強化学習によるロボットの動作獲得のための基底関数に基づく行動空間生成手法 DCOB
– 実機多自由度ロボットの匍匐動作への適用 –

DCOB: Constructing Action Space from Basis Functions for Motion Acquisition of Robots by Reinforcement Learning
- Application to Crawling Task of Actual Small Size Multi-link Robot -

○学 山口 明彦 (奈良先端大)　　正 高松 淳 (奈良先端大)　　正 小笠原 司 (奈良先端大)

Akihiko Yamaguchi, Jun Takamatsu, Tsukasa Ogasawara
Nara Institue of Science and Technology
{akihiko-y, j-taka, ogasawar}@is.naist.jp

This paper investigates an applicability of the discrete action set *DCOB* proposed by Yamaguchi et al. to motion acquisition task of an actual small size multi-link robot by reinforcement learning. We apply the DCOB to a crawling task of a spider robot (ROBOTIS Bioloid), and compare the DCOB with a *grid action set* as a conventional method. The experimental results demonstrate the advantage of the DCOB both in learning speed and ability to acquire performance.

Key Words: Reinforcement Learning, Discrete Action Set, DCOB, Motion Learning, Crawling

1. はじめに

将来ロボットが日常生活などの多様な環境に進出する上で，エンドユーザ側で行動目的を指示し，ロボットに獲得させられるような技術が不可欠である．この具体的な解決策として，強化学習をロボットの行動獲得に応用する研究がなされている [1, 2, 3, 4]．しかしながら，高次元の制御入力空間を持つロボットの扱いは，未解決問題のひとつであり，タスクやロボットの特性に基づいて階層構造や低次元の行動空間を構成する方法 [2, 3]，模倣などで，あらかじめ基本となるパタンを与える方法 [4] などが研究されている．

本論文では，Yamaguchi ら [1] によって提案された離散行動集合 DCOB（Directed to the Center Of a Basis function）の実機における有効性を検証する（[1] ではシミュレーション実験のみ）．DCOB とは，多関節のロボットに対して，価値関数を近似するために与えられた基底関数集合から離散行動集合を生成する手法である．DCOB は，従来の離散化手法と比べて高いパフォーマンスが得られるように工夫されており，さらに，タスクやロボットの性質を利用して基底関数の数を削減すると，学習速度も向上させられるという特徴を持つ．具体的には，本論文では，(1) ROBOTIS 社製のロボット Bioloid（King Spider として組み立てた）の匍匐タスクに，DCOB を用いた強化学習を適用，(2) シミュレーション結果 [1] で，学習速度が大きくかつ比較的高いパフォーマンスが得られた「グリッド行動集合」を用いた場合と比較，を行う．実機実験の結果，DCOB の方がグリッド行動集合と比べて学習速度，獲得されたパフォーマンスのいずれにおいても優れているという結果が得られた．

2. 強化学習の基礎

強化学習の目的は，入力が**状態** $x_n \in \mathcal{X}$ と**報酬** $R_n \in \mathbb{R}$，出力が**行動** $a_n \in \mathcal{A}$ であるシステム（**エージェント**）が，**割引収益**の期待値 $\mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} R_{n+k}\right]$ を最大化するような**方策** $\pi(x_n) : \mathcal{X} \rightarrow \mathcal{U}$ を獲得することと定義される．$n \in \mathbb{N} = \{0, 1, \dots\}$ は時間ステップ，$\gamma \in [0, 1)$ は**割引率**を表す．Q-learning や Sarsa などの強化学習手法では，**行動価値関数** $Q(x, a) : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ が状態 x で行動 a を取ったときに将来期待される割引収益を推定するように学習される．このとき，最適な行動は**グリーディ方策** $\pi(x) = \arg\max_a Q(x, a)$ によって行動価値関数から得られる．

状態空間 $\mathcal{X}$ や行動空間 $\mathcal{A}$ が連続である場合には，適当な関数近似器が行動価値関数 $Q(x, a)$ を近似するために用いられる．本論文では，$\mathcal{X}$ は連続，$\mathcal{A}$ は離散である．そこで，実装が容易で安定に学習できる線形の関数近似器を用いて，行動価値関数を以下のように近似する．

$$Q(x, a) = \theta_a^\top \phi(x) \tag{1}$$

$$\phi(x) = (\phi_1(x), \dots, \phi_{|\mathcal{K}|}(x))^\top \tag{2}$$

ただし $\theta_a \in \mathbb{R}^{|\mathcal{K}| \times 1}$ は行動 a に関するパラメタベクトル，$\phi(x)$ は特徴ベクトル（基底関数の出力），$\mathcal{K} = \{\phi_k \mid k = 1, 2, ..\}$ は事前に定義された**基底関数**の集合をそれぞれ表す．「価値関数が線形」とは，$\phi(x)$ のパラメタを学習の対象としないことを意味する．このときパラメタベクトルは $\theta^\top = (\theta_1^\top, \theta_2^\top, \dots, \theta_{|\mathcal{A}|}^\top)$ と表される．

本論文では，基底関数として，強化学習の研究でしばしば用いられる正規化ガウシアンネットワーク（NGnet: Normalized Gaussian Network）を仮定する．NGnet の基底関数は，

$$\phi_k(x) = \frac{G(x; \mu_k, \Sigma_k)}{\sum_{k' \in \mathcal{K}} G(x; \mu_{k'}, \Sigma_{k'})}, \tag{3}$$

と表される．ただし $G(x;\mu,\Sigma)$ は平均 μ，共分散 Σ のガウス関数を表す．

3. DCOB

DCOB とは，価値関数の近似のために与えられた基底関数集合 $\mathcal{K}$ から，離散行動集合 $\mathcal{A}_{\mathrm{DCOB}}$ を生成する手法である [1]．基底関数 $k \in \mathcal{K}$ は，パラメタとして中心状態 μ_k を持つことを前提としており，DCOB のひとつの行動は，ある基底関数の中心状態へ向かう状態変化を引き起こすようにロボットを制御する．$\mathcal{A}_{\mathrm{DCOB}}$ は状態変化の素早さを決定するパラメタ $g \in \mathcal{I}$（$\mathcal{I}$ は離散集合）も含んでいて，$\mathcal{A}_{\mathrm{DCOB}} \triangleq \mathcal{I}\times\mathcal{K}$ と定義される．

3.1 DCOB の特徴

DCOB は以下の特徴を持つ．

(1) 基底関数の解像度以上の細かさで方策を表現できないことを考慮して，ひとつの行動が引き起こす状態変化が，隣接する基底関数に到達する程度となるように設計されている．このため報酬が伝搬しやすく，学習速度の向上につながる．

(2) 状態空間の制約を考慮した行動が生成できる．例えば，関節角に制約がある場合，通常は基底関数を関節可動領域の内部または境界に配置するため，関節可動領域の外側へ向かうような行動は生成されない．このため同数のグリッド行動集合などと比較した場合，DCOB の方が関節可動領域の境界付近でより多様な行動を選択可能となるため，より高いパフォーマンスが期待できる．

(3) ロボットのダイナミクスに基づく基底関数の配置手法 [1, 5] などによって基底関数集合の要素数を削減することで，DCOB の要素数も削減できる．これにより，学習速度が向上する．

3.2 実行の概要

DCOB の各行動に対するロボットの制御の概要について述べる．DCOB は，以下を仮定する．

(A) 個々の基底関数 $k \in \mathcal{K}$ は，固定された中心 $\mu_k \in \mathcal{X}$ を状態空間上に持つ．

(B) 「参照軌道」上の各点が定義される空間 $\mathcal{Q}$（例えば関節角空間）が定義されており，状態 $x \in \mathcal{X}$ から $q \in \mathcal{Q}$ を抽出する関数 $q = C_{\mathrm{p}}(x) : \mathcal{X} \to \mathcal{Q}$，及び，状態 $x \in \mathcal{X}$ から $q \in \mathcal{Q}$ の時間微分（例えば関節角速度）を抽出する関数 $\dot{q} = C_{\mathrm{d}}(x)$ が与えられている．

(C) 参照軌道 $q^{\mathrm{d}}(t)$ を追従する制御器，すなわち制御入力 $\tilde{u} \in \tilde{\mathcal{U}}$（例えばトルク）を参照軌道と現在の状態から計算する関数 $\tilde{u}(t) = Ctrl(x(t), q^{\mathrm{d}}(t+\delta\mathrm{t}))$（例えば PD 制御器）が与えられている（$\delta$t は制御周期）．

ある行動 $(g,k) \in \mathcal{I}\times\mathcal{K}$ が与えられたとき，DCOB では，以下の流れで行動を生成する（Fig.1）．

Algorithm 1: Executing an action in the DCOB

Input：開始状態 $x_n = x(t_n)$，

行動 $a_n = (g,k) \in \mathcal{I}\times\mathcal{K} = \mathcal{A}_{\mathrm{DCOB}}$

1: $x^{\mathrm{trg}} \leftarrow \mu_k$

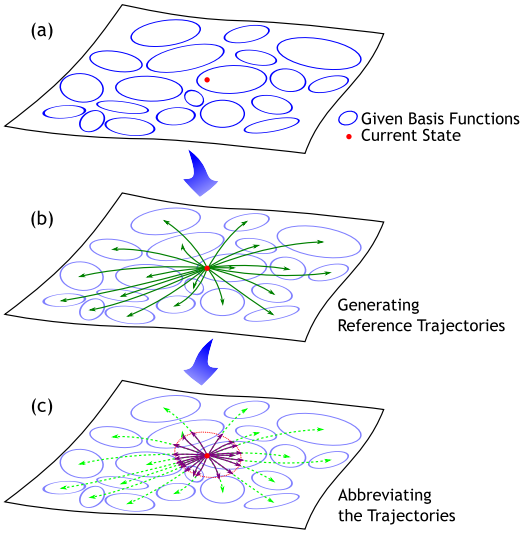

Fig.1 Illustration of how the possible actions of the DCOB at the current state are generated [1]

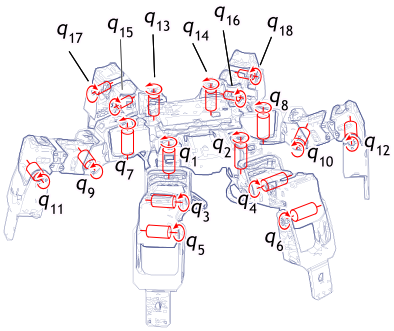

Fig.2 King Spider (ROBOTIS Bioloid) which has 18 DoF. Its DoF is constrained to five such that $q_1{=}{-}q_2$, $q_3{=}q_5{=}{-}q_4{=}{-}q_6$, $q_7{=}{-}q_8$, $q_9{=}q_{11}{=}{-}q_{10}{=}{-}q_{12}$, $q_{15}{=}{-}q_{16}$, and $q_{13}{=}q_{14}{=}q_{17}{=}q_{18}{=}0$

2: g, x_n, x^{trg} から到達時間幅 T_{f} を計算

3: **軌道生成**： x_n から x^{trg} に時間幅 T_{f} で遷移する参照軌道 $q^{\mathrm{d}}(t_n + t_a)$, $t_a \in [0, T_{\mathrm{f}}]$ を生成

4: **軌道短縮**： 参照軌道を $t_a \in [0, T_{\mathrm{n}}] \subseteq [0, T_{\mathrm{f}}]$ に短縮．ここで $T_{\mathrm{n}} \leqslant T_{\mathrm{f}}$ は，隣接する基底関数に遷移する程度の時間幅として計算される

5: **軌道追従**：短縮された参照軌道を，与えられた制御器で追従： $\tilde{u}(t_n + t_a) = Ctrl(x(t_n + t_a), q^{\mathrm{d}}(t_n + t_a + \delta\mathrm{t}))$, $t_a \in [0, T_{\mathrm{n}})$

6: 行動 a_n が終了; $n \leftarrow n+1$

詳細は [1] を参照されたい．

4. 実機実験

本章では，強化学習を ROBOTIS 社製のロボット Bioloid（King Spider）の匍匐タスクに適用し，DCOB を用いた場合と，グリッド行動集合を用いた場合を比較する．

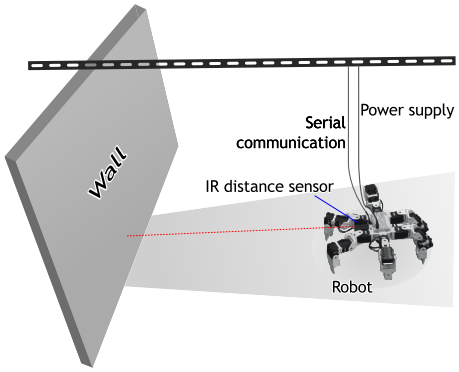

Fig.3 Setup of experimental environment

4.1 ロボットシステムと実験環境

制御入力： King Spider は 18 個のサーボを有するが，学習を高速化するためにいくつかの関節を連動・固定し，Fig.2 に示す 5 自由度に拘束してある．すなわち，ロボットの制御入力（目標関節角）は

$$\tilde{u} = (q_1^{\mathrm{d}}, q_3^{\mathrm{d}}, q_7^{\mathrm{d}}, q_9^{\mathrm{d}}, q_{15}^{\mathrm{d}})^\top, \tag{4}$$

によって定義される．また，これらの可動範囲は，

$$\tilde{u}_{\max} = (-\pi/9, \pi/2, \pi/4, \pi/2, \pi/2)^\top, \tag{5}$$

$$\tilde{u}_{\min} = (-\pi/4, -\pi/2, -\pi/4, -\pi/2, -\pi/2)^\top, \tag{6}$$

に制限されている．

状態： マルコフ性を満たすようにロボットの状態を選ぶと，ロボットのグローバルな位置・姿勢，全関節角（上記の連動や固定は，達成できないことがあるため），及びこれらの速度項が必要となる．しかし，計測の困難さ，次元数の増大による学習時間の増加を考慮して，

$$x = (q_1, q_3, q_7, q_9, q_{15})^\top, \tag{7}$$

を強化学習に与える状態とする．このときタスクは非マルコフとなる．

センサ： 距離センサ：ロボットには，障害物までの距離を計測する赤外線センサが搭載されている．このセンサ値（低域通過フィルタを適用したもの）を d_{IR} とおく．

実験環境： ロボット及び環境は，Fig.3 に示すように配置される．ロボットは壁の前に置かれており，赤外線距離センサは壁に反射させて用いる．ロボットは計算機[*1]とシリアル通信で接続されており，約 $\delta \mathrm{t} = 0.1$[s] ごとに目標関節角 $\tilde{u}$ がロボットに送られる．

4.2 タスク設定

匍匐タスクの目的は x 軸の正の方向にできるだけ移動するというものであり，以下のように報酬関数を設計する．

$$r(t) = -\dot{d}_{\mathrm{IR}}(t) - 0.15 \tag{8}$$

ここで $\dot{d}_{\mathrm{IR}}(t)$ は $d_{\mathrm{IR}}(t)$ を時間微分したものであり，$-\dot{d}_{\mathrm{IR}}(t)$ はロボットの速度に相当する．報酬関数が連続時間について定義されているのは，異なる行動集合の比較を公平にするためであり，ある行動に対して与えられる報酬は，その行動の実行時間について $r(t)$ を積分することで計算される．よって，ロボットの報酬の総和（収益）は，$-\dot{d}_{\mathrm{IR}}(t)$ の項は移動距離を表し，定数項（0.15）は所要時間に対するペナルティを表す．

各エピソードは，初期状態（$q_1 = -\pi/9$, $q_3 = 0$, $q_7 = 0$, $q_9 = 0$, $q_{15} = 0$）からはじまり，$t > 50$[s]，またはロボットがテーブルに接する（操作者が判定），またはケーブルが外れるなどのトラブルで終了する．

4.3 強化学習手法

強化学習手法としては，Peng らの Q(λ)-learning [6] を用いる．割引率は $\gamma = 0.9, \lambda = 0.9$ とし，ステップサイズパラメタは $\alpha = 0.3\exp(-0.002 N_{\mathrm{eps}})$ で減衰させる．探索手法にはボルツマン選択を用い，温度パラメタを $\tau = 0.1\exp(-0.004 N_{\mathrm{eps}})$ で減衰させる．N_{eps} はエピソード数を表す．なお，Q(λ)-learning を実装する際，エリジビリティトレースを安定化するために *replacing trace* [7] を適用する．

行動価値関数の近似には線形関数近似器を使用し，パラメタはゼロで初期化する．基底関数には NGnet を用い，$3 \times 3 \times 3 \times 3 \times 3$ のグリッド上に計 243 個の基底関数を配置する．

4.4 行動空間の設定

DCOB： DCOB を適用するために，C_{p}, C_{d}, 及び $\mathcal{I}$ を以下のように定める．

$$C_{\mathrm{p}}(x) = (q_1, q_3, q_7, q_9, q_{15})^\top \tag{9}$$

$$C_{\mathrm{d}}(x) = (0, 0, 0, 0, 0)^\top \tag{10}$$

$$\mathcal{I} = \{0.5\} \tag{11}$$

なお，本実験で用いるロボットは目標関節角を直接与えられるので，*Ctrl* はロボットに備わっているものとみなされる（$q^{\mathrm{d}}(t + \delta \mathrm{t})$ を直接ロボットに出力する）．

グリッド行動集合（Grid3）： ここでは「グリッド行動集合」 $\mathcal{A}_{\mathrm{G}}$ を，目標関節角をグリッドで区切って離散化した行動集合として定義する．

$$\mathcal{A}_{\mathrm{G}} = \{\Delta q \mid \Delta q = (\delta q_1, \delta q_3, \delta q_7, \delta q_9, \delta q_{15})^\top, \\ \delta q_{1,3,7,9,15} \in \{0, \pm\Delta\varphi, \ldots, \pm\tfrac{\mathrm{N_{grid}}-1}{2}\Delta\varphi\}\} \tag{12}$$

ただし $\Delta\varphi = \pi/12$ とする．$\mathrm{N_{grid}}$ はグリッドの分割数を表し，$\mathrm{N_{grid}} = 3$ を用いる．ある行動 $\Delta q \in \mathcal{A}_{\mathrm{G}}$ は以下のように実行される．

$$q^{\mathrm{d}}(t) = C_{\mathrm{p}}(x(t)) + \Delta q \tag{13}$$

ただし $t = t_n + t_a$, $t_a \in [0, \mathrm{T_G})$ と定義される．$\mathrm{T_G} = 0.5$[s] は行動が実行される時間を表す定数である．

4.5 結果と考察

DCOB, Grid3 とも 4 回ずつ試行を行わせた．各試行は，学習曲線及び動作パタンから収束を目視で判

[*1] Pentium M, 2[GHz], 512[MB] RAM, Debian Linux.

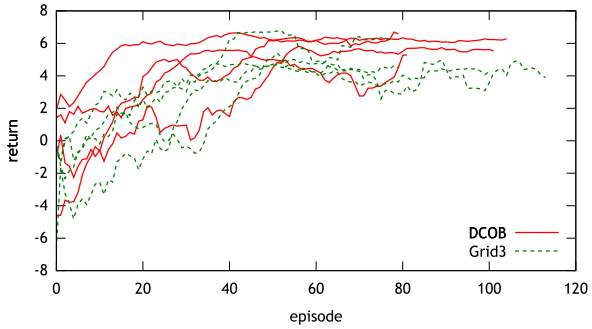

Fig.4 Resulting learning curves of the crawling task. Each curve shows the return per episode in a run. To see the tendency of each curve, a low-pass filter with a time constant of 10 episode is applied

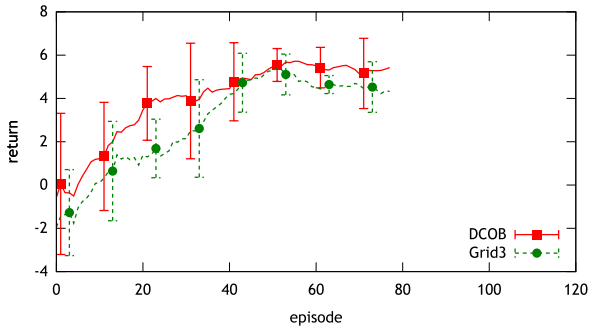

Fig.5 Averaged learning curves of the crawling task. Each curve shows the mean of the return over 4 runs in 0 ∼ 77 episode. Error bar denotes ±1 standard deviation. A low-pass filter with a time constant of 10 episode is also applied

定し，終了とした．結果の学習曲線を Fig.4 に示す．Fig.5 は各 4 つの曲線が重複する 0 ∼ 77 エピソードの間で平均と分散を計算しプロットしたもの，Fig.6 は各曲線の最後の 10 エピソード分から求めた平均と分散をプロットしたものである．Fig.7 は DCOB によって獲得された動作を表す（添付の動画も参照されたい）．Fig.5 から，DCOB が 20 ∼ 30 エピソード付近で既にある程度高い収益を獲得しており，DCOB の方が Grid3 よりも速く学習していると言える．一方，Fig.6 は獲得された動作のパフォーマンスを表しており，DCOB の方が Grid3 よりも高い傾向にある．よって，実機の匍匐タスクにおいても，DCOB が Grid3 と比べて学習速度，動作パフォーマンスのいずれにおいても優れていることが確認された．

5. 結論

本論文では，Yamaguchi ら [1] によって提案された離散行動集合 DCOB を，実機の多自由度ロボットの運動学習に適用した．具体的には，(1) ROBOTIS 社製のロボット Bioloid（King Spider）の匍匐タスクに，DCOB を用いた強化学習を適用，(2)「グリッド行動集合」を用いた場合と比較，を行った．実験の結果，DCOB の方がグリッド行動集合と比べて学習速度，獲得されたパフォーマンスのいずれにおいても優れているという結果が得られた．

ロボットのダイナミクスに基づく基底関数の配置手法 [1, 5] を用いれば，より高次元の運動学習タスクに

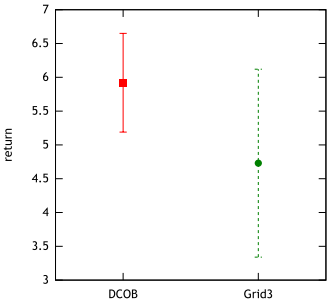

Fig.6 Performance of the acquired motion: the average and the ±1 standard deviation of the return over the last 10 episodes

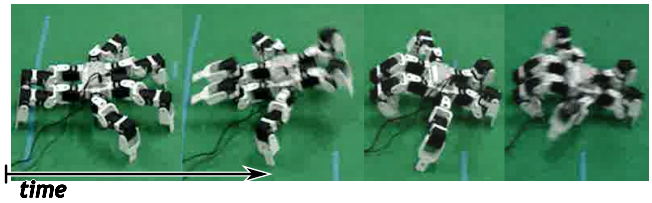

Fig.7 Snapshots of acquired crawling motion (4.8[s], 5.4[s], 6.0[s], 7.2[s])

も適用できると考えられ，今後の課題としたい．

[1] A. Yamaguchi, J. Takamatsu and T. Ogasawara: “Constructing action set from basis functions for reinforcement learning of robot control”, the IEEE Internactional Conference in Robotics and Automation (ICRA’09), Kobe, Japan, pp. 2525–2532 (2009).

[2] J. Morimoto and K. Doya: “Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning”, Robotics and Autonomous Systems, **36**, 1, pp. 37–51 (31 July 2001).

[3] Y. Nakamura, T. Mori, M. Sato and S. Ishii: “Reinforcement learning for a biped robot based on a CPG-actor-critic method”, Neural Networks, **20**, 6, pp. 723–735 (2007).

[4] J. Peters, S. Vijayakumar and S. Schaal: “Reinforcement learning for humanoid robotics”, Humanoid2003, IEEE-RAS International Conference on Humanoid Robots (2003).

[5] 山口, 杉本, 川人：“回避行動の再利用メカニズムを備えた強化学習手法と多関節ロボットの全身運動学習への応用”, 日本ロボット学会誌, **27**, 2, pp. 209–220 (2009).

[6] J. Peng and R. J. Williams: “Incremental multi-step Q-learning”, International Conference on Machine Learning, pp. 226–232 (1994).

[7] S. P. Singh and R. S. Sutton: “Reinforcement learning with replacing eligibility traces”, Machine Learning, **22**, 1-3, pp. 123–158 (1996).