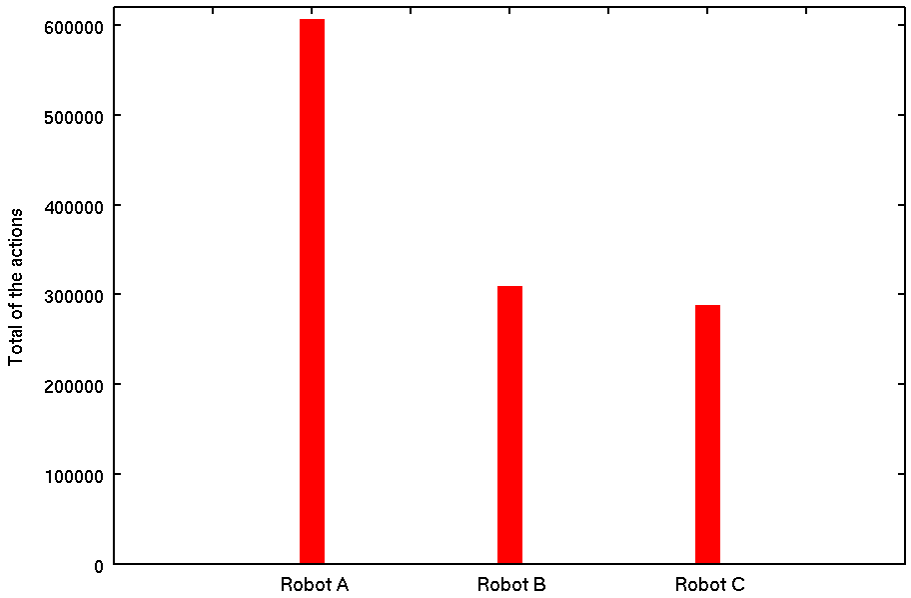


Fig.4.27: ε -greedy を用いた場合の 1000 回試行での行動数の総和の比較

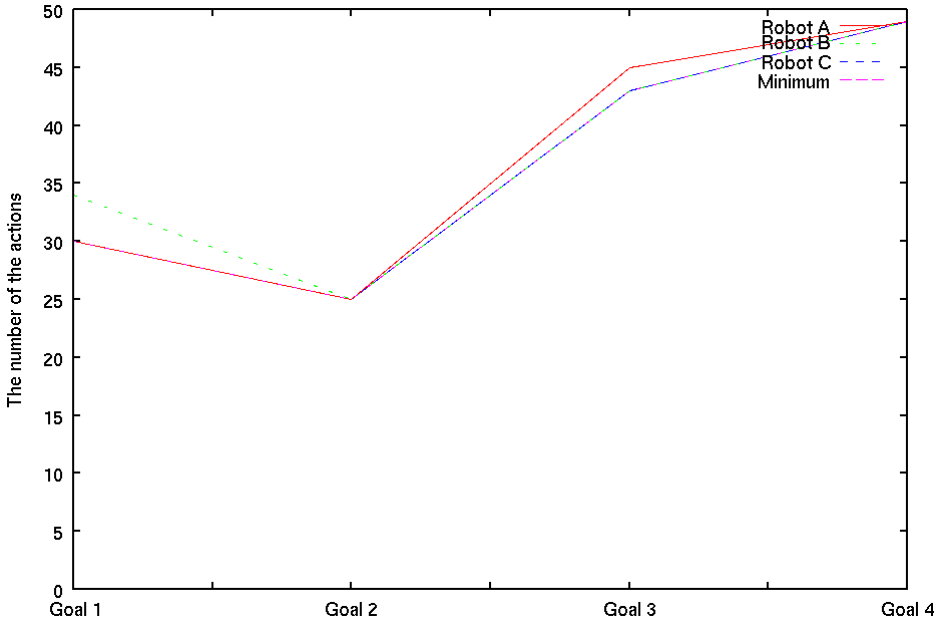


Fig.4.28： ε -greedy を用いた場合の各試行ごとの最短行動数の比較

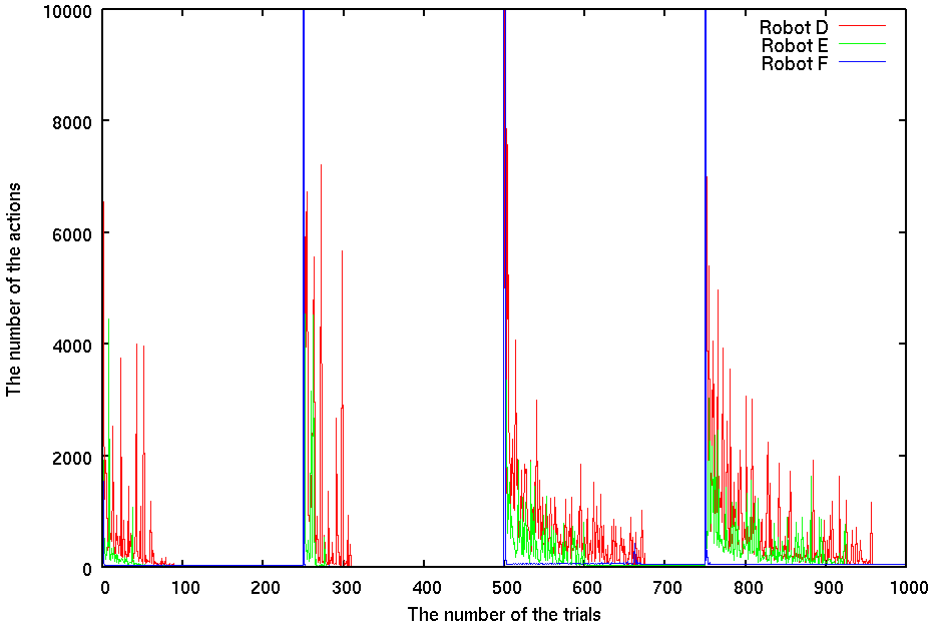


Fig.4.29：Pursuit を用いた場合の各試行における行動数の比較

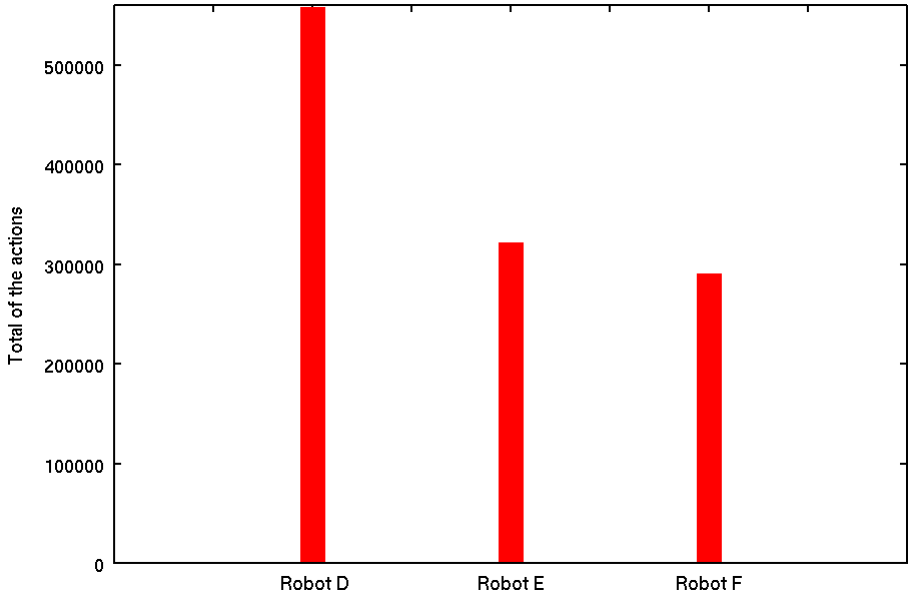


Fig.4.30：Pursuit を用いた場合の 1000 回試行での行動数の総和の比較

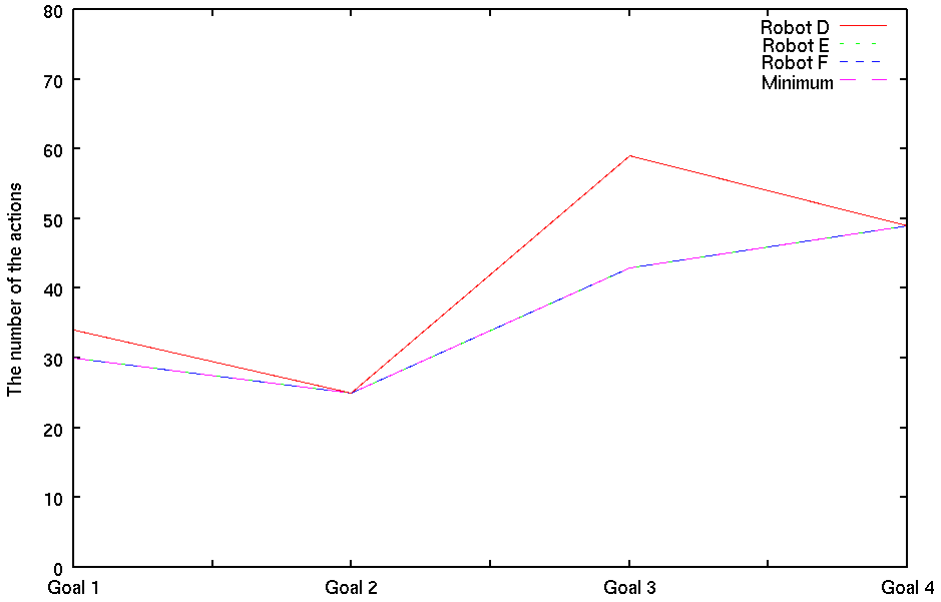


Fig.4.31 :Pursuit を用いた場合の各試行ごとの最短行動数の比較

（3）迷路サイズ 64×64

同様に行動選択部の手法に ε -greedy を用いた実験結果を Fig.4.32～Fig.4.34 に, Pursuit を用いた実験結果を Fig4.35～Fig4.37 に記す.

Fig.4.32・Fig.4.35 ではグラフを見やすくするため Y 軸の上限を 20000 にしてグラフへ出力している.

Fig.4.34 では強化学習の特徴が顕著に現れたのでこの結果について考察を行う. Fig.4.34 に注目すると, Robot A はゴール 2 についてはゴール 1 と比べるとかなり最短距離に近いルートを見つけられているのが見て取れる. この理由は Fig.4.19(c)を見るとわかる. ゴール 1 とゴール 2 の関係に注目すると. スタートからの方向がほぼ一致している. またゴール 1 のほうが遠くにあるためゴール 1 の学習結果を完全にリセットする必要が無い. つまり途中までのルートの学習結果はゴール 2 について学習する際に使える情報となっていた. そのため強化学習のみでも比較的良いルートを見つけることができた. 一方で Fig.4.34 のゴール 2 とゴール 3 についての部分に注目すると, 今度は Robot A と他のロボットとの差が広がっているのが見て取れる. この理由は先ほどのとは逆で, ゴール 2 とゴール 3 のスタートからの方向が大きく異なっているからである. ゴール 3 にゴールが移動した時, 強化学習ではゴール 2 へのルートの周辺から探索をし直す. その結果なかなかゴール 3 への学習が進まなくなってしまったのである. またゴール 4 について注目すると Robot A と他のロボットとの差はさらに広がってしまっている. ゴール 3 とゴール 4 についても同様のことが言える. このように強化学習は前の目的の学習結果の影響を受けることが多い. 特に今回のように環境が大きくなるとその影響は顕著に現れる. そこで報酬非依存型知識を用いることで別の目的に変わったときに新しい目的に対してスムーズに学習を行えるという

ことが実験の結果として現れた.

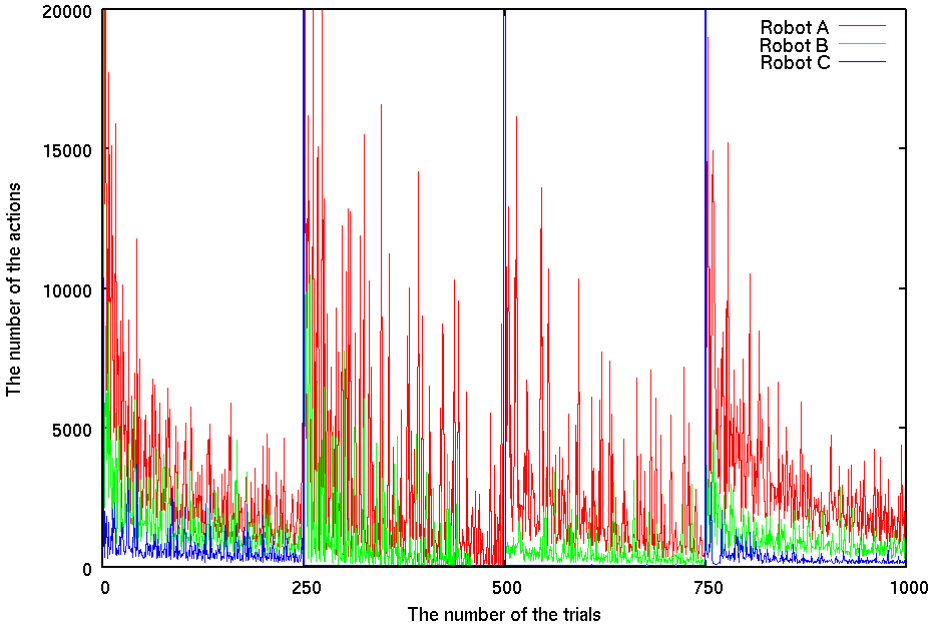


Fig.4.32 : ε -greedy を用いた場合の各試行における行動数の比較

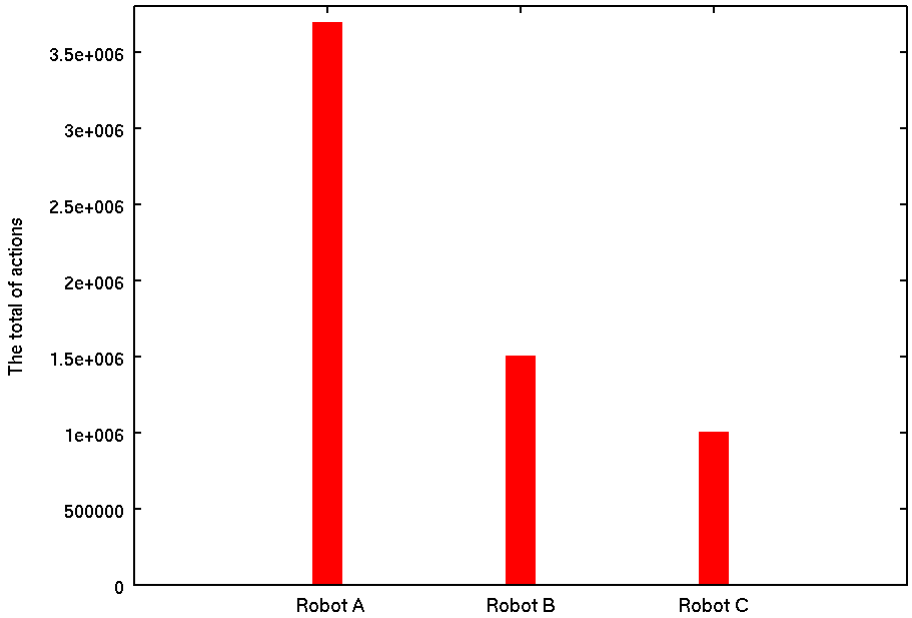


Fig.4.33: ε -greedy を用いた場合の 1000 回試行での行動数の総和の比較

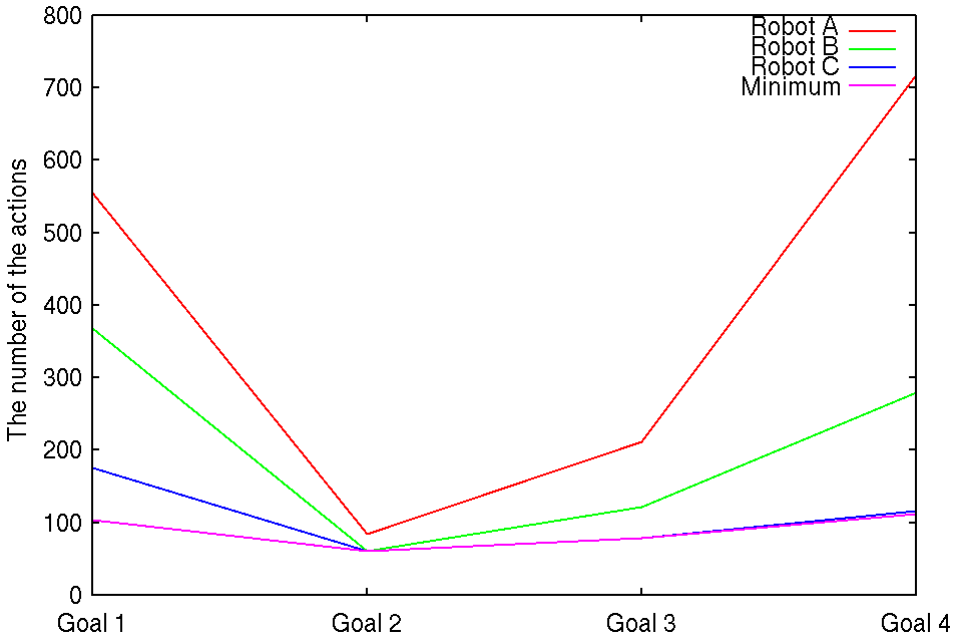


Fig.4.34： ε -greedy を用いた場合の各試行ごとの最短行動数の比較

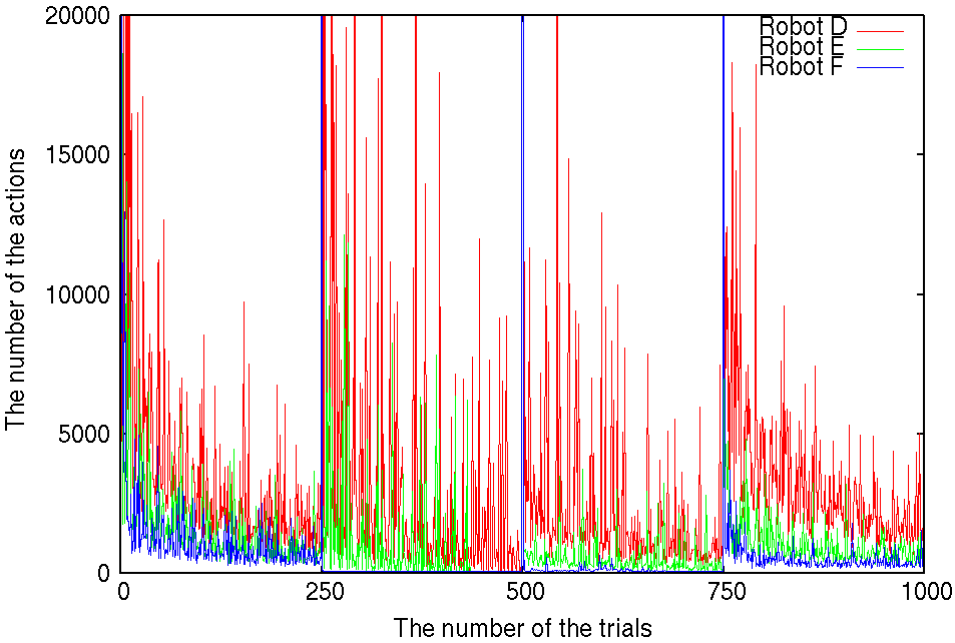


Fig.4.35：Pursuit を用いた場合の各試行における行動数の比較

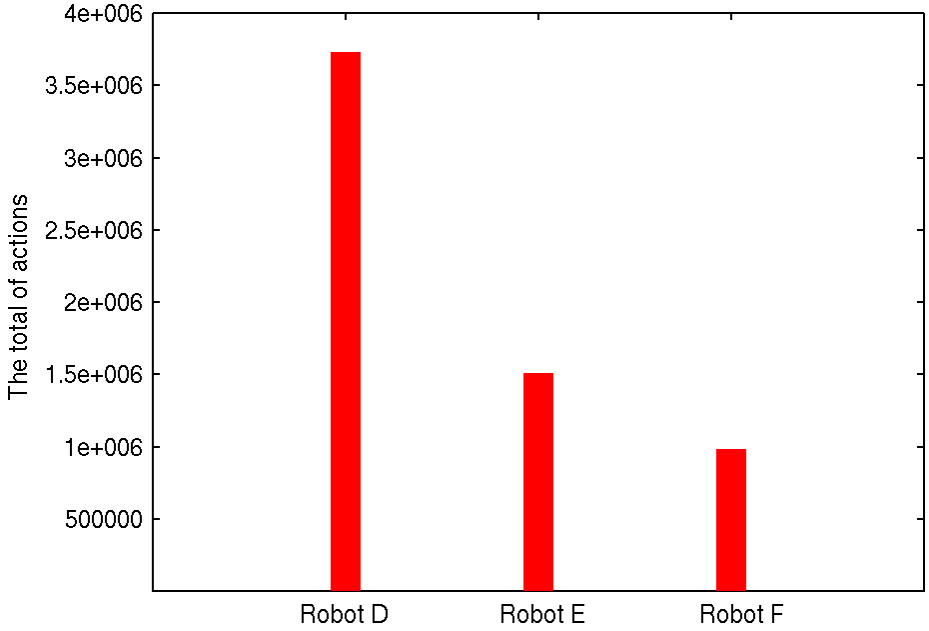


Fig.4.36：Pursuit を用いた場合の 1000 回試行での行動数の総和の比較

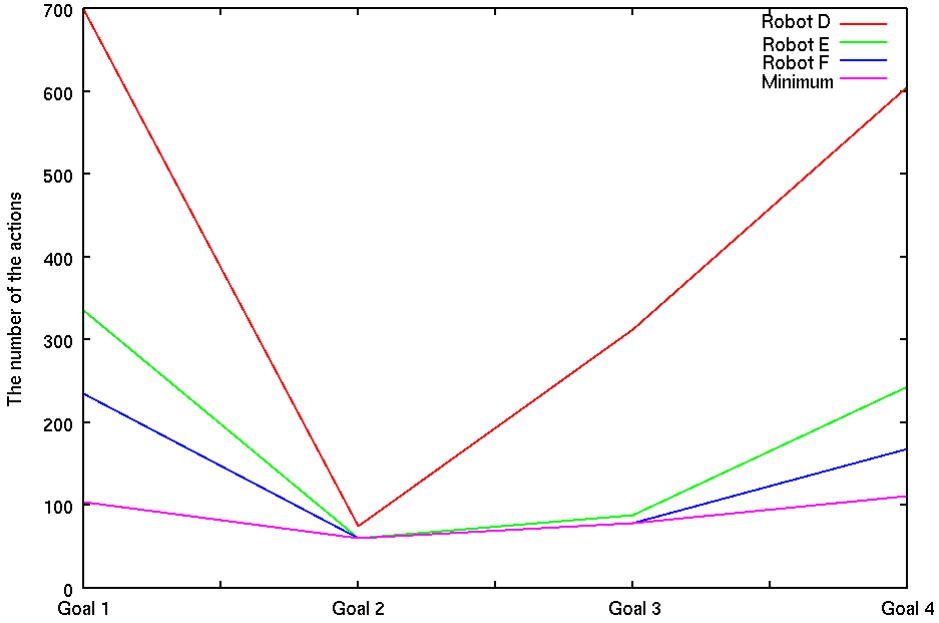


Fig.4.37 :Pursuit を用いた場合の各試行ごとの最短行動数の比較

E) 実験結果の考察

実験結果としては実験１とほぼ同等の結果が得られた．この実験２ではゴールまでの最短距離についての結果も示した．この最短距離についての結果は先ほど述べたように強化学習の特徴を良く表した結果となった．これについては迷路のサイズが小さい場合はどのロボットもほぼ代わりが無いが，迷路のサイズが大きい場合には報酬非依存型知識を持ったロボットのほうがより良いルートを発見できていた．強化学習の問題となっていた部分を報酬非依存型知識が補っているということがわかる結果となった．また，迷路問題において報酬非依存型知識が多い場合には迷路のサイズが大きくなっても最短距離を見つける可能性がかなり高いということがいえる結果になった．実社会ではより早く学習するということが求められているため，このように限られた試行の中でより最適なルートを見つけ出せたことは実社会においては重要なことである．

4.2.3 まとめ

4.2 節では静的環境下での実験を行った．迷路構造・迷路サイズ・行動選択部の手法の違いからそれぞれ実験を行った．

基本的には迷路構造や迷路サイズ，行動選択部の手法の違いによる実験結果の傾向は同じであった．報酬非依存型知識を有するロボットはスタートからゴールまでのルートを学習する試行回数は強化学習のみと比べて少なかった．また，学習したルートもより最短ルートに近いルートを学習していることもわかった．これらの結果は迷路のサイズが大きくなるほど顕著に現れていた．つまり，環境が大きくなるとそこで学習するためには膨大な時間がかかるが，報酬非依存型知識を用いれば難なく目的を達成できるということになる．

以上の実験の結果より，報酬非依存型知識は強化学習と共に用いることで強化学習の問題点を改善することができるといえる．

4.3 動的環境下での実験

本節では動的環境下での実験（実験３）について詳しく述べる．実験３では報酬非依存型知識を用いることで環境の変化に対応できることを示す．本実験では迷路サイズが「16×16」の環境についてのみ実験を行う．また，4.2 節の実験１と実験２より行動選択手法による影響は無いと考え，ε-greedy のみでの実験とした．

4.3.1 実験 3

本実験は報酬非依存型知識を用いることで環境変化を認識し，素早く対応できるということを示すために行う．そのため本実験ではゴールが変化する設定は無い．

B)　環境（迷路構造）

実験３では実験 2 で用いる迷路構造と同じものを用いる．実験３の重要な設定として環境変化が存在する．本実験では環境変化は一定試行ごとに行われるようにした．また，迷路のマスに障害物を置くことで環境変化させることとする．障害物が置かれたマスにはロボットは行くことができなくなる．つまり，迷路内に障害物が存在し，一定試行ごとに障害物の位置が変わることで動的環境とした．その例を Fig.4.38 に示す．

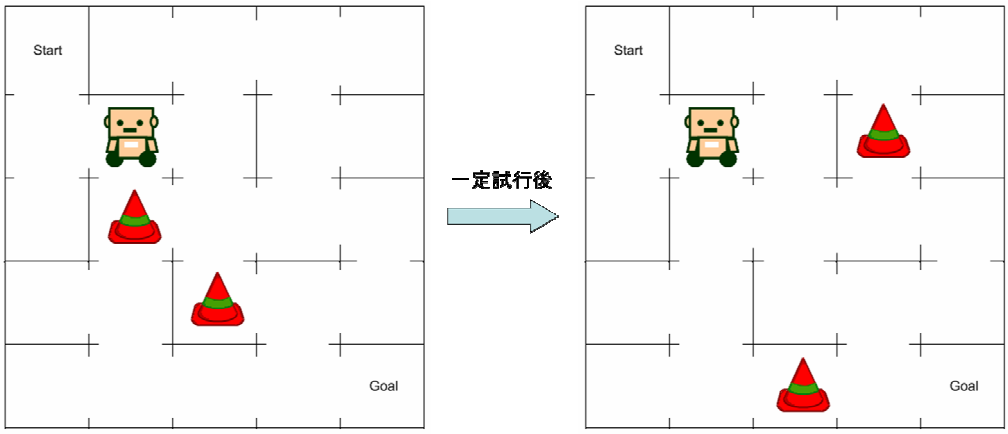


Fig.4.38：本実験における環境変化の例

また，実験３では迷路のサイズとして「16×16」の迷路のみで実験を行った．Fig.4.39 が実際に使用した迷路である．色の付いているマスがゴールとなっている．

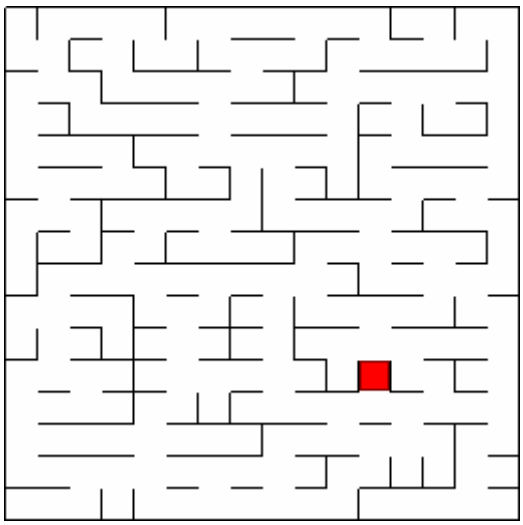


Fig.4.39：実験３で使用した迷路

B)　実験パラメータ設定

実験 3 で用いたパラメータ設定を表 4.9 に記す. 本実験ではゴールが変化しない設定となっている．また，環境が変化するまでの試行回数を 30 回とし，迷路内に存在する障害物の数を 51 とした．障害物の数は迷路の全マスの約 20%としている．

表 4.9 :設定

迷路サイズ	16×16
報酬（ゴールのみ）	100
実験終了までの試行数	100
ゴールが変化するまでの試行数	変化無し
環境が変化するまでの試行数	30
同時に存在する障害物の数	51
Q 値の初期値	0.001
ε（ε -greedy)	0.05
α	0.5
γ	0.5

C)　実験結果

以下に実験結果を示す．まずは行動選択部の手法に ε -greedy を用いた実験結果を Fig.4.40・Fig.4.41 に記す．比較している対象は強化学習のみを用いた Robot A，強化学習に加えて報酬非依存型知識を有する Robot B（f=0.5）と Robot C（f=1.0）である．

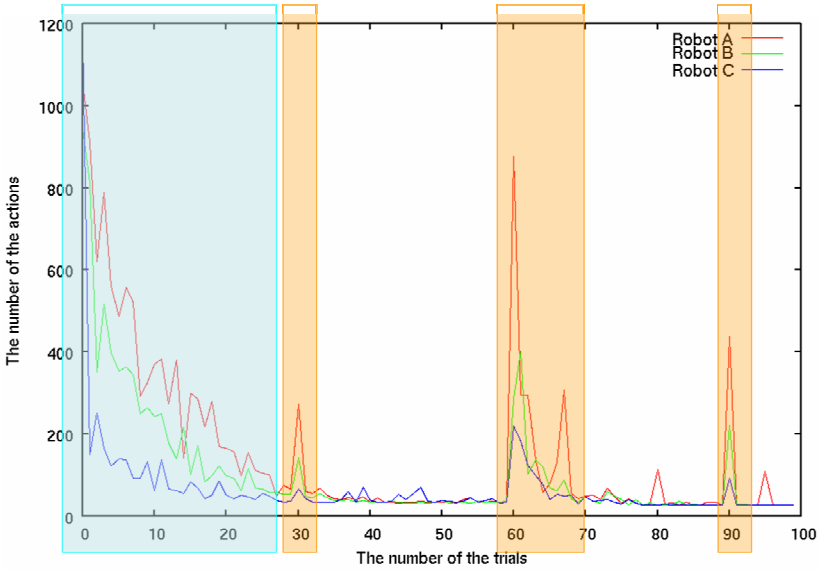


Fig.4.40： 各試行における行動数の比較

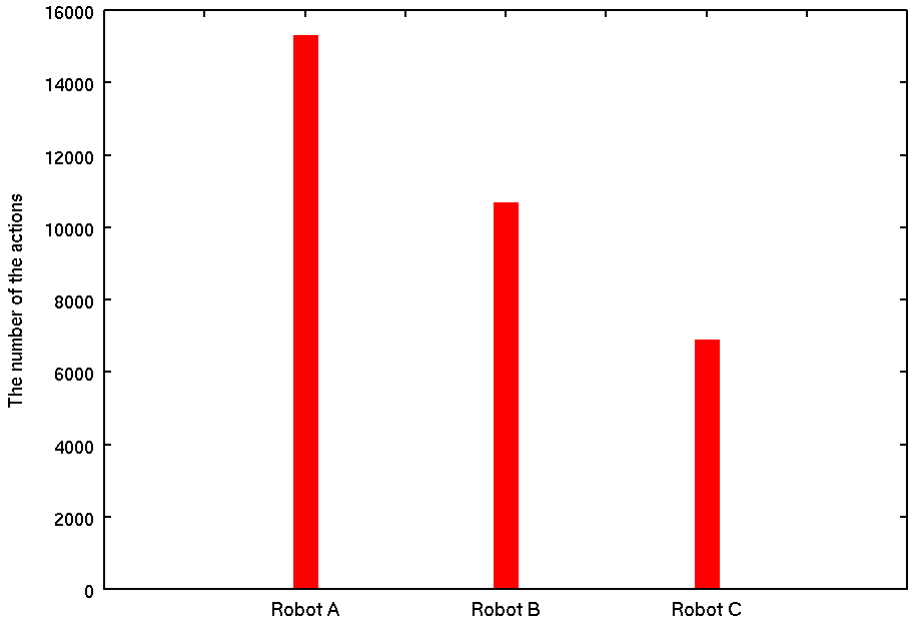


Fig.4.41：100 回試行での行動数の総和の比較

D)　実験結果の考察

Fig.4.40 において試行回数が 30 未満の前半部分（水色で囲まれた部分）ではゴールに対しての学習が行われている．この部分については実験 1・実験 2 の結果と同じ結果となった．報酬非依存型知識を用いたロボットと強化学習のみのロボットで収束する早さに差が出ているのがわかる．これは，今回は迷路のサイズが大きくないので Robot B や Robot C は早い段階で報酬非依存型知識が十分な量を保持できたからである．

注目するべき点は試行回数が 30 回目・60 回目・90 回目付近（オレンジで囲まれた部分）である．この部分では環境変化が起きているため，それまでに使っていたルートが使えなくなっている可能性がある．実際にグラフを見てみると，どのロボットも行動数が一時的に上昇しているのがわかる．30 回目・90 回目付近では比較的小さな影響であるが，これはそれまでに使っていたルート上に障害物が少しだけ現れたと思われる．それに対して試行回数 60 回目付近ではかなりの影響を受けていると見て取れる．これは障害物が良く使うルート上に多く配置されたためである．環境変化によってどのロボットも影響を受けてはいるが，報酬非依存型知識を利用したロボットは利用しないロボットと比べて素早く対応できているというのがわかる．

この実験結果から基本的な環境構造が変化せずに環境の一部が変化するような場合に対しては報酬非依存型知識が利用できることがわかった．これは実社会を考えた場合重要なことである．ロボットが動作する環境というのは今回の実験での設定のように環境の一部が変化するような場合が多いと考えられる．例としてロボットが動作する環境を家の中とすればある程度の家具の配置は決まっている．このような場合に報酬非依存型知識を用い

ることで環境の変化に対して素早く対応できるといった結果となった.

ただし，今回の実験では 1 つの環境で行っているのでもっと大きな環境などでの実験を行う必要があると思われる.

4.3.2 まとめ

4.3 節では動的環境下での実験を行った．この実験によって報酬非依存型知識が環境の一部が変化するような環境に対しても有効であることがわかった.

第 5 章 結論

5.1 まとめ

本論文では強化学習において，目的が変わった場合に素早く対応できるシステムの構築を目的とした．強化学習が報酬による学習であることに注目し，報酬による学習から起こる問題点について取り上げた．このような問題点を解決するために報酬とは別の情報を利用して学習するアプローチを取った．本論文では報酬とは別の情報としてロボットが認識する「状態」とロボット自らが取る「行動」による「状態遷移」に注目した．この遷移情報一つ一つを報酬非依存型知識と定義し，各報酬非依存型知識を独立の情報として保持する知識テーブルも定義した．報酬非依存型知識を獲得することで環境に対する学習を行うことで報酬に依存しない学習とした．この報酬非依存型知識は環境が変わらない場合，目的が変わったとしても正しい情報として扱うことができるものであった．そこで本論文では目的が変わった際に新たな目的に対して蓄えた報酬非依存型知識を利用していくことを考えた．今回はロボットが強化学習において学習を行っていく上で，報酬非依存型知識を利用することで目的に対してどのように学習していけばよいかを示した．つまり，目的に対してどのように行動していけば良いか予測し，その予測した行動を取りやすくなるように強化学習の価値関数に対してバイアスをかけた．それによって目的が変わった場合にでも素早く対応することができるようにした．このような考えの下システムを構築し，実験を行った．実験は報酬非依存型知識を有するロボットと強化学習のみのロボットで比較を行った．今回対象とした環境は静的環境を中心に動的環境についても実験を行った．どの環境に対しても報酬非依存型知識を用いることで目的が変わったとしても素早く対応できることを示すことができた．また，動的環境下での実験では環境変化に対応できるかどうかを検証した．報酬非依存型知識は環境についての情報であるので，環境変化にも利用できるのではないかと考えたからである．予想したとおり環境の一部が変わるような環境においては素早く対応ができているといえるような結果を獲た．これらの結果を得ることで提案したシステムの有効性を示すことができた．

5.2 今後の課題

5.2.1 動的環境下における更なる検証

本論文では動的環境下における実験は静的環境下における実験と比べると少ないものとなった．また，動的環境についてもっと複雑な環境に対しても検証を行ってみる必要がある．こういった理由から動的環境下における検証をもっと行う必要があると考える．

5.2.2　実ロボットへの適用

強化学習は実ロボットに用いられることが多い手法である．ゆえに本論文で提案したシステムを実ロボットへ適用したいと考えている．しかし実機で用いる場合に存在する問題点もある[11]．提案システムを実機で用いる場合に考えられる問題点としては以下のものが挙げられる．

・ ロボットのセンサ能力

本論文ではシミュレーション実験を行ったためロボットについてのセンサ能力については特に触れることは無かった．そのため今回定義した報酬非依存型知識は状態遷移を確定的な情報として定義した．しかしロボットの認識能力によっては本来は全く別の状態なのに同じ状態と認識してしまう可能性がある．例えば迷路問題ならば，ロボットが壁の有無しか認識できない場合同じ状態と認識しても別の状態遷移が起きる可能性が高い．Fig.5.1 を例とすると状態 S4 と S8 は同じ状態と認識してしまうということである．こういった場合は確定的な遷移情報は使えないと考える．よって報酬非依存型知識の定義を見直す必要がある．

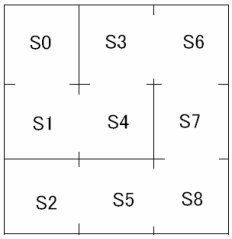


Fig.5.1：迷路の例

・ ロボットのメモリ能力

本論文では報酬非依存型知識を保持できる量については特に制限を持たせなかった．しかし，実ロボットを用いる場合にはメモリの関係上そうはいかない．よって知識テーブルの大きさやどの情報を捨てるのかなどについても考える必要がある．

・ 実ロボットの行動に要する時間とこのシステムで計算を行う際の時間との差異

本論文で提案したシステムでは知識テーブルの中から目的の状態までどのように遷移していけばよいか検索している．これは報酬非依存型知識の量が増えるほど時間がかかるものとなっている．そのためシミュレーション実験にかかる時間が強化学習のみのシステムと比べて多少増えてしまう．この時間がロボットが実際に行動を取る時間を大幅に超えていた場合は強化学習のみのシステムに時間的遅れを取ってしまう．考えて行動するのが早いかとりあえず行動するのが早いかの問題となっている．

参考文献

[1] M. Hirose and K. Ogawa, "Honda humanoid robots development", Phil. Trans. R.Soc. A, Vol.364, No.1850, pp.11-19, 2007

[2] Thomas Mitchell, "Machine Learning" ,McGraw-Hill Science/Engineering/Math, 1997

[3] Zbigniew Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer, 1992

[4] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning", The MIT Press, 1998

[5] 森川幸人，“マッチ箱の脳(AI) － 使える人工知能のお話”，新紀元社，2000

[6] 伊藤一之，高山明宏，“身体と環境の特性を利用した状態-行動空間の抽象化 –強化学習を用いた自立ヘビ型ロボットへの適用-”，知能と情報（日本知能情報ファジィ学会誌），Vol.21，No.3，pp.402-410，2009

[7] 薮内佳孝，加藤昇平，犬塚信博，“強化学習に基づいたルーティングアルゴリズムの一手法”，参考 URL http://www-kasm.nii.ac.jp/jsai2004_schedule/pdf/000150.pdf

[8] Daniel W. Ttroock, "An Introduction to Markov Process", Springer, 2005

[9] 山村忠義，馬野元秀，瀬田和久，“段階的な視覚をもつエージェントにおける強化学習について-追跡問題を例にして-”，日本ロボット学会誌，Vol.18，No.5，pp.561-570，2006

[10] 井谷優, “マイクロマウスの歩んだ路”, 日本ロボット学会誌, Vol.27, No.9, pp.979-982, 2009

[11] 浅田稔, “強化学習の実ロボットへの応報とその課題”, 人工知能学会誌, Vol.12, No.6, pp.831-836，1997

謝辞

本論文を結ぶにあたり，日ごろより懇切なるご指導を賜りました倉重健太郎先生に深く感謝の意を表します．また，ご助言，ご指導をいただいた畑中雅彦先生，渡辺修先生，渡邉真也先生に感謝の意を表します．そして，論文の査読や助言をしていただいた認知ロボティクス研究室の池田憲弘さん，木島康隆さん，中南義典君に感謝いたします．