遺伝的プログラミングと強化学習の統合に基づく実ロボットの行動獲得

神尾 正太郎† 三橋 秀行† 伊庭 斉志†

本論文では，遺伝的プログラミングと強化学習を統合した，実ロボットでの実時間学習が可能な手法を提案する．提案手法では，高精度のシミュレータは必要としない．なぜなら学習はシミュレータ上だけではなく，実ロボット上でも行われるからである．このため実ロボットでの最適行動を得ることができる．本論文では提案手法に基づき実ロボット AIBO で行った実験結果を示し，提案手法が従来の Q 学習手法よりも性能が良いことを示す．

Integration of Genetic Programming and Reinforcement Learning for Real Robots

Shotaro Kamio,† Hideyuki Mitsuhashi† and Hitoshi Iba†

We propose an integrated technique of genetic programming (GP) and reinforcement learning (RL) that allows a real robot to execute real-time learning. Our technique does not need a precise simulator because learning is done with a real robot. Moreover, our technique makes it possible to learn optimal actions in real robots. We show the result of an experiment with a real robot AIBO and represents the result which proves proposed technique performs better than traditional Q-learning method.

1. はじめに

自律的なロボットにタスクをさせる際には，ロボット自身に行動させ環境との相互作用の中から学習を行うことが考えられる．ロボットの行動を作りこみではなく自動的に生成する手法として遺伝的プログラミングや強化学習がある．

遺伝的プログラミング（Genetic Programming, GP）[3),8)] では，多数の個体（プログラム）を何世代も繰り返し評価を行う必要がある．そのため，1 個体の評価に時間のかかる問題に適用するのは困難であり，実ロボット上で学習を行う研究はほとんどない．

強化学習（Reinforcement Learning，RL）[6),10)] で最適な行動を獲得するためには何度も試行を繰り返す必要があり，実ロボットで複雑なタスクを学習させるには膨大な時間がかかる．実ロボットへの適用としては，文献 9) のように行動ごとに即時報酬が得られるもの，もしくは文献 12) のようにシミュレーションで学習させた結果を実ロボットに載せるという研究などがある．また，強化学習で得られるのは刺激反応的な行動であり，より複雑な行動を生成させるのは難しい．

このように実ロボットを動かす行動を獲得するためにシミュレータでの学習が一般的に行われており，その場合には正確なシミュレータが必要である．しかし正確なシミュレータを作成困難なタスクも多く存在する．正確でないものを用いると，シミュレータ上では最適に行動するが実ロボットでは最適でない行動を学習することになりうる．また実ロボットの動作特性は製作時の微妙な誤差や経年変化により，その動作特性には 1 台 1 台ばらつきが出る．これらの問題にシミュレータで対応するのは困難である．

本論文では以上の問題を解決するための，遺伝的プログラミングと強化学習を統合した，実ロボットでの実時間学習が可能な手法を提案する．本手法では，シミュレータだけでなく実ロボットでも学習するのでタスクの本質を表現したシミュレータがあればよい．それには単純なシミュレータで十分であり，シミュレータを作成するコストは格段に減らすことができる．さらにシミュレータではなく実ロボットで学習することで，シミュレータでは再現されないハードウェアや環境の特性をも学習し，思いがけない行動を獲得できる

† 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Science, The University of Tokyo

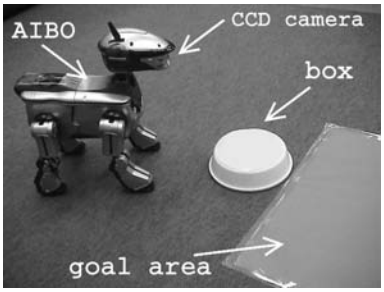


図 1 実験に用いたロボット AIBO．箱とゴール領域
Fig. 1 The robot AIBO. The box and the goal area.

可能性が生まれる．遺伝的プログラミングではプログラムを生成できるため，強化学習のように刺激反応的な行動だけでなく，複雑な行動を生成できる可能性がある．また，シミュレータで獲得された行動をもとに実ロボットで学習するので，強化学習のみで行うよりも高速に学習することができる．

次章では本研究で行った実験の問題設定を説明する．3 章では提案手法の説明を行い，4 章では実ロボットを用いた実験結果を示す．5 章では従来手法との比較と今後の展望を述べる．最後に結論を述べる．

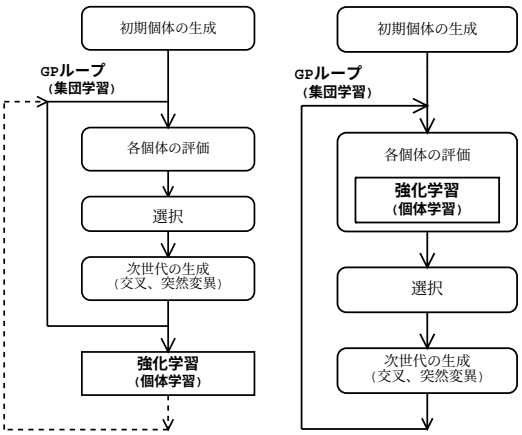
2. 問題設定

実験で用いる実ロボットとして，SONY より販売されているエンターテイメントロボット，AIBO ERS-220（**図 1**）を利用した．AIBO は開発環境が公開されており，C++言語を用いてプログラミングを行うことができる[4]．AIBO では CCD による画像入力を利用できる．しかも画像認識のためのハードウェアを備えているため，指定した色の物体だけを画像中から切り出すことが高速かつ容易にできる．

本実験で設定したタスクは，荷物に見立てた箱をゴール領域まで押して運ぶというものである．この問題の難しい点はこのロボットが 4 本足であることで，ロボットが前進する場合であっても，箱と足の位置関係によっては箱が前に進んだり左右にそれたりするという異なった振舞いとなることである．そのため，箱の振舞いを正確に表現するシミュレータを作成するのは非常に困難である．

3. 提案手法

本研究では，遺伝的プログラミングと強化学習の統合手法を提案する．本提案手法は**図 2** (a) のように，個体学習である強化学習が GP ループの外側にある．この手法により，(1) 実ロボットでの学習の高速化，(2) シミュレータで獲得された行動を微調整し実ロボットの環境に適応させること，が可能となる．



(a) Proposed technique of the integration of GP and RL.
(b) Traditional method combined GP and RL [1),2)].

図 2 アルゴリズムの概略
Fig. 2 The flow of the algorithm.

本手法は 2 段階から成っている．

(1) 簡略化されたシミュレータ上で遺伝的プログラミングを行い，タスクに必要な行動の判断基準を持つプログラムを作り上げる．
(2) (1) により得られたプログラムを実ロボットに載せ個体学習（強化学習）を行う．

第 1 段階では遺伝的プログラミングを用いて，実ロボットがタスクを達成するために必要な行動判断を持ったプログラムを作り出す．第 2 段階では強化学習を行うが，第 1 段階で得られた判断基準により状態空間が部分空間に分割されるので学習が高速化されることが期待できる．また，シミュレータ上で学習しているために，ロボットは第 2 段階の開始時からタスクに応じた行動をとることが期待できる．図 2 (a) の外側の点線のループは，本研究では実現されていないが，理想的には個体学習で獲得された実環境のパラメータなどをこのループでフィードバックさせるべきであると考えている．

従来手法[1),2)] との比較については 5.2 節で述べる．

3.1 実ロボットでの強化学習の設定

本研究では強化学習として Q 学習を用いた．Q 学習を適用するために必要な行動，状態空間を以下のように設定した．

3.1.1 行　　動

ロボットが選択できる行動として前進，後退，左回転，右回転，後退+左回転，後退+右回転の 6 種類の行動を用意した．これらの行動は理想的な行動とは異なっている．たとえば，“前進” はロボットがまっすぐ

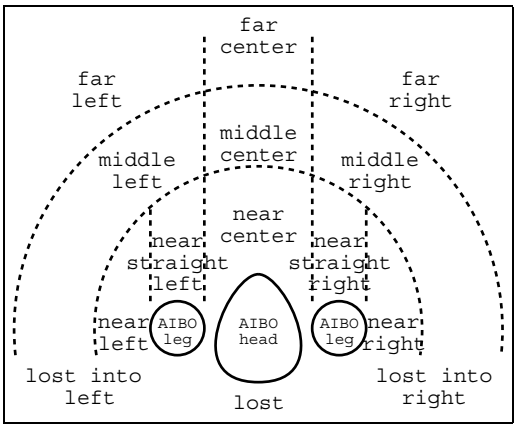


図 3 状態の分割（箱の場合．上がロボットの前方向である）
Fig. 3 States in real robot for the box. The front of the robot is upside of this figure.

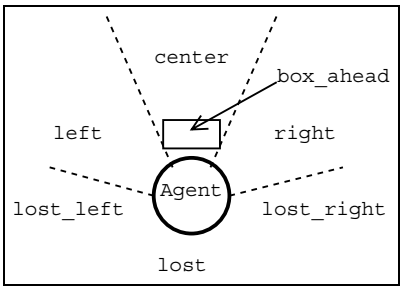


図 4 箱，ゴール領域に対するシミュレータ上での状態区分（box_ahead はif_box_ahead ノードが第 1 引数を実行する領域を示す）
Fig. 4 States for box and goal area in the simulator. The area box_ahead is not the state but the place where if_box_ahead executes first argument.

前に進むだけではなく左右へのぶれもいくらか生じ，“左回転” では小回りがきかないので，単に左回転するだけではなく，わずかに前進してしまう．ロボットはこのような行動の特性を学習しなければならない．

各行動には 4 秒程度かかり，以下で述べる首振り行動も含めると 8 秒程度かかるので，学習時間が短ければ短いほど望ましい．

3.1.2 状態空間

状態空間は文献 12) と同様に CCD から得られる画像上での箱，ゴール領域の位置を基に構成した．ただし AIBO の CCD の視野は非常に狭く，1 方向の画像だけでは箱またはゴール領域が見えない状態が大部分となってしまう．これを避けるために首振りを行って周囲の画像を補う仕組みを与え，一行動ごとにこの首振りを行い状態認識を行うものとした．本研究ではこの首振りを学習要素としていないため，一行動ごとにつねに行われる．

箱についての状態を地面上に投影すると**図 3** のようになる．“near center” となっている位置は箱が両前足の間にちょうどはまり込む位置であり，この状態で前進すれば箱を移動させることができる．さらに，箱が “near center” の位置にあるときに回転を行っても，両前足の間にはまり込んでいるために箱は回転後も “near center” の位置に来る．このほかに箱が視界にない状態を “lost”，そして箱が視界にはないが 1 行動前には左側にあった場合を “lost into left”，1 行動前には右側にあった場合を “lost into right” とした．

AIBO は 4 本足であるため，足元の認識には注意が必要である．ロボットが前進する場合であっても，箱と足の位置関係によっては箱が前に進んだり左右にそれたりするという異なった振舞いとなるのである．これを表現する適切な状態を定義しなければ，強化学習が前提とする環境のマルコフ性を満たさなくなり最適な行動を見つけることができない．そこで，前足の前方領域の状態として “near straight left”，“near straight right” を定義した．

箱については以上で述べたように 14 状態である．ゴール領域の状態については “near straight left”，“near straight right” がそれぞれ “near left”，“near right” に含まれている以外は同じであり，12 状態である．環境の状態は箱の状態とゴール領域の状態の積で表現することができるので，この環境の全状態数は $14 \times 12 = 168$ である．

3.2 遺伝的プログラミングの設定

3.2.1 シミュレータ

シミュレータは一般的には，実環境や物理特性を細かく測定し，できるだけ現実に近づけることを目指すものである．しかし本手法ではタスクの本質的な情報が表現されている限り単純なシミュレータでよい．なぜなら学習はシミュレータ上で終わりではなく，実ロボット上でも行われるからである．

本研究で用いたシミュレータは，2 次元平面上に円で表現したロボット，箱，そして平面上に固定したゴール領域を用いている．この平面上でロボットが箱を押し，ゴール領域に重ねるとタスク完了である．

シミュレーション上のロボットのために，実ロボットの環境を単純化した行動と状態空間を定義した．定義した行動は，前進，左回転，右回転の 3 つである．これらの行動は実ロボットの場合と異なり理想的な行動である．状態空間は実ロボットで用いる状態空間を簡略化し，**図 4** のように定義した．

この状態分割は実ロボットに近いがまったく同じというわけではない．さらに箱の重さや摩擦など物理パ

ラメータの類はいっさい測定しておらず，ロボットの形状も考慮していない．このように単純なシミュレータであるため製作にかかるコストは少ない．

このシミュレータ上では以下のような箱の 2 つの移動特性が表現されている．

- ロボットが前進するときに箱がロボットの正面で接触するなら箱も前進する☆．
- ロボットが回転する際に箱がロボットの正面手前にあれば，回転後の箱もロボットの正面手前になる☆☆．

という動作特性である．

3.2.2 遺伝的プログラミングの設定

本研究で遺伝的プログラミング（GP）には以下のノードを使用した．

- 終端ノード：`move_forward`，`turn_left`，`turn_right`
- 関数ノード：`prog2`，`box_where`，`goal_where`，`if_box_ahead`

終端ノードはそれぞれシミュレータ内での前進，左回転，右回転に対応している．関数ノードの`box_where`，`goal_where` は 6 引数の関数で，ロボットから見た箱（`box`），ゴール領域（`goal`）の状態（図 4）に応じて引数の 1 つを実行するものである．`if_box_ahead` は条件判断で，箱がロボットの正面手前の位置（図 4）にある場合に第 1 引数を実行し，箱が正面手前にない場合に第 2 引数を実行する．遺伝子の先頭ノードには`box_where`，`goal_where` だけが来るようにした．遺伝子を実行する際には先頭ノードから実行され，終端ノードを実行し終えたときにはまた先頭ノードから実行を繰り返す．これによって得られるプログラムは明示的な記憶を持たないものであり，状態入力に応じて行動が決定される刺激反応型の構造となる．ただし`prog2` を用いることでプログラム中の文脈を利用した連続した行動を作り出すことも可能であり，刺激反応ではない行動を作り出す可能性がある．

1 試行はロボットと箱がランダムな初期位置に置かれた状態から始まり，タスクを達成するか一定回数の行動をとると試行は終了となる．この試行中の行動に対して以下のように適合度を割り当てた．

- タスクを達成した場合：

$$f_{goal} = 100$$

$$f_{\text{remaining}} = 10 \cdot \left(0.5 - \frac{\text{移動回数}}{\text{移動回数の上限値}}\right) + 10 \cdot \left(0.5 - \frac{\text{回転の回数}}{\text{回転回数の上限値}}\right)$$

- 少なくとも 1 回，箱を移動させた場合：

$$f_{\text{move}} = 10$$

- 少なくとも 1 回，箱の方向を向いた場合：

$$f_{\text{see\_box}} = 1$$

- 少なくとも 1 回，ゴールの方向を向いた場合：

$$f_{\text{see\_goal}} = 1$$

- つねに与えるペナルティ：

$$f_{\text{lost}} = -\frac{\text{箱を見失った回数}}{\text{行動回数}}$$

以上の和が i 回目の試行に対する適合度 $fitness_i$ となる．

初期位置に依存しないロバストな行動を獲得させるため，個体の適合度計算ではランダムに初期位置を変えた 100 試行での平均を用いた．個体の適合度は以下の式で計算される．

$$fitness = \frac{1}{100}\sum_{i=0}^{99} fitness_i + 2.0 \times \frac{(\text{遺伝子長の上限}) - (\text{遺伝子長})}{(\text{遺伝子長の上限})} \quad (1)$$

右辺第 2 項は，同じ行動をとる遺伝子であれば遺伝子が短いものほど大きな適合度とするための項である．係数の 2.0 は予備実験を行って定めた．

以上のような適合度のもとで最大遺伝子長を 150 とし，個体数 1000 で 50 世代の学習を実行した．この実行は Athlon XP 1800+の Linux システム上で 10 分程度である．

ここで得られた最も成績の良い個体を実ロボットでの学習に適用した．

3.3 遺伝的プログラミングと強化学習の統合

シミュレータ上で最適となった行動を実ロボットの動作特性に適応させるために強化学習を行う．これはシミュレータ上で`move_forward`，`turn_left`，`turn_right` となっていた行動を実ロボット上での最適行動となるよう適応させるのである．

実現方法としては，行動ノードである`move_forward`，`turn_left`，`turn_right` のそれぞれに対し，Q 値を表にした Q テーブルを割り当てる．すなわち全体では 3 つの Q テーブルを持つことになる．この Q テーブルで用いる状態は実ロボット上での状態である．GP 個体を実ロボットで実行する際には，`move_forward`，`turn_left`，`turn_right` のいずれかに達したときに，

☆ 実ロボットでの前進に相当．
☆☆ 実ロボットでの回転で両前足の間に箱がある場合に相当．

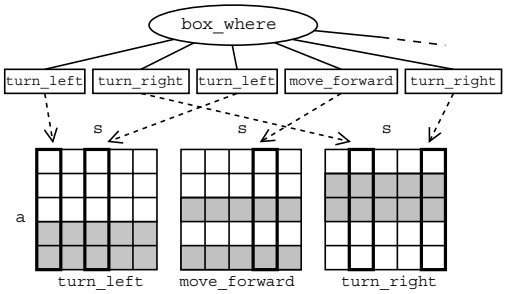


図 5 GP 個体から Q テーブルを参照し行動を選択する例
Fig. 5 Action nodes pick up a real action according to the Q-value of a real robot's state.

表 1 行動ノードとその選択可能な行動
Table 1 Action nodes and their selectable real actions.

行動ノード	Q テーブルが選択可能な行動
move_forward	前進 *，後退 + 左回転，後退 + 右回転
turn_left	左回転 *，後退 + 左回転，後退
turn_right	右回転 *，後退 + 右回転，後退

* 選択されやすいように初期化した行動

それぞれの Q テーブルを参照し行動を決定するのである．図示すると**図 5** のようになる．これにより，実ロボット上での状態が同じでも，どのテーブルを参照するかによって行動が変わることがある．実ロボットにしか存在しない “near straight left”，“near straight right” 状態はシミュレータでの “center” と対応づける．

本手法では，各 Q テーブルで選択可能な行動を制限する．これはたとえば`turn_left` ノードにおいて右回転を学習する必要はないという考えからである．本研究で用いた制限は**表 1** に示した．本手法では各 Q テーブルはバイアスをかけた初期化を行う☆．これにより，Q 学習の初期状態であっても GP で学習した結果を基に行動することが可能となる．本研究では`move_forward` の Q テーブルでは前進が選択されやすいようにし，`turn_left` では左回転が，`turn_right` では右回転が選択されやすいようにした（表 1）．このように特定の行動が選択されやすくするために，Q テーブルに初期値として 0.0001 を書き込んでおき，その他の行動の Q 値を 0.0 とした．

3 つの Q テーブルを合わせたサイズは通常の Q テーブルの 1.5 倍となる．理論的には通常の Q 学習よりも最適解への収束に時間がかかると考えられる．しかし本手法では GP で得られたプログラムを基に行動するのでテーブルの全状態を参照するわけではなく，しかも Q 学習開始直後でもタスクに応じた行動が可能であるので，実行時の性能は良いと考えられる．

ロボットで画像を基に状態空間を構成し Q 学習を実行する場合には，「状態と行動のずれ」問題[12] を考慮する必要がある．これは，画像から構成した状態空間では状態遷移にばらつきがあるために，最適行動が学習できないという問題である．この問題に対処するため “状態の変化” を再定義した☆☆．その定義は，プログラムで実行している終端ノードが同一でなおかつ実行している状態（実ロボットが認識した状態）が同じ場合を状態が変化していないとする．状態が変化しない間は同じ行動をとり続け，Q 値の更新も行わない．すなわち，プログラム中で実行する終端ノードが変わるか，同一の終端ノードであっても実ロボットの認識した状態が変化してしていれば，Q 値の更新を行う．

Q 学習のパラメータは学習率 $\alpha = 0.3$，割引率 $\gamma = 0.9$ とし，報酬はタスクを達成した状態で 1.0，それ以外の状態では 0.0 とした．

4. 実験結果

4.1 学習開始直後

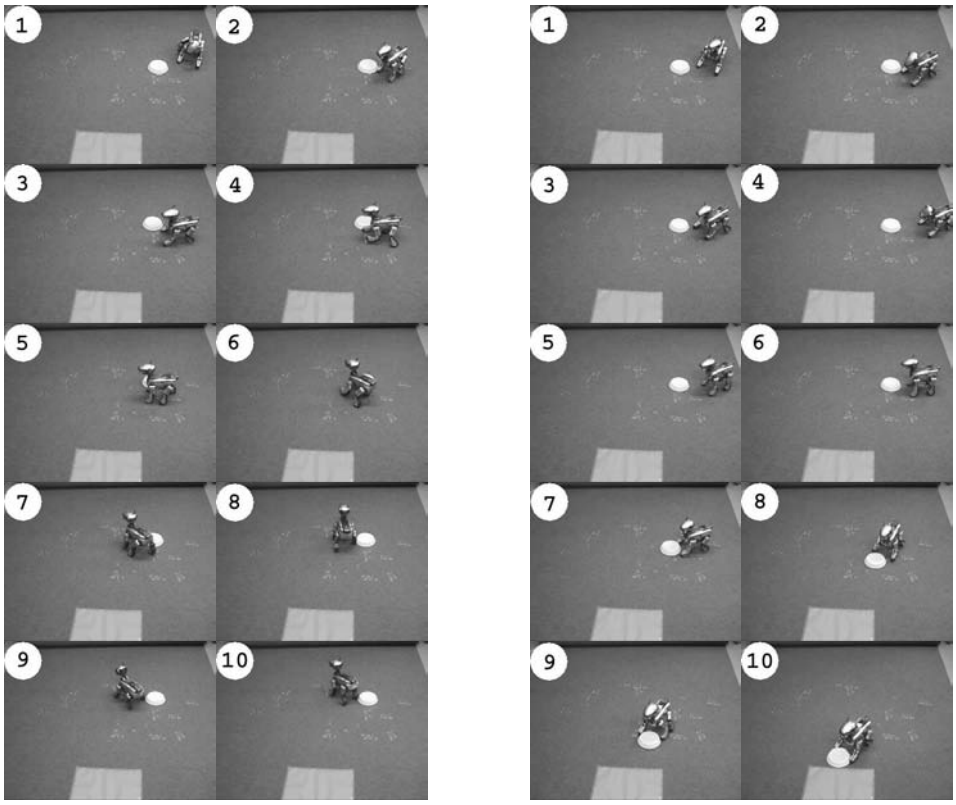
本手法を用いたことで，実ロボットでの Q 学習を開始した時点ですでに，大部分の配置においてタスクを達成することができた．これは GP を用いて学習した結果を利用して行動できるためである．

ロボットが移動して箱がロボットの正面手前に来る場合には，試した状況すべてでタスクを達成した．しかし移動しても箱が正面手前に来ない場合，たとえば箱が足のやや外側になってしまう場合などでは箱の移動を失敗することがあった．**図 6** (a) にその典型的な行動系列を示した．ロボットは箱に近付こうと右回転を繰り返すのだが，箱の周りを回ってしまいなかなか近付くことができない．これは実ロボットがシミュレータ上とは異なり小回りがきかないためである．箱に近付くことができず右回転を繰り返すあまり，箱を見失ってしまうこともあった（図 6 (a) の最終図）．

このことは，実ロボットとシミュレータの表現には差異が存在しているために，シミュレータ上での最適行動が実環境でも最適とは限らないという典型例である．この場合，足の位置関係や実ロボットで回転の小回りがきかないことまではシミュレータで再現されていないために GP では学習できていなかったのである．

☆ Q 学習では Q 値の初期値のとり方は任意であり，初期値によらず最適解に収束する[6]．

☆☆ 本提案手法では複数の Q テーブルを持つので，浅田らの定義[12] を変更して用いた．



(a) Failed actions losing the box at the beginning of learning.

(b) Successful actions after 10 hours learning.

図 6 行動系列の例
Fig. 6 Typical series of actions.

4.2 10 時間後（約 4000 行動後）

学習した結果，図 6 (b) のように最適な行動が見られた．学習開始直後では箱に近付くことができなかった配置において，ロボットは “後退” や “後退 + 回転” の行動をとるようになった．その結果，ロボットは箱に近付くことができ，ゴール領域まで箱を押し，タスクを達成した．

このほかの配置においても学習の効果が見られた．ロボットが箱にスムーズに近付くことができるようになったため，箱やゴール領域を見失う回数が減少したのである．これはロボットが学習初期よりも効率的に行動するようになったことを示している．

5. 考　　察

5.1 従来手法との比較

提案手法の有効性を確認するために，従来の Q 学習手法（シミュレータで学習し，実ロボット上で再学習を行う手法．以下，“RL+RL” と呼ぶ），そして文献 13) で提案されている政策のみを引き継ぐ手法（以下，“Policy Transfer” 手法と呼ぶ）との比較を行った．

Q 学習（RL+RL）と Policy Transfer 手法はシミュレータ上で Q 学習を行う必要があるため，3.2.1 項（図 4）で述べた状態空間をより細かくしたシミュレータを用いた．これはたとえば箱が “center” 状態であっても，箱がロボットの目前にある場合もあれば遠く離れている場合もあり，その区別がつかないことには Q 学習で最適行動を得ることができないためである（GP では状態のほかに `if_box_ahead` ノードを用いることで区別が可能である）．そのため，箱とゴール領域それぞれの “left”，“center”，“right” の 3 つの状態に実ロボットと同様の距離（“far”，“middle”，“near”）を導入することで詳細な状態空間（$12 \times 12 = 144$ 状態）を構成した．

RL+RL 手法では，シミュレータで学習した結果の Q テーブルを実ロボットの Q テーブルに引き継いだ．実ロボットで追加された状態（“near straight left”，“near straight right”）の Q 値については，シミュレータでの “near center” 状態の Q 値を引き継ぐよ

表 2 提案手法（GP+RL），Q 学習手法（RL+RL），Policy Transfer 手法での比較
Table 2 Comparison of proposed technique (GP+RL) with Q-learning (RL+RL) and Policy Transfer method.

#. situation	GP+RL			RL+RL			Policy Trasnfer		
	avg. steps	lost box	lost goal	avg. steps	lost box	lost goal	avg. steps	lost box	lost goal
1	**19.6**	0	1	20.0	0	1	19.5	0	1
2	**14.7**	0	0	53.0	2	2	29.0	0	1
3	**24.0**	0	1	26.7	0	1	26.0	0	1
4	**10.3**	0	0	11.0	0	0	10.5	0	0
5	**21.6**	0	0	88.0	3	3	41.0	1.3	1.3
6	13.5	0	0	**10.5**	0	0	10.0	0	0
7	26.7	0	1	**26.0**	0	1	27.5	0	1
8	23.0	0	1	**13.0**	0	0	23.0	0	1
9	21.5	0	0	**10.5**	0	0	12.5	0	0
10	**13.5**	0	0	29.0	0	1	30.0	0	1

うに設定した．これは提案手法での “center” への対応づけ（3.3 節）と同等である．実ロボットで追加された行動に対する Q 値は 0.0 で初期化した．

Policy Transfer 手法は，シミュレータで学習された Q テーブルから行動政策（各状態での最適行動）を抽出し，実ロボットに導入する手法である．実ロボットではこの行動政策に基づき行動するが，タスクの成功率が低下したことを認識するとタスク失敗状態の周辺で Q 学習における探索行動を行う[13]．本比較実験では，タスクの成功率を測定するために最近 4 回の試行結果を利用した．このためにはタスクの失敗状態を定義する必要があるので，実験フィールド外に出た場合と，ステップ数が 40 に達し，それ以後数ステップでタスクを達成する見込みがない場合（人間により判断）を失敗と定義した．実ロボットで追加された状態（“near straight left”，“near straight right”）の政策には，“near center” 状態の政策を引き継いで用いた．これも提案手法での “center” への対応づけ（3.3 節）と同じとするためである．

これら比較手法の実ロボットでの学習は基本的には通常の Q 学習で行うので，「状態と行動のずれ」に関しては浅田らの定義[12] をそのまま用いて対処した．

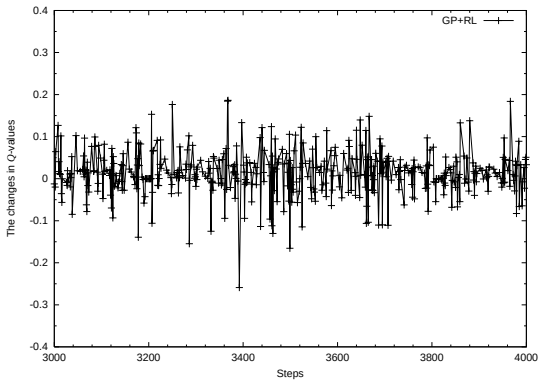
提案手法とこれら手法を比較するために，10 通りの初期配置を設定した．この初期配置は，シミュレータと実ロボットの違いのために，シミュレータの学習完了直後にはうまくタスクを達成できない配置からランダムに選んだものである．この配置について，10 時間の実ロボットでの学習後にどれだけ効率的に行動するようになったかを測定した．この測定は，各状態で Q 値に基づき最適な行動をとる greedy 戦略のもとで行った．

表 2 が提案手法（GP+RL），Q 学習（RL+RL），そして Policy Transfer 手法の比較結果である．この表ではタスクを達成するまでに必要とした平均行動数（avg. steps）とタスク達成までに箱またはゴール領域を見失った回数（lost box，lost goal）を示している．

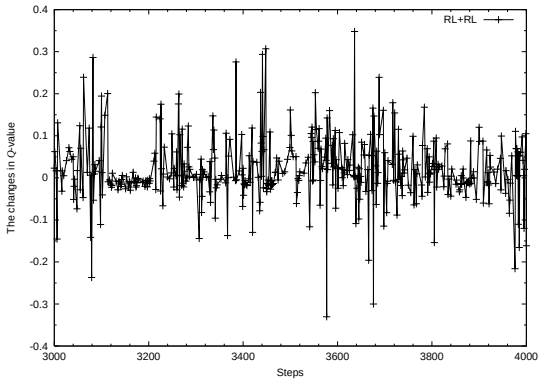
まず提案手法（GP+RL）と Q 学習（RL+RL）との比較を行う．提案手法（GP+RL）は平均行動数において，6 つの配置で RL+RL 手法より優れた値を示した（表 2 中の太字）．一方，RL+RL 手法のほうが優れている配置は 4 つである．さらに，提案手法は RL+RL 手法より箱またはゴール領域を見失う回数がはるかに少ないことが分かる．これらのことから，提案手法が Q 学習よりも効率的な行動を学習できるということが確認された．

次に提案手法（GP+RL）と Policy Transfer 手法の比較を行う．平均行動数において，Policy Transfer 手法のほうが明らかに優れているのは 2 つの配置であり（表 2 中の下線つきの数値），その他の 5 つの配置において提案手法のほうが良く，提案手法のほうが効率的に行動しているといえる（配置 1，4，8 では両手法に有意な差がないとみられる）．また，箱またはゴール領域を見失う回数で比べても，提案手法のほうが効率的に行動していることが分かる．

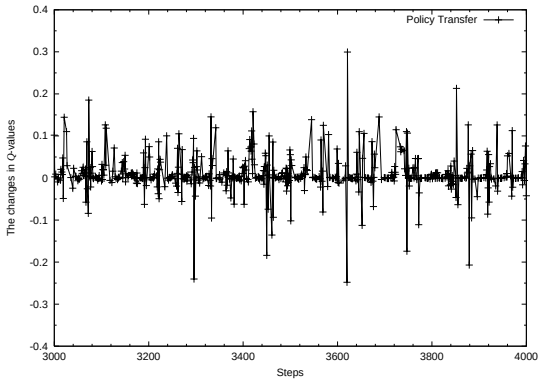
図 7 では実ロボットでの学習中における Q 値更新時の変動を示している．この Q 値変動の絶対値は学習が進むにつれ小さくなるはずであり，最適な Q 値からのずれを示していると考えることができる．図 7 (a) と図 7 (b) を比較すると分かるように，Q 学習（RL+RL）では提案手法よりも頻繁に大きな変動が見られた．このことから，提案手法のほうが Q 学習（RL+RL）よりも短い時間で最適解に収束できると結論できる．Policy Transfer 手法（図 7 (c)）については Q 値の変動が少ないが，これは後に述べるように十分な探索を行っていないためと考えられるので，これを提案手法と比較するのは適当ではないといえる．



(a) Proposed technique (GP+RL).



(b) Q-learning (RL+RL).



(c) Policy Transfer method.

図 7 実ロボットで学習中の Q 値の変動の比較（学習 8 時間後から 10 時間後まで）

Fig. 7 Comparison of changes in Q-values after about 8-hour to 10-hour Q-learning with a real robot.

以上のような差が生じた原因を考察してみる．まず提案手法と Q 学習（RL+RL）との差を考察すると，原因として実ロボットでの行動がシミュレータとまったく同じわけではないことがあげられる．行動の差異によってシミュレータと実ロボットで状態遷移に違いが生じる．それも大まかな状態遷移では同じだが（これは RL+RL 手法でもタスクが達成できていることから分かる），細かな部分でシミュレータでの状態遷移と違いが生じていると考えられる．状態遷移が異なってくると，Q 学習（RL+RL）では実ロボットでの最適な Q 値を探索するので Q 値が変動する．また，シミュレータから実ロボットに Q 値を引き継ぐ際の新しい状態と行動の追加によっても状態遷移が変化している．これらが，上で見られた RL+RL 手法での Q 値の大きな変動を引き起こしていると考えられる．さらにこのように Q 値が大きく変動するために学習に時間がかかっていると考えられる．提案手法でもシミュレータで獲得された結果を利用しているが，実ロボットでの Q 値は最初から学習し直している．そのため，このような状態遷移の細かな変化の影響を受けていないといえる．

提案手法と Policy Transfer 手法を比較すると，後者は行動の最適性を求めていないことが大きな違いである．Policy Transfer 手法では実ロボットはシミュレーション環境で得られた行動政策に基づき行動し，タスク成功率が低下すると探索行動を行う．その一方で，「最適でない行動政策であってもタスク達成に影響を及ぼさなければ保存される」[13] ため，最適な行動を求める手法となってはいない．本実験環境では，シミュレーションでの学習結果を用いるとタスクの完全な失敗という状況はあまり見られなかった．箱やゴール領域を見失っても 1 方向の回転を続けるなどにより復帰し，ステップ数が多くかかるとしてもタスクを達成する場合がほとんどであった．さらにタスクの成功率が低下し探索行動をとる場合にも，Policy Transfer 手法では失敗状態の周辺から探索するので，その周辺の多少の改善で成功率が回復した場合，それ以上の探索が行われない．本実験での失敗状態とは実験フィールドをはみ出た場合，ステップ数が 40 を超えていてタスクを達成する見込みがない場合であり，その周辺から探索を進めても最適行動は得られず，準最適行動で探索を終えてしまったと考えられる．図 7 (c) で見るように Q 値が早期に収束に向かっているのも，十分な探索を行っておらず，行動政策として頻繁に使われる Q 値だけが収束に向かったためであると考えられる．提案手法では実ロボットでの学習により最適行動を探索しており，それが上のような結果の差を生じたといえる．

5.2 関連研究

本提案手法と実現手法は異なるが，遺伝的プログラ

ミングと強化学習を統合した研究はこれまでにいくつか行われている[1),2)]．それらの手法では強化学習として Q 学習を用い，GP の個体は探索する状態空間の構造化を表現している．Iba の QGP[2)] は通常の Q 学習よりも探索効率が上がったと報告されている．

しかしこれらで用いられている手法も多数の個体を用いた集団学習である．図 2 (b) のように，個体学習である強化学習が GP ループの中に入っているために集団中の多数の個体に対して強化学習を行う必要がある．これをそのまま実ロボットに適用すると学習には膨大な時間が必要とされる．そのため，これら手法で実ロボットを対象にした研究はない．

シミュレータと実環境との差異を埋める有効な手法としてシミュレータへのノイズの導入がある[5)]．しかし本手法ではノイズのない理想的なシミュレータで学習し，実ロボット上でも学習することで十分な精度で行動できることが確認された．この大きな理由としては，画像から構成した非常に粗い状態空間を用いたことがあげられる．これは主に状態数を減らし学習時間を短くする目的で用いたが，ノイズのある実環境に対してロバストな認識を行うためにも役立つ．本手法の場合とノイズを用いた場合とでの行動のロバストさに関する比較は今後の検討課題である．

提案手法ではシミュレータと実ロボットの両方での学習を行った．文献 11) ではシミュレータを用いず実機だけで状態空間を構成しつつ強化学習を行う手法が提案されている．本研究の実験環境では，ロボットの 1 行動あたりにかかる時間が比較的長い．この環境下で，状態空間を構成するために実機で偏りのない十分なデータを収集するには，膨大な時間がかかることが予想される．このような難点が考えられるものの，状態空間を動的に構成するという点で非常に興味深い手法である．

5.3 今後の展望

本研究では簡単化のため離散的な行動のみを用いた．しかし連続的な行動を用いるとしたほうがより現実的な応用が可能となる．たとえば，学習初期に「左に 30.0 度回転」であったものが，実ロボットの動作特性に適応し「左に 31.5 度回転」になるなどというものである．今後はこのような連続的な行動での実験を考えている．

本提案手法の拡張として，マルチエージェント問題のようにより複雑な問題への応用が考えられる．さらに本手法に基づけば，形態が異なるロボットであっても，ほぼ同じシミュレータと強化学習の設定を用いて動かすことができる．今後，さまざまなロボット，たとえば “HOAP-1”（富士通オートメーション株式会社）や “Khepera” などを用いた実験を考えている．その実験の初期の結果は文献 7) に述べられている．

6. おわりに

本研究では遺伝的プログラミングと強化学習を組み合わせた手法に基づき実ロボットでの実時間学習を行う手法を提案し，実験を通してその有効性を示した．

学習開始時には実ロボットの特性が分かっていないために箱の移動に失敗する動作も見られた．しかし本手法の 10 時間の学習を通じて，シミュレータで再現されていないような実ロボットの動作特性に適応した．これは本手法の実ロボットでの学習が有効に働いたことを示している．

本実験では非常に粗い状態空間を用いた．これは主に状態数を減らし学習時間を短縮するためである．この粗い状態空間と限定された行動のみを用いて 4 足のロボットがこのタスクを達成できたことは注目すべきことといえるだろう．

本手法にはまだ改善すべき点がある．それは図 2 (a) に示した点線ループ，実環境での学習から GP，シミュレータ側へのフィードバックである．実ロボットを用いると機械的な動作をともなうため時間がかかるのは不可避である．シミュレータで学習可能な要素については実ロボットではなくシミュレータ上で学習させるべきである．そのため実環境で行動中に得られたパラメータなどをフィードバックし，シミュレータの精度を改善することで，実ロボットへの適用をスムーズにすることが考えられる．この実現については今後の課題である．

参考文献

1) Downing, K.L.: Adaptive genetic programs via reinforcement learning, *Proc. 3rd Annual Genetic Programming Conference* (1998).
2) Iba, H.: Multi-Agent Reinforcement Learning with Genetic Programming, *Proc. 3rd Annual Genetic Programming Conference* (1998).
3) Koza, J.R.: *Genetic Programming, On the Programming of Computers by means of Natural Selection*, MIT Press (1992).
4) OPEN-R プログラミング SIG（著），ソニー株式会社エンタテイメントロボットカンパニー（監修）：C++で AIBO を自在に動かす：OPEN-R プログラミング入門，インプレス (2002).
5) Schultz, A.C., Ramsey, C.L. and Grefenstette, J.J.: Simulation-Assisted Learning by Competition: Effects of Noise Differences Be-

tween Training Model and Target Environment, *Proc. 7th International Conference on Machine Learning*, San Mateo, pp.211–215, Morgan Kaufmann (1990).
6) Sutton, R.S., Barto, A.G.（著），三上貞芳，皆川雅章（共訳）：強化学習，森北出版 (2000).
7) Yanai, K. and Iba, H.: Multi-agent Robot Learning by Means of Genetic Programming: Solving an Escape Problem, *Evolvable Systems: From Biology to Hardware, Proc. 4th International Conference on Evolvable Systems, ICES'2001*, Tokyo, October 3-5, 2001, Liu, Y., et al.(Eds.), pp.192–203, Springer-Verlag, Berlin, Heidelberg (2001).
8) 伊庭斉志：遺伝的プログラミング，東京電機大学出版局 (1996).
9) 木村 元，山下 透，小林重信：強化学習による 4 足ロボットの歩行動作獲得，電気学会電子情報システム部門誌，Vol.122-C, No.3, pp.330–337 (2002).
10) 畝見達夫：強化学習，人工知能学会誌，Vol.9, No.6, pp.830–836 (1994).
11) 高橋泰岳，浅田 稔：実ロボットによる行動学習のための状態空間の漸次的構成，日本ロボット学会誌，Vol.17, No.1, pp.118–124 (1999).
12) 浅田 稔，野田彰一，俵積田健，細田 耕：視覚に基づく強化学習によるロボットの行動獲得，日本ロボット学会誌，Vol.13, No.1, pp.68–74 (1995).
13) 港 隆史，浅田 稔：環境の変化に適応する移動ロボットの行動獲得，日本ロボット学会誌，Vol.18, No.5, pp.706–712 (2000).

(平成 15 年 4 月 11 日受付)
(平成 15 年 6 月 14 日再受付)
(平成 15 年 7 月 1 日採録)

神尾正太郎

1979 年 7 月 8 日生．2002 年東京大学工学部電子情報工学科卒業．同年東京大学大学院新領域創成科学研究科基盤情報学専攻修士課程入学，現在に至る．進化論的計算手法や強化学習のロボットへの応用に興味を持つ．

三橋 秀行

1979 年 8 月 26 日生．2003 年東京大学工学部電気工学科卒業．同年東京大学大学院新領域創成科学研究科基盤情報学専攻修士課程入学，現在に至る．強化学習のロボット応用に興味を持つ．

伊庭 斉志（正会員）

1962 年 10 月 28 日生まれ．1985 年東京大学理学部情報科学科卒業．1990 年東京大学大学院工学系研究科情報工学専攻博士課程修了．工学博士．同年電子技術総合研究所入所．1998 年から東京大学大学院工学系研究科電子情報工学専攻助教授．1999 年から東京大学大学院新領域創成科学研究科基盤情報学専攻助教授．進化システムおよび人工知能基礎の研究に従事．特に遺伝的プログラミング，学習，推論，知能ロボットに興味を持つ．