3G2-01

遺伝的プログラミングを用いた 自律移動ロボット制御プログラムの生成

Generating autonomous mobile robot control program using genetic programming

白山　政敏[*1]　　越野　亮[*1]
Masatoshi Shirayama　Makoto Koshino

[*1]石川工業高等専門学校
Ishikawa National College of Technology

This paper describes an evolutional computation which gives an autonomous mobile robot skills of obstacle avoidance and searching for the shortest route in the path planning problem. Finding the shortest route after numbers of the generation iteration, the robot reached the goal from the starting point traveling through 24 different sized boxes, placed randomly in the simulator map. The genetic programming (GP) we applied to the robot control is an optimum algorithm which evolves the tree structure, while the genetic algorithm (GA) evolves a genotype of each individual by operating crossover and mutation.

1. はじめに

近年，サッカーやレスキュー活動を対象としたロボット大会や，人間と同じ形態のヒューマノイド開発が盛んになり，ロボット開発の勢いは加速を続ける一方である．しかしながら，ハードウェアやアプリケーションの面としては，飛躍的に進歩を続けているが，知的な情報処理の側面ではあまり進歩が見られていない．その原因の一つとして，リアルタイムに行動を制御し続けるための行動規則を明確に記述することが難しいという点があげられる．

自律移動ロボットの制御プログラムは，複雑で膨大な行動規則になり，様々な問題に対して頑強（ロバスト）なプログラムを人手によって書くのは容易ではない．そこで，環境から情報を学習し，自律的に障害物回避が遂行できるプログラムを生成する手法が求められている．

Koza は，Brooks らの自律移動ロボットの枠組みである包摂アーキテクチャ（Subsumption Achitecutre）型ロボット [1, 2] に，遺伝的プログラミング（GP：Genetic Programming）[3] を適用し，壁に沿って行動するプログラムの生成を行った [4]．実機のロボットへの GP を利用した研究として，Reynolds はロバストな制御を行うために，センサおよびアクチュエータにノイズを導入し，障害物を回避するプログラムを生成している [5]．Ito らはプログラムの冗長性が頑強性の実現に不可欠であることから，Prog などの関数を用いて適切に非終端記号を設計することで GP の生成するプログラムに冗長性が導入することを実験的に確かめている [6]．Nordin らは，赤外線近接センサを利用したミニチュアロボット（Khepera：ケペラ）の制御方式に GP を用いている [7]．

GP を用いた移動ロボットの制御プログラムを生成する研究において，実機を使った研究はまだあまり行われていないのが現状であり，移動ロボットの視覚カメラからのから見える状況を想定した研究は未だなされていない．本研究では，従来の赤外線センサを利用したミニチュアロボットではなく．実際の無人搬送車や自動走行車椅子を想定し，支柱の高い位置に取り付けられたカメラからの画像に基づき，障害物を探知し，GP により制御プログラムを生成する．また，個体の多様性を維持させるために，遺伝的アルゴリズム（GA：Genetic Algorithm）[8] で用いられている割り当て関数を GP に導入し，考察を行う．最後に，シミュレーション結果から割り当て関数を用いた提案手法の有効性を示す．

連絡先: 白山　政敏，石川工業高等専門学校，石川県河北郡津幡町字北中条タ 1，TEL:076-288-8131，FAX:076-288-8146，e-mail:shira@shikawa-nct.ac.jp

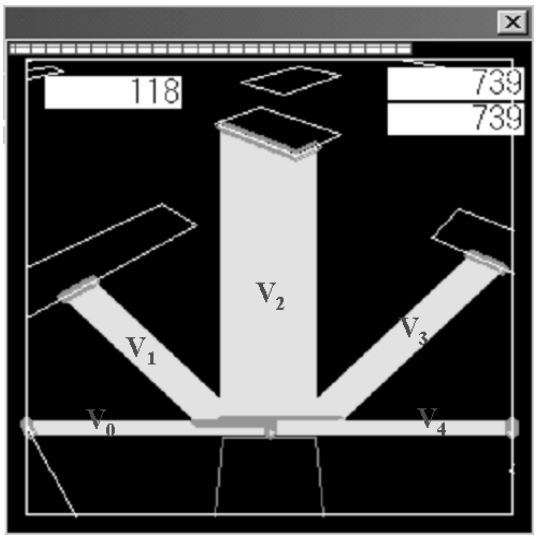

図 1: ロボットのカメラからの画像

2. 提案手法

2.1 移動ロボットシミュレータ

移動ロボットには，支柱の高い位置に CCD カメラが取り付けてあり，5 方向の幅のある帯状画素ラインにより障害物を感知することができる（図 1）．それぞれの方向 (V_0〜V_4) に障害物が近ければ近いほど小さな値を返すものとする．

2.2 GP へのロボット動作関数の導入

遺伝的プログラミング（GP) は遺伝子型の遺伝的アルゴリズム（GA）を拡張し，木構造を扱えるようにした個体を用いる．この木構造は，ノードで構成され，非終端ノードに関数が割り当てられ，終端ノードに関数に取り込まれる変数あるいは定数が割り当てられる．本手法で用いた木を構成するノードを

表 1: 木の構成要素

名称	種別	処理内容
Move	非終端	引数で与えられた距離だけ移動 衝突した場合は一定距離だけ後退
Handle	非終端	操舵角を引数の値だけ変更
Movewj	非終端	衝突時に後退し逆ハンドルを切る
Handlewj	非終端	ゴールの方向を取得し 進行方向を修正
IFILE	非終端	if (第 1 引数 < 第 2 引数) then 第 3 引数を返す else 第 4 引数を返す
Prog2	非終端	引数 2 個を順次実行
Prog3	非終端	引数 3 個を順次実行
$v_{0\sim4}$	終端	5 方向障害物センサ測定距離

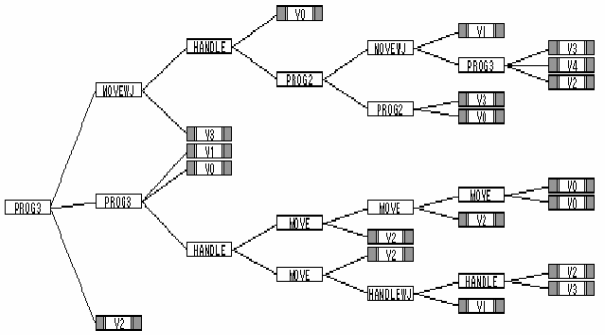

図 2: 遺伝的プログラミングによる木構造の一例

表 1 に示す．従来の Khepera ロボットで用いられた操作関数は Move，IFLTE，prog2 である [9] が，本手法では狭隘な道を通過でき，また行き止まり通路で前進・後退を繰り返すことで反転できるような Movewj 関数，5 方向センサ値 (V_0〜V_4) から障害物が少ない方向へハンドル角を決め，かつ，ゴールに常に向かうように進行方向を修正する Handlewj 関数の 2 つを加えた．

本手法の遺伝的プログラミングによる木構造の一例を図 2 に示す．また，この図における LISP の S 式は以下のように表される．

(Prog3 v2 (Prog3 (Handle (Move (Handlewj v1 (Handle v3 v2)) v2) (Move v2 (Move v2 (Move v0 v0)))) v0 v1) (Movewj v3 (Handle (Prog2 (Prog2 v0 v3) (Movewj (Prog3 v2 v4 v3) v1)) v0)))

2.3 遺伝的プログラミングの処理手順

遺伝的プログラミングの処理手順を以下に示す．

Step1. 乱数を使って初期集団を生成する．

Step2. 各個体の適合度を評価する．

Step3. ルールに基づき個体を選択・淘汰する．

Step4. 遺伝子列の交叉・突然変異を行う．

Step5. 終了条件を満たさなければ *Step*2 へ戻る．

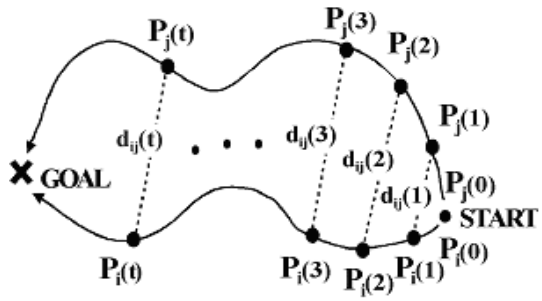

図 3: 個体間のユークリッド距離

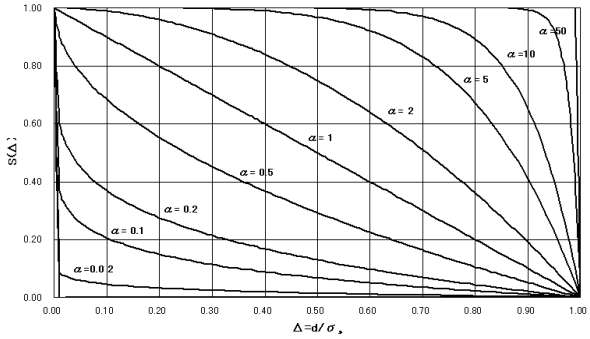

図 4: 割り当て関数 $S(d)$

2.4 多様性維持のための適合度の分割の提案

GP では個体の多様性を維持しながら進化をすることが求められる．そこで，ロボットの移動軌跡の各時点でのユークリッド距離を求め，この値が大きい程，個体間に差があるとする．多様性維持のため個体の適合度 $f(p_i)$ を割り当てに応じて分割する式 f_s を図 3 に示す個体間のユークリッド距離の概念図を用いて定義する．

定義 1 (個体間のユークリッド距離) 時刻 0〜t における 2 つのロボットの経路を $p_i(0)\cdots p_i(t)$，$p_j(0)\cdots p_j(t)$ とする．

ある時刻 t におけるロボットの位置座標 $p_i(t)$ と $p_j(t)$ のユークリッド距離 $d_{i,j}(t)$ は以下の式で表される．

$$d_{ij}(t) = ||p_i(t) - p_j(t)|| \tag{1}$$

定義 2 (割り当て関数) 個体間の距離を d，d の標準偏差を σ_s，*Goldberg の adjustable factor[8]* を α とする．

割り当て関数 $S(d)$ は以下の式で表される（図 *4* 参照*)*.

$$S(d) = 1 - \left(\frac{d}{\sigma_s}\right)^{\alpha} \tag{2}$$

定義 3 (適合度の分割) 割り当て関数を $S(d)$，適合度関数を $f(p_i)$，個体数を n とする．適合度の分割式 f_s を以下の式で定義する．

$$f_s(p_i) = \frac{f(p_i)}{\sum_{j}^{n} S(||p_i(t) - p_j(t)||)} \tag{3}$$

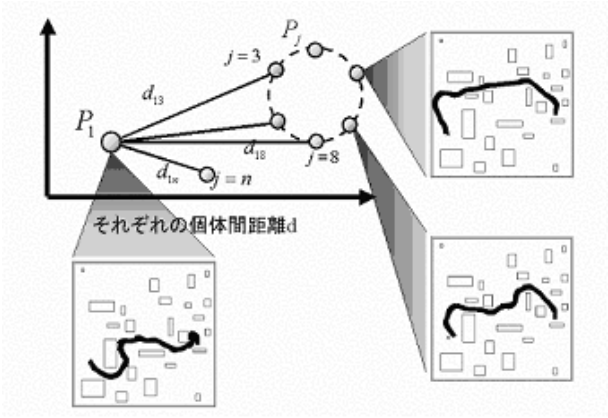

図 5: 個体の多様性維持

この式から，多くの個体が近くにある個体ほど分割の度合いが減ることになるため，集団内での特定の種の増長を防ぐことができる．

定義 4 (適合度関数) ロボットの衝突回数を $n_c(p_i)$，ロボットの軌跡とゴール地点の最接近距離を $r_{min}(p_i)$，ロボットの総移動距離を $l(p_i)$ とする．各個体間の適合度関数 $f(p_i)$ を以下の式で表す．

$$f(p_i) = \begin{cases} 10n_c(p_i) + l(p_i), \\ \quad \text{(ゴールに到着した場合)} \\ 10n_c(p_i) + 16r_{min}(p_i) + l(p_i), \\ \quad \text{(ゴールに到着しなかった場合)} \end{cases} \tag{4}$$

図 5 は，ユークリッド距離の違いと軌跡の関係を示している．図に示すように，右のユークリッド距離が近い集団より，左のユークリッド距離が大きい個体のほうが，より違う軌跡を通ってゴールにたどり着いていることがわかる．

3. 実験

本実験で用いた GP のパラメータを表 2 に示す．

表 2: GP のパラメータ

パラメータ名	設定
個体数	100
木の最大深さ	初期：6，交叉：17，突然変異：10
交叉確率	0.69(関数ノード:0.495, 任意ノード:0.195)
突然変異確率	0.215
選択方式	トーナメント方式（サイズ 4）

3.1 実験 1：割り当て関数の有効性の検証

本提案手法である割り当て関数の有効性を検証するために，世代数（0〜500）に対して平均適合度と最良適合度の推移を調べた．実験結果を図 6 に示す．ただし，α=0.1 とした．この結果から，割り当て関数を使用していない場合，40 世代以降で平均適合度が大きく下がり最良適合度に近づき，多様性が失われたのに対し，使用した場合は，500 世代でも平均適合度が下がることなく，多様性が維持できていることがわかる．

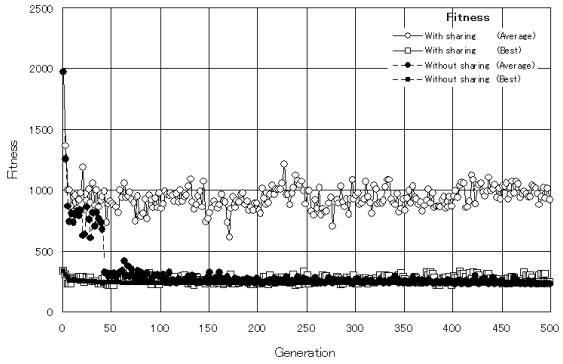

図 6: 世代数に対する適合度

参考のために，割り当て関数を使用した場合としない場合のロボットの軌跡を図 7 に示す．この結果から，割り当て関数を使用したほうが，衝突回数も少なく，距離もより短い経路を通ることががわかる．

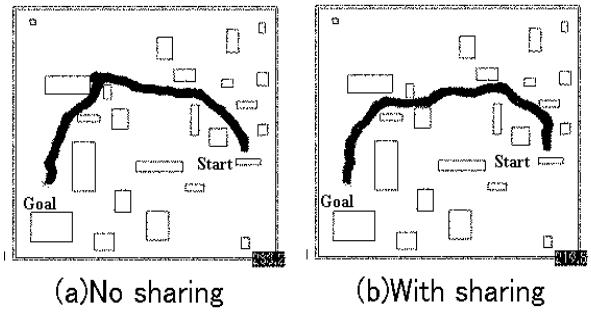

図 7: 割り当て関数を使用した場合と使用しない場合の軌跡の比較

次に，割り当て関数を使用しない場合の最良軌跡の改善の様子を図 8，割り当て関数を使用した場合の最良軌跡の改善の様子を図 9 に示す．この結果から，割り当て関数を使用して，多様性を維持したほうが遥かに早く最良個体の改善が行われていることがわかる．

3.2 実験 2：割り当て係数 α の影響

割り当て関数 $S(d)$ の係数 α（図 4 参照）の値による適合度の影響を調べる．図 10 に世代数に対する割り当て係数 $\alpha = 0.01$ と $\alpha = 16$ の場合の適合度を示す．この結果から，常に $\alpha = 16$ において $\alpha = 0.01$ より良い適合度を保っていることがわかる．参考のために，割り当て係数 $\alpha = 0.01$ と $\alpha = 16$ の最良軌跡を図 12 に示す．

図 4 に示されるように，α を変化させることで，割り当て関数は大きく変わることになる．α の値によっては多様性がほとんど見られないことが考えられるため，良い α を求める基準が必要となる．

3.3 実験 3：ロボット制御プログラム

遺伝的プログラムで獲得した木構造の特性がロボットの制御動作に大きく影響する．このため学習と同一の環境地図を使う

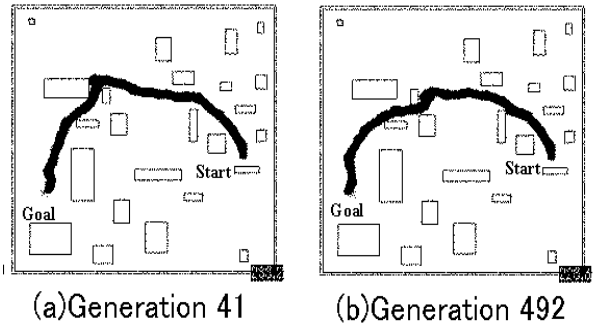

図 8: 割り当て関数を使用しない場合の最良軌跡の改善

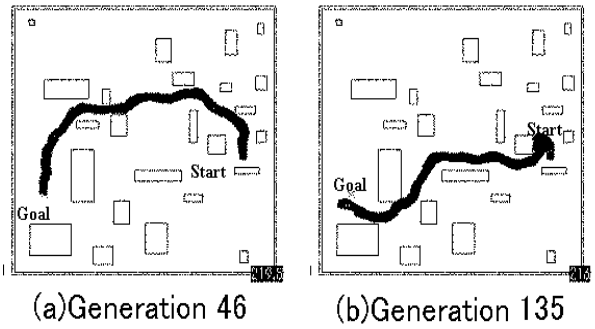

図 9: 割り当て関数を使用した場合の最良軌跡の改善

エミュレータを作成し，ロボットの制御動作を検証した．

実験 1 の割り当て関数を使用した場合の木構造 (図 9) をエミュレータに搭載し，ロボットを走行させた結果を図 12 に示す．

4. まとめ

本論文では，割り当て関数を使用し，多様性を維持することができる遺伝的プログラミングの手法を提案した．今後は，本手法を実機ロボットへ導入する予定である．

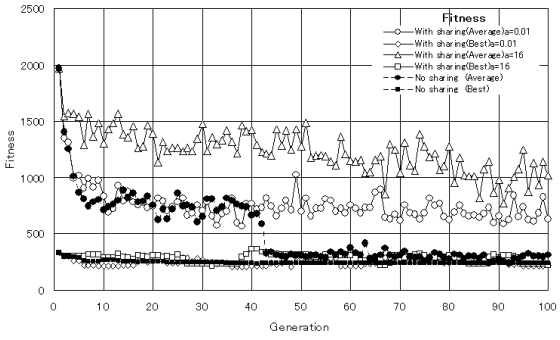

図 10: 世代数に対する割り当て係数 ($\alpha = 0.01$, $\alpha = 16$) の適合度

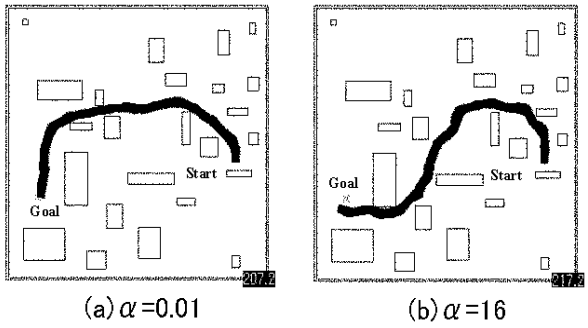

図 11: 割り当て係数 $\alpha = 0.01$ と $\alpha = 16$ の最良軌跡の比較

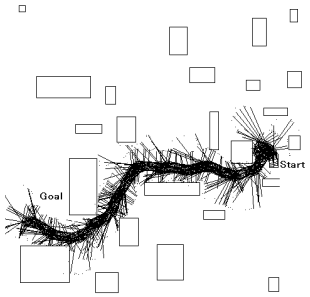

図 12: ロボット制御プログラムの走行結果

参考文献

[1] R.A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, RA-2(1), pp.14–23, 1986.

[2] R.A. Brooks, New approaches to robotics, Science, vol.253, pp.1227–1232, 1991.

[3] J.Koza, Genetic programming, MIT Press, 1992.

[4] J.Koza, Evolution of a subsumption architecture that performs a wall following task for an autonomous mobile robot via genetic programming, in Computational Learning Theory and Natural Learning Systems, vol.2, T.Petsche (ed.), MIT Press, 1994.

[5] C.Reynolds, Evolution of obstacle avoidance behavior : using noise to promote robust solutions, in Advances in Genetic Programming, K.Kinnear (ed.), MIT Press, 1994.

[6] T.Ito, H.Iba and M.Kimura, Robustness of robot programs generated by genetic programming, in Genetic Programming (GP96), MIT Press, 1996.

[7] P.Nordin, W.Banzhaf and M.Brameier, Evolution of a world model for a miniature robot using genetic programming, Journal of Robotics and Autonomous System, vol.25, pp.105–116, 1998.

[8] D.E.Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Niche Methods for Genetic Search, Addison Wesley, 1989.

[9] 伊庭斉志，遺伝的プログラミング入門，東京大学出版会，2001.