Vol.2016-MPS-108 No.3
Vol.2016-BIO-46 No.3
2016/7/4

モジュール切替による未知環境に適応可能な探索エージェントの行動制御

畠 崇人[1,a)]　菅沼 雅徳[1]　長尾 智晴[1]

概要：本論文では，未学習の環境においても適応可能な探索エージェントの行動規則の構築のため，複数の学習環境を用いて多彩な行動の獲得を試みる．従来の学習手法の場合，1 つの行動規則を複数の環境で学習させても，それらを網羅した行動を獲得することは困難である．そこで本手法では，学習環境を性質の異なるいくつかの基本環境に分割し，各基本環境に特化したモジュールを複数生成する．行動規則の最適化にはグラフ構造プログラムの自動生成手法である Graph Structured Program Evolution(GRAPE) を用いる．未知環境では，学習した各モジュールをスイッチャによって切り替える．スイッチャは周囲の環境で有効に働くモジュールを選択する機能を有する．部屋の探索を行う自律エージェントの構築において，従来の単純 GRAPE と比較し未知環境でより高い探索性能を示すことができた．

キーワード：自律エージェント，モジュール化，汎用化，自動プログラミング，進化計算

Controling Adaptable Exploration Agent using Switching Modules

Takahito Hata[1,a)]　Masanori Suganuma[1]　Tomoharu Nagao[1]

Abstract: In this paper, we try to acquire various behavior patterns using several learning environments in order to construct behavior rules of an adaptable exploration agent. In case of a privious learning method using one behavior rule, it is hard to acquire the behavior that covers all learning environment. In our method, we devide learning enviornments into some primitive environments whose properties differ each other, and then generate modules that are specialized for each primitive environments. To optimize behavior rules of agents, we adopt GRAPE which automatically generates a graph structured program. In unknown environments, each modules are switched by the "switcher". The switcher selects the module that acts better in a neighboring environment. In constructing exploration agent, our method could mark higher exploration rate in unknown environments compared to simple GRAPE.

Keywords: autonomous agent, modularization, generalization, automatic programming, evolutionally computing

1. はじめに

進化計算を用いて木構造プログラムを最適化する遺伝的プログラミング (Genetic Programming;GP)[1] が提案されて以来，様々な問題に対して GP の有効性が示されている．しかし，複雑な行動が要求される自律エージェントの構築に際し，木構造プログラムを扱う GP 手法では，木構造プログラムの表現力の低さが問題点として挙げられる．自律エージェントの行動規則を木構造プログラムで表現した場合，毎回ルートノードから遷移させ次の行動を決定させる必要があり，エージェントに対する入力が変わらない場合は同じ行動を選択し続けることになる．これでは構築されるエージェントが単調な動きのみしか行動獲得することができないと考えられる．この問題に対して，エージェント

[1] 横浜国立大学大学院環境情報学府
Graduate School of Environment and Infomation Sciences, Yokohama National University, Yokohama, Kanagawa 240-8501, Japan
[a)] hata-takahito-hg@ynu.jp

© 2016 Information Processing Society of Japan

の行動規則を任意のグラフ構造で表現することよって，木構造では表現できないループを伴う繰り返し行動などを表現することができるほか，出力ノードから他のノードへ遷移させることが可能であるため，エージェントに対する入力が同じ場合でも，ノードの遷移状況によって異なる動きを表現することができる．また出力ノードが連続して結合されていれば，前進と右旋回を組み合わせて曲線を描きながら前進などといったように，設定した出力ノードでは表現しきれない行動も獲得することができる．

また，一般に複雑な行動を表現するためには，複数のタスクに分割して学習する必要がある．分割されたタスクをGP によって学習させる手法として，GP の進化プロセスを階層的に行う手法が提案されている [2], [3]．機械学習全般における階層的な学習手法は，Stone らによって Layered Learning[4] として定式化されており，直接解くことが困難なタスクをサブタスクに分割し，それらを複数のレイヤ $\{L_1, L_2, ..., L_n\}$ を通してボトムアップ式に学習を行うものである．学習が簡単とされる低次のレイヤから学習を始め，それぞれのレイヤ L_i は次層 L_{i+1} での学習を促進するように学習を進める．文献 [4] では，この Layered Learning の考え方をサッカーエージェントの行動規則の自動獲得に応用している．サッカーエージェントの学習を，①個々のエージェントがボールを受け取る (奪う) 学習，②複数のエージェントを使ったパスの学習，③チーム全体を絡めたパスかシュートかを選択するための学習の 3 つのレイヤに分割して学習を行っている．この Layered Learning の枠組に則り，文献 [2] や [3] では直接解くことが困難なタスクを分割し，GP の個体を進化させてゆく途中でレイヤを順に切替えることで，難しいタスクでも比較的速く良好な個体を得ることができている．しかし，レイヤの前後で共通の個体群を扱うため，サッカーエージェントの学習のように下位レイヤの学習が上位レイヤの学習を助長するような場合であれば有効であるが，レイヤごとの学習環境が互いに性質の異なるものであった場合，レイヤの移動によって個体の構造が破壊され学習が進まなくなってしまう．この問題に対し Jackson は，分割されたそれぞれのタスクにおいて最適化された GP 木を部分木として保存し，最終的にすべての部分木を統合する手法を提案している [5]. これと同様に GP の部分木を構築する手法として，Automatically Defined functions(ADFs)[6] が Koza によって提案されているが，ADF-GP ではすべての部分木を同時に進化させるため，それぞれの部分木がどのような機能であったり，どのような構造を持っているかはわからない．一方文献 [5] では，対象とするタスクを複数のサブタスクに分割して，それぞれのサブタスクごとに部分木を最適化することで，特定の機能を持つ部分木を獲得することが可能である．入力の分割が簡単な even-parity 問題や majority-on 問題の回路設計実験において，バイナリで表現される非負整数値の入力を 2,4,8 分割してそれぞれのケースごとに GP 木を最適化しており，従来の GP 手法と比較して高い成功率を示している．

本手法ではまず，木構造では表現できない行動規則の獲得のため，エージェントの行動規則にグラフ構造プログラムを採用する．グラフ構造プログラムの最適化には，Graph Structured Program Evolution(GRAPE)[7] を用いる．さらに，未知環境に対する適応性向上のため，学習環境を複数の基本環境に分割して汎用的な行動パターンを持つ探索エージェントの構築を試みるが, 探索エージェントの学習を階層的に分割することを考えた場合，低次の層の学習が高次の層の学習を助長するように学習プロセスを分割することが困難であり，階層的に学習を進めることが有効ではない．また，階層的な学習をさせず，複数の基本環境を 1 つの行動規則で学習させようとした場合，すべての基本環境を網羅するような行動を表現することが難しく，最適化が困難であると考えられる．そこで本手法では，分割した基本環境を別々の行動規則として学習し，各基本環境に特化したモジュールを構築する．さらに，周囲の環境と基本環境との類似度を計り，環境に応じて行動規則を切り替えることで，未知環境でも適応的に行動可能な探索エージェントを構築する．

2. Graph Structured Program Evolution(GRAPE)

進化計算によって木構造プログラムを最適化する手法として J.Koza が GP を提案し，関数同定や回路設計問題，またエージェントの行動規則の最適化などで利用されている．一方で GRAPE[7] は，グラフ構造で表現されたプログラムを最適化する手法である．GRAPE の構造を**図 1** に示す．GRAPE のプログラムは最適化対象である有向グラフと“データセット”から成る．有向グラフ中の各ノードにおいて，そのノードに応じた処理がデータセット内の値に対して行われる．これによって複数のデータ型の取り扱いが可能となる．また，GRAPE では 1 次元整数列を遺伝子型とすることで，GA で用いられる突然変異や一様交叉などの遺伝オペレータを利用することができる．さらに，この遺伝子型を固定長とすることで，GP の問題の 1 つとされているブロートを回避することができる．世代交代モデルとしては Simple GA(SGA) よりも Minimal Generation Gap(MGG)[8] を用いたほうが，基本的なプログラム生成問題において効率の良い進化が行われることが示されている．先行研究において，階乗やフィボナッチ数列などを求める基本的なプログラムを初め，リストのソーティングプログラム [9]，マルチエージェントの行動規則の自動獲得 [10] などにおいて GRAPE の有効性が示されている．

本手法においても行動規則に GRAPE を用いることで，

© 2016 Information Processing Society of Japan

Vol.2016-MPS-108 No.3
Vol.2016-BIO-46 No.3
2016/7/4

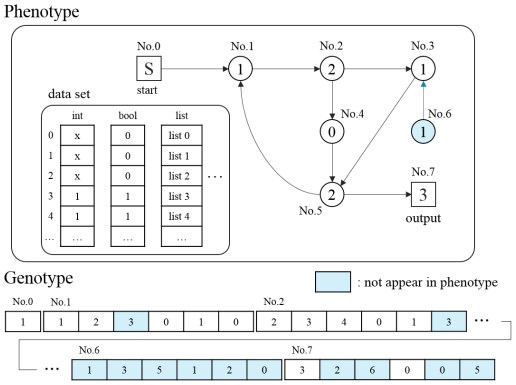

図 1　GRAPE の構造例

Fig. 1　An example of the structure of GRAPE.

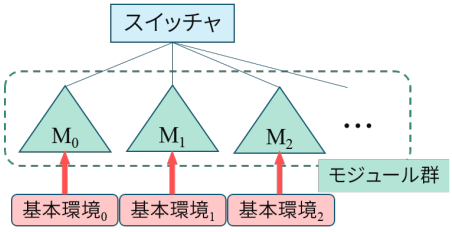

図 2　提案手法の構造

Fig. 2　An architecture of the proposed method.

進化途中でのブロートを抑えつつ，グラフ構造プログラムのによる表現力の高い探索エージェントを構築できると考えられる．

3. 提案手法

本手法では，性質の異なる複数の行動規則を統合することで未知環境にも適応可能な探索エージェントを構築する．性質の異なる行動規則を構築するため，本手法ではまず互いに性質の異なるような学習環境を設計し，それぞれの環境ごとに行動規則を GRAPE によって最適化することを考える．1 つの行動規則を性質の異なる環境で最適化させようとすると，複数の行動パターンを 1 つの行動規則で表現しなければならないため最適化が困難であり，より良い性能のエージェントを構築することはできないと考えられる．一方，学習環境ごとに行動規則を分割することで，特定の環境に特化した行動規則が容易に最適化され，さらにそれらを組み合わせることで，より多彩な行動が表現できると考えられ，単純 GRAPE を用いた場合より未知環境での性能が高い自律エージェントが獲得できると期待する．**図 2** に本手法の構造を示す．

3.1 基本環境の設計

本手法ではまず，エージェントが学習すべき基本環境の設計を行う．この基本環境はエージェントの目的に応じて設計する必要がある．本論文では環境内を探索するエージェントを想定し，基本環境として「広い環境 room_0」，「通路 room_1」，「障害物の多い環境 room_2」，「部屋の出入りを要する環境 room_3」を用意した．広い環境では周囲に壁や障害物のない方向へ一度に長い距離を移動することが有効であると考えられ，壁際に沿った動きは有効ではない．一方通路のような部屋では，常に周囲に壁のある状態で探索をする必要があるため広い環境で得られた行動は有効ではない．このような部屋では，壁に沿って動くことが有効であると考えられる．これら 2 つの環境は特定の箇所で方向転換ができれば良いが，障害物の多い環境では頻繁な方向転換が必要とされる．また，部屋の出入りが必要となる環境では，今いる環境の中から出入り口を見つけ出し，出入りするという動作が必要となる．壁伝いの行動では出入り口の場所が特定できてもそこへ出入りすることは困難であると考えられる．**図 3** に本手法で用意した基本環境を示す．

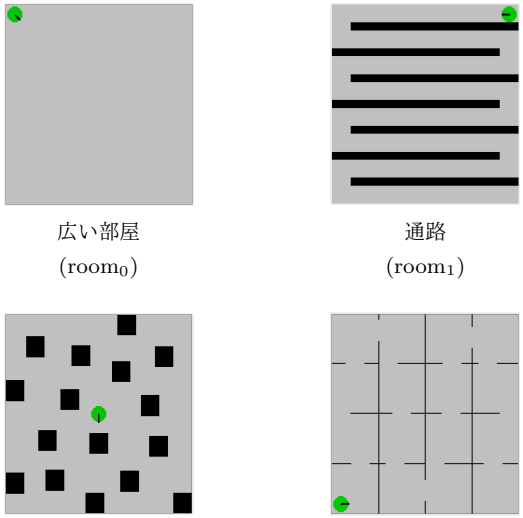

図 3　基本環境

Fig. 3　Primitive environments.

3.2 モジュールの最適化

続いて，各基本環境を学習させ行動規則のモジュールを生成する．前節で示した $\mathrm{room}_0 \sim \mathrm{room}_3$ のそれぞれに対して GRAPE を用いて行動規則を最適化させることによって，各基本環境に特化した行動規則を構築してゆく．しかしながら，学習の結果，行動規則が類似したモジュールが生成されたり，1 つのモジュールで 2 つ以上の行動に対応できるモジュールが獲得されたりすることも考えられる．たとえば，広い部屋で学習させたモジュールでも，壁が近くにあるときには壁を伝うような動きをすることも十分考えうる．このような場合を想定し，新たなモジュールの生成前に今まで生成した全モジュールを新たな学習対象である基本環境に対して適用し，一定の性能が得られなかった場合のみその基本環境での学習を行うこととした．これによって，不必要なモジュールが生成されることを避けるこ

© 2016 Information Processing Society of Japan

とができると考えられる．

3.3 モジュール切替機構の設計

最後に，モジュール切替機構であるスイッチャについて説明する．このスイッチャは，周囲の環境やエージェントの状態に合わせて，有効に働くモジュールを使い続け，使用しているモジュールが周囲の環境に合わず性能が低下したと判断された場合，新たに有効に働くと考えられるモジュールに切り替えるよう設計する．

本手法では有効なモジュールを使い続けるために，モジュールの使用制限時間をスイッチャに設ける．基本的に毎フレームに制限時間を 1 ずつ減らしてゆくが，前進または後退操作が選択されているにも関わらず移動距離が 0 だった場合，現在のモジュールが有効に働いていないと考えられるため，迅速にモジュールを切り替える必要がある．このとき，単に壁にぶつかったために移動距離が 0 であっただけで，使っているモジュールがまだ有効である可能性もあるため，移動距離が 0 である時間が続くほど制限時間の減少量を徐々に増やすよう設計した．これによって，移動距離が 0 であった時間が微小である場合は制限時間の減少を抑えることができる．制限時間の増加条件はエージェントが未探索領域を探索した場合で，その面積分 ΔS だけ増加させる．たとえば 1 フレームで新たに 20[pixel] 探索できた場合，現在使用しているモジュールの使用時間を 20 増加させる．これによって，有効に探索に働くモジュールをより長い時間使用することができると考えられる．

制限時間が 0 以下になった場合，別のモジュールに切り替える．このときモジュール群の中から周囲の環境に即したモジュールを選択する必要があるが，これを実現するために本手法ではエージェントに搭載されている距離センサの平均発火時間 $\boldsymbol{l}^t$ を利用する．距離センサの平均発火時間は [0,1] の実数値を取り，搭載されている距離センサと同数の次元を持つベクトルで表現される．エージェントが距離センサ S0 ～ Sn を持つときの t フレーム前までの $\boldsymbol{l}^t$ は式 (1) で計算される．なお，$T^t_{\mathrm{S}k}$ は直近の t フレーム内で障害物が距離センサ Sk の検出範囲内に入っていたフレーム数を表す．

$$\begin{aligned} \boldsymbol{l}^t &= (l^t_{\mathrm{S}0}, l^t_{\mathrm{S}1}, \cdots, l^t_{\mathrm{S}k}, \cdots, l^t_{\mathrm{S}n}) \\ l^t{}_{\mathrm{S}k} &= T^t_{\mathrm{S}k}/t \end{aligned} \tag{1}$$

まず準備段階として，モジュールの学習が済んだ後，得られたモジュールを基本環境全てに適用し，モジュール m を基本環境 room_i に適用した時の距離センサの平均発火時間 $\boldsymbol{l}^{maxstep}$ を,$\boldsymbol{L}_{m,\mathrm{room}_i}$ としてテーブルに保存しておく．$maxstep$ はその環境での実行ステップ数を指す．未知環境で使用しているモジュールが m であった場合，学習終了時に計測した $\boldsymbol{L}_{m,\mathrm{room}_i}$ と未知環境で計測した $\boldsymbol{l}^{100}$ との類似度 $sim_{\mathrm{room_i}}$ を式 (2) で計算し，現在使っているモジュール m 以外の中から，類似度の高かった部屋で学習したモジュールに切り替える．なお，sim は値が 0 に近いほど類似度が高いとみなす．

Algorithm 1: モジュール切替 (スイッチャ)

```
if モジュール m の制限時間 ≤ 0 then
    if 制限時間が 0 になる前までにモジュール m で探索できた then
        for i ≤ 基本環境数 do
            式 (2) で類似度を計算
        式 (3) で制限時間を計算類似度の高い部屋で学習した
        m 以外のモジュールに切り替える
    else
        /* モジュール m でうまく探索できなかった場合  */
        次に類似度の高い部屋で学習したモジュールに
        切り替える
        式 (3) で制限時間を計算
        (すべてのモジュールを試した場合は類似度を再計算)
```

$$sim_{\mathrm{room_i}} = ||\boldsymbol{L}_{m,\mathrm{room}_i} - \boldsymbol{l}^{100}|| \tag{2}$$

切り替え先のモジュールでの使用制限時間の初期値と上限値は，切り替え時に算出した類似度を元に式 (3) で決定する．

$$timelimit = 100 \times (1 + \sqrt{n} - sim) \tag{3}$$

sim が小さいほど選択したモジュールがうまく働くと推測されるため，そのような場合ほど制限時間が長くなるよう設定した．類似度は距離センサの個数が n 個の場合 $[0,\sqrt{n}]$ の実数値を取りうるので，どんなに sim が大きくても制限時間が 100 未満にならないようにした．また，切り替え先のモジュールがうまく機能しなかった場合は次に類似度の高かったものを順に選択してゆき，すべてのモジュールが使用された場合は環境の類似度が再計算される．**Algorithm 1** にモジュール切替のアルゴリズムを簡単に示す．

4. 探索エージェントの構築実験

本手法による探索エージェントの性能の確認のため，掃除ロボットを想定した部屋の探索エージェントの構築実験を行った．本実験で部屋の全探索問題を採用した理由として，部屋の障害物の配置を変えることで部屋の性質を容易に変えることが可能であることが挙げられる．自律エージェントによる探索は**図 4** に示すようなシミュレータ上で行う．部屋は矩形状であり，灰色の箇所は未探索領域 (未清掃領域)，白色の箇所は探索済みの領域 (清掃済み領域)，黒色の箇所は障害物を表しており，灰色の未探索領域を出来るだけ少なくすることがエージェントの目的である．学習に用いた基本環境は図 3 に示す通りで，大きさは 200 × 200[pixel] である．また，エージェントが動いた軌跡は点線で表している．

© 2016 Information Processing Society of Japan

Vol.2016-MPS-108 No.3
Vol.2016-BIO-46 No.3
2016/7/4

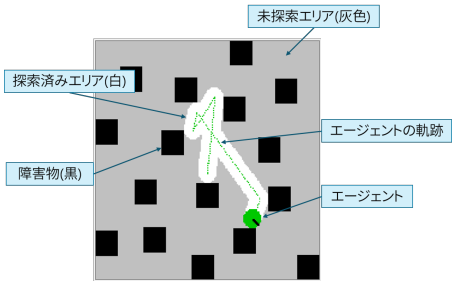

図 4　シミュレーション環境
Fig. 4　A Simulation environment.

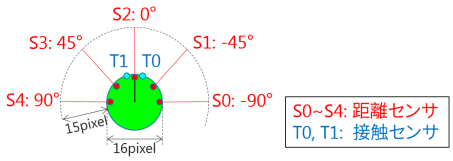

図 5　エージェントの設定
Fig. 5　Configuration of the exploration agent.

表 1　条件ノードの設定
Table 1　Configuration of judge (non-terminal) nodes.

記号	判定内容	分岐数
IR0	距離センサ S0 の値	3
IR1	距離センサ S1 の値	3
IR2	距離センサ S2 の値	3
IR3	距離センサ S3 の値	3
IR4	距離センサ S4 の値	3
TS0	接触センサ T0 の発火	2
TS1	接触センサ T1 の発火	2
D_S	ゴミセンサの発火	2

4.1　自律エージェントの設定

本実験で用いるエージェントは左右に車輪を備えた移動ロボットを想定している．エージェントには 5 つの距離センサ，2 つの接触センサ，そして前方に未探索エリア (ゴミ) があるかを検知するゴミセンサが搭載されている．距離センサは検出範囲内にある障害物の距離を測ることができるものとする．**図 5** にエージェントのセンサに関する設定を示す．エージェントの取りうる行動は前進・後退・右旋回・左旋回の 4 種類で，前進速度は 2[pixel/frame]，後退速度は 1.5[pixel/frame]，旋回速度は 0.125[rad/frame](約 7°) に設定されている．

4.2　GRAPE の設定

エージェントの行動規則を獲得する上で，GRAPE の各種ノードを設計した．中間ノードはエージェントの各センサに対応したものを用意した．**表 1** に条件ノードの設定について示す．また，出力ノードはエージェントの行動である前進 (AHEAD)，右旋回 (RIGHT)，左旋回 (LEFT)，後退 (BACK) と対応させた．

また，GRAPE に用いられている各種パラメータについて**表 2** に示す．学習時の実行ステップ数は 5000[frame] として，モジュールの進化に用いた個体の適応度は探索率 (ゴミの収集率) を元に算出し，探索率が 9 割を超えた個体に対しては，9 割探索するまでのステップ数が少ないほど適応度が高くなるようにした．式 (4) に適応度の算出式を示す．

表 2　GRAPE のパラメータ
Table 2　GRAPE Parameters for evoluting behavior rules.

パラメータ	値
各基本環境での世代数	20,000
世代交代モデル	MGG*
親個体数	50
子個体数	15
トーナメントサイズ	2
最大ノード数	20
エリート数	1
交叉確率	0.7
一様交叉確率	0.1
突然変異確率	0.04
最大ノードステップ数	100

(*Minimal Generation Gap[8])

$$fitness = \begin{cases} \frac{S}{S_{\mathrm{room}}} + \alpha \cdot (step_{0.9})^{-1} & (if \quad \frac{S}{S_{\mathrm{room}}} \geq 0.9) \\ \frac{S}{S_{\mathrm{room}}} & (otherwise) \end{cases} \tag{4}$$

S_{room} は部屋の面積，S は探索済みの面積，$step_{0.9}$ は部屋を 9 割探索するまでにかかったステップ数を表し，$\alpha = 10$ とした．1 つの部屋につき 3 種類のスタート地点を設定した．各スタート地点から探索を行い，各適応度の平均値を最終的な個体の適応度とする．

4.3　未知環境の設定

本実験では 300 × 300[pixel] の部屋を未知環境として用意する．100 × 100[pixel] の環境を 1 ブロックと定義し，それを縦横 3 × 3 ブロックに連結して未知環境を構築する．ブロック内に設置される障害物は以下の 6 種類の中からランダムで選択される．

(1) 直径 20[pixel] の円形物体 (最大 3 つをランダムに配置)
(2) 10 × 50[pixel] の矩形物体 (最大 3 つをランダムに配置)
(3) 50 × 10[pixel] の矩形物体 (最大 3 つをランダムに配置)
(4) 右回りの螺旋状壁 (通路幅 20pixel)
(5) 左回りの螺旋状壁 (通路幅 20pixel)
(6) 障害物なし

ブロック間にはランダムで壁を設置し，隣接するブロックへ移動するための出入り口がランダムで設置される．エージェントのスタート位置と角度もランダムに決定されるが，スタート位置周辺に障害物がある場合はエージェントが移動できるよう障害物を変形するものとする．なお，

© 2016 Information Processing Society of Japan

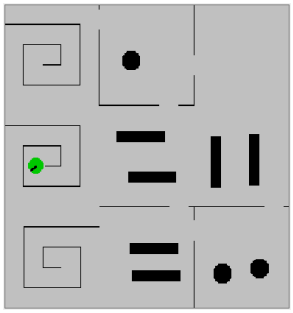

図 6　未知環境の例
Fig. 6　An example of unknown environments.

未知環境での実行ステップ数は 18000[frame] とした．**図 6** に未知環境の例を示す．

4.4　実験結果

提案手法によって探索エージェントを構築し，部屋の全探索問題に対して適用する．未知環境は全部で 10 部屋用意し，個体を 10 部屋に対して稼働させたときの平均探索率を用いて評価を行う．平均探索率 $eval$ は式 (5) によって算出される．なお，$S_{\mathrm{testroom_i}}$ は未知環境 i の面積，S_i は個体を未知環境 i に適用したときの探索率を表す．

$$eval = \sum_{i=0}^{9} \frac{S_i}{S_{\mathrm{testroom}_i}} \tag{5}$$

比較手法はモジュール化しない単純 GRAPE と，壁にぶつかるとランダムに角度を変え前進し続けるランダムサーチの 2 種類とした．単純 GRAPE のノード数はモジュール 4 つ分の 80 で，これは提案手法の最大のモジュール数と等しい．また進化の世代数も，提案手法での世代数の合計と等しく 80,000 世代に設定した．**表 3** に 10 試行中の平均探索率と最良探索率を示す．単純 GRAPE が未知環境で平均して 2 割 ~3 割ほどの探索率しか得られなかったのに対し，提案手法では 6 割近い探索率が得られた．**図 7** に提案手法で構築された最良個体の未知環境での探索結果を示す．エージェントやその軌道の色は，使用されたモジュールと対応しており，緑は広い部屋，青は通路，赤は物体の多い部屋，黄は部屋の出入りのある部屋で学習したモジュールを表している．全体を通して，物体の多い部屋で学習したモジュールが中心に使われているが，障害物の無いブロックでは広い部屋で学習したモジュールに切り替えることで探索が促進されている (**図 8**)．一方，学習環境には無かった螺旋状に壁が配置されている箇所では，通路状の環境で学習させたモジュールが有効に働いており，螺旋中心部での方向転換にも対処できていることが分かる (**図 9**)．また，通路で学習させたモジュールでは壁によって仕切られたブロック間の移動が満足に出来なかったが，部屋の出入りを学習させたモジュールに切り替えることで対処することができた (**図 10**)．単純 GRAPE での探索率が大幅に低下した要因として，単純 GRAPE の場合は性質の全く異なる環境を同時に学習させると，充分な世代数を設定してもうまく最適化ができず，多様性のある行動規則を獲得できなためであると考えられる．そのため，未知環境において偏った箇所のみしか探索できなかったケースが多く見受けられた．一方本手法の場合は，環境ごとに行動規則を別々に進化させることでその環境に特化したモジュールを容易に最適化でき，また未知環境において性質の異なる行動規則のモジュールを切り替えることでより多彩な行動が表現できたため，未学習である未知環境においても探索エージェントの性能を維持することができたと考えられる．

表 3　未知環境での探索率
Table 3　Exploration rate in unknown environments

	10 試行平均	(± 標準偏差)	10 試行最良
提案手法	**0.5880**	(± 0.1111)	**0.7667**
単純 GRAPE	0.2441	(± 0.1368)	0.4691
ランダムサーチ	0.5160	(± 0.03377)	0.5690

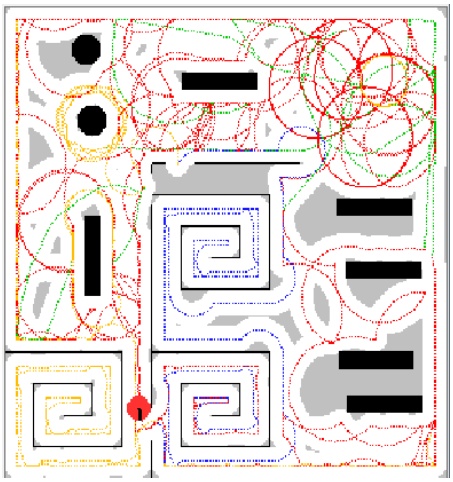

図 7　最良個体の未知環境での探索結果の例 (探索率: 0.8998)
(緑:広い部屋，青:通路，赤:物体の多い部屋，
黄:部屋の出入りのある部屋)
Fig. 7　An example of the best individual in an unknown environment.(exploration rate: 0.8998)
(green:broad room, blue: passage, red: lots of obstacles, yellow: doorways)

5.　まとめ

本論文では進化計算を用いたプログラムの最適化手法によって並列的なモジュールを生成し，機能の切り替えによって未知環境に対する適応性を高めた探索エージェントを構築した．互いに性質の異なる基本環境を用意し，それぞれに対して行動規則を最適化することによって，特定の行動に特化したモジュールを構築した．モジュールの切り替えでは，周囲の環境と学習時での環境の類似度を測り，類似度の近かった基本環境で学習したモジュールを選択することで，有効に働くと思われるモジュールを選択す

© 2016 Information Processing Society of Japan

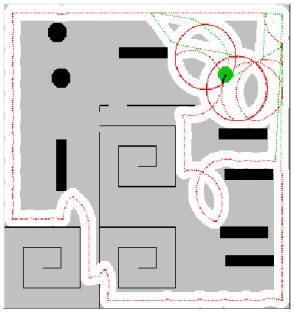

図 8　広い部屋で学習したモジュールが有効であった例

Fig. 8　An example that the module of a broad room was effective.

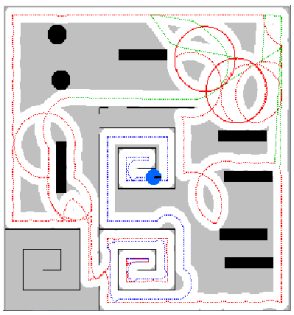

図 9　通路で学習したモジュールが有効であった例

Fig. 9　An example that the module of a passage was effective.

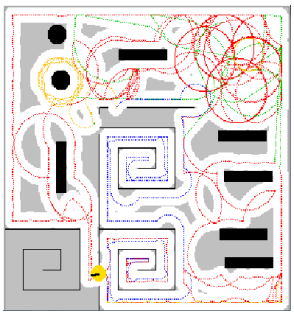

図 10　部屋の出入りのある部屋で学習したモジュールが有効であった例

Fig. 10　An example that the module of doorways was effective.

ることとした．本論文では，グラフ構造状プログラムの自動生成手法である GRAPE を用いて探索エージェントの行動規則を構築し部屋の全探索問題に適用した．実験の結果から，提案手法によって構築した探索エージェントがモジュールを切り替えて探索することで，モジュール化しない単純 GRAPE に比べて，未知環境において良好な探索率が得られたことを示した．

今後の課題として，スイッチャを含めて最適化できる学習手法の検討，どのモジュールも有効では無かった場合の追加学習機構の実装，および多数の基本環境の中から必要十分な基本環境のみを選択して学習する手法の検討が挙げられる．

参考文献

[1] Koza, J. R.: *Genetic programming: On the programming of computers by means of natural selection*, MIT Press (1992).
[2] Gustafson, S. M. and Hsu, W. H.: *Layered learning in genetic programming for a cooperative robot soccer problem*, Springer (2001).
[3] Hsu, W. H. and Gustafson, S. M.: Genetic Programming And Multi-agent Layered Learning By Reinforcements., *Genetic and Evolutionary Computation Conference*, pp. 764–771 (2002).
[4] Stone, P. and Veloso, M.: Layered learning, *Machine Learning: ECML 2000*, Springer, pp. 369–381 (2000).
[5] Jackson, D.: Hierarchical genetic programming based on test input subsets, *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, pp. 1612–1619 (2007).
[6] Koza, J. R.: *Genetic programming II: Automatic discovery of reusable programs*, MIT Press (1994).
[7] Shirakawa, S., Ogino, S. and Nagao, T.: Graph structured program evolution, *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, ACM, pp. 1686–1693 (2007).
[8] 佐藤浩，小野功，小林重信：遺伝的アルゴリズムにおける世代交代モデルの提案と評価，人工知能学会誌，Vol. 12, No. 5, pp. 734–744 (1997).
[9] Shirakawa, S. and Nagao, T.: Evolution of sorting algorithm using graph structured program evolution, *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, pp. 1256–1261 (2007).
[10] 近藤義隆，長尾智晴：Graph Structured Program Evolution を用いたマルチエージェント制御における協調行動の自動獲得，第 28 回ファジィシステムシンポジウム，pp. 1014–1017 (2012).

© 2016 Information Processing Society of Japan