遺伝的アルゴリズムを用いたロボットの経路学習

胡 蘇暁[1]　中山功一[1]

概要：本論文では，障害物との衝突センサのみを装備した掃除ロボットを想定し，室内の状況を把握して効率的な掃除経路を探索するアルゴリズムに取り組んだ．提案したアルゴリズムは，3 つのフェーズからなる．フェーズ１：地図生成では，未知の部屋においても，正しく地図が作成できることを示した．フェーズ２：障害物判定では，障害物が移動しても，おおむね正しく障害物をマッチングできることを示した．フェーズ３：掃除経路探索では，地図情報に基づいて，効率的な経路を探索できることを示した．

キーワード：掃除ロボット，遺伝的アルゴリズム，機械学習，経路探索

1. はじめに

近年，IoT と呼ばれるモノのインターネットが注目されており，住居の知能化として，例えばロボットによる料理や掃除や洗濯などが期待されている．本研究では，掃除をする知的ロボット[1]を想定し，室内の状況を把握して地図を作成し，効率的な掃除経路の学習に取り組む．本論文で提案するロボットの経路学習は，次の 3 つのフェーズからなる．フェーズ１：地図生成，フェーズ２：障害物判定，フェーズ３：掃除経路探索．この 3 つのフェーズを順番に実行することで，従来よりも効率的に経路探索を可能とすることを目指す．

2. プログラム

2.1 シミュレーション環境

コンピュータシミュレーションにより，部屋の中で動作する掃除ロボットに経路学習させる．プログラミング環境は Processing3 で，プログラミング言語は JAVA である．

シミュレータの部屋は，頂点の座標で定義された多角形として構成される．部屋の内部に，こちらも頂点の座標で定義された多角形からなる障害物が設置される．掃除ロボットは，これらの多角形の辺を通過することはできない．

掃除ロボットは円形とし，中心座標と半径で定義される．掃除ロボットには，移動速度，移動方向が設定される．移動ロボットは，衝突センサが設定され，移動方向への障害物の有無を検知できる．衝突センサ以外のセンサは設置されない．掃除ロボットの半径は，5pixel とする．掃除ロボットは，移動方向に対し，2pixel/frame の一定の速度で移動し，衝突時にランダムに選択された方向に移動方向を変更する．移動方向への変更は，1frame で完了する．掃除ロボットの初期位置を座標（0,0），初期角度を横軸の正方向（0 度）とする．シミュレーション環境の例を図 1 に示す．

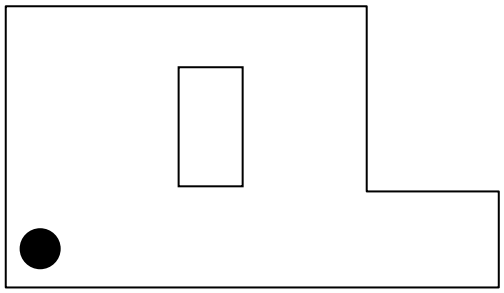


図 1　部屋と掃除ロボットの表示

2.2 移動ロボット

移動ロボットは，自らの移動方向と移動距離を認識し，初期位置を原点とする地図情報を作成する．地図情報は，図 2 に示すようなラスタ形式画像とする．移動できないところを赤で，移動できるところを緑で表示する．緑色で示す移動できるところを，掃除空間と定義する．

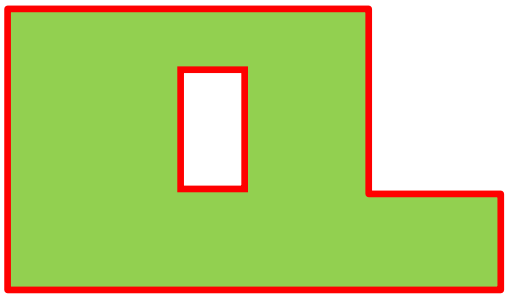


図 2　地図情報の表示

掃除ロボットは，自身のいずれか 1 画素でも，部屋の壁や障害物の輪郭線に重なった場合，衝突センサにより，部屋の壁や障害物に衝突したと判定する．ただし，衝突対象が壁であるか障害物であるかは区別できない．移動時に衝突しなかった領域（ロボットの面積）を掃除空間と判定する．

移動速度が 2pixel/frame であるため，掃除ロボットが壁や障害物に 1pixel 以上重なった場合は，図 3 に示すように，重ならない位置に修正する．

1 佐賀大学

衝突した場合　　位置修正

図 3　衝突時の位置の修正

3.　地図情報の学習（フェーズ１）

本章では，掃除ロボットが掃除するべき部屋の情報を，掃除ロボットが移動しながら学習するアルゴリズムについて述べる．なお，このフェーズでは，掃除ロボットは，掃除するべき空間(掃除空間)を識別することが目的であり，壁と障害物を区別しない．そのため，本章では壁も含めて障害物と表記する．

3.1　障害物と掃除空間の保存形式

掃除ロボットは，自身が通過できた空間を掃除空間と認識し，障害物に衝突した位置を障害物と認識する．この認識結果に基づき，自身の地図情報に更新していく．衝突センサに基づく地図情報の更新手順を以下に示す．なお，シミュレーション環境と掃除ロボットは，以外の衝突センサ以外のデータのやり取りは発生しない．

(1)　掃除空間の学習

掃除ロボットは，図 2 に示す地図情報を，掃除空間か障害物化を画素ごとに認識して更新する．初期状態では，掃除ロボット内の地図情報における全ての x,y 座標において，座標の評価値V(x,y)は 0 で表現されている．掃除ロボットが移動して，ある座標(x,y)を掃除空間と認識したら，その座標の評価値 V(x,y)は以下の更新式に従って更新される．

$$V_t(x,y) = V_{t-1} \times 0.99+1 \qquad (式１)$$

通過した全ての座標について，上記の更新式で評価値Vを更新する．掃除ロボットが通過していない全ての座標の評価値Vは，更新しない．

(2)　障害物の学習

掃除ロボットは，ある座標(x,y)を障害物と認識したら，掃除ロボット内の地図情報の座標(x,y)の評価値 V(x,y)を，以下の更新式に従って更新する．

$$V_t = V_{t-1} \times 0.99-1 \qquad (式２)$$

障害物と認識した全ての座標について，上記の更新式で評価値Vを更新する．掃除ロボットが障害物と認識していない全ての座標の評価値Vは，更新しない．

式１と式２の更新式の結果，評価値 V の上限は 100，下限は-100 となり，一度も判定していない座標は 0 となる．

(3)　地図の表示

全ての座標(x,y)について，ラスタ形式の画像として表示する．図 4 に示すように，評価値 V がプラスなら緑で，マイナスなら赤で表示し，評価値の絶対値の大きさを色の濃さにする．

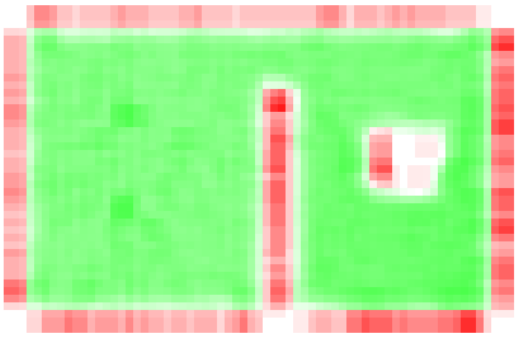


図 4　地図の保存形式

3.2　移動ロボットの移動方法

地図学習における移動ロボットは，以下の 3 つのモードを利用できる．

（A）ランダムモード

障害物に衝突したら，ランダムに決定する角度に回転し，次の障害物に衝突するまで前進する．具体的な流れは以下の通りである．

1. 障害物に衝突するまで，現在の移動方向で移動する．
2. 障害物に衝突したら，回転し，ステップ 1 に戻る．

回転する角度は, 移動方向に対して 120 度から 240 度(移動方向を 180 度反転した角度に対し＋60 度～－60 度)の範囲内で，ランダムに決定する角度である．

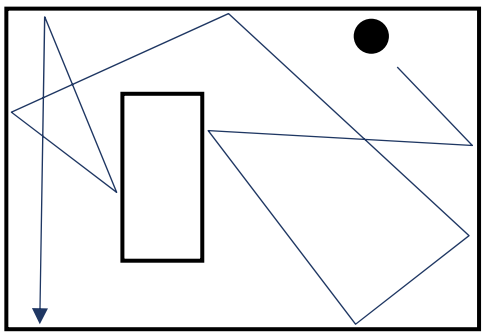


図 5　ランダムモード

（B）沿いモード

衝突した対象物を障害物の位置と形を識別するために，その障害物の輪郭に沿うように移動する．具体的な流れは以下の通りである．

1. 障害物に衝突したら，衝突位置を記録し少し後退する．
2. 反時計回りで 90 度回転する．
3. 前進しながら時計回りで 270 度まで回転する．

4. 障害物に衝突したら，少し後退し，ステップ２に戻る．
5. 最初に記録した位置に戻ったら，沿いモードから脱出．

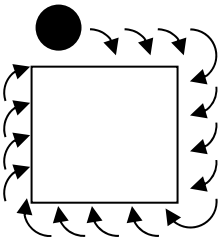


図 6　沿いモード

（C）誘導モード

周辺にある学習程度の低い座標（評価値 V の値が小さい座標）に誘導する．具体的な流れは以下の通りである．

1. 現座標から半径 15pixel の範囲で，評価値 V(x,y)の値が正でかつ最小である座標(x,y)を求める．
2. 求めた座標(x,y)の方向に直進する．
3. 目標に到達したらステップ１から繰り返す．
4. 途中で障害物に衝突したら，沿いモードに変更する．

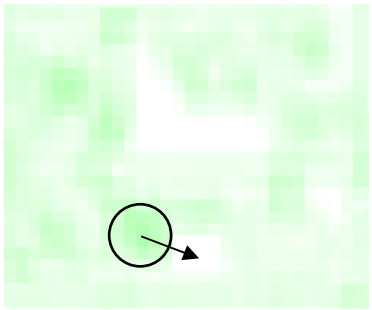


図 7　誘導モード

3.3 ランダムモードと誘導モードの比較実験

同じ部屋配置で二つのモードに対してそれぞれシミュレータで実験した．ここで，地図情報における評価値Ｖの値が１以上の座標の数を被覆面積と定義し，被覆面積の推移を図 8 に示す．ランダムモードは被覆面積が階段状に上昇した一方，誘導モードは段時間でほぼ最大値に至った．

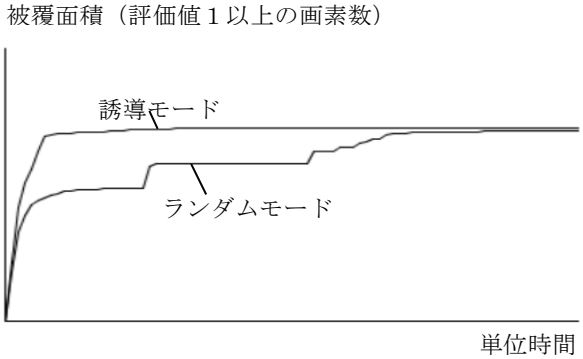


図 8　被覆面積の対比

また，地図上の全ての画素における評価値 V の絶対値の合計を学習の完成度と定義し，学習の完成度の推移を図 9 に示す．完成度の上昇速度は誘導モードの方が速いことから，完成度も誘導モードの方がより高いことが分かる．

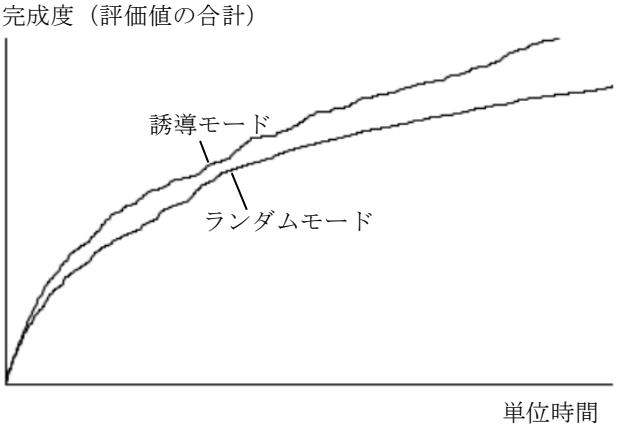


図 9　完成度の対比

4. 障害物の判定（フェーズ２）

本章では，掃除ロボットが掃除するべき部屋にある壁と障害物の形状を認識する．また，障害物が移動した場合を想定して，ある障害物が過去に学習済みの障害物と一致するかを判定するマッチングアルゴリズムを用い，素早く掃除空間を認識することを目指す．

4.1 障害物の形状と位置の検出

フェーズ 1 で学習した壁を含む障害物のラスタ形式の画素をベクトル化し，形と位置を検出する．具体的な流れは以下の通りである．処理の例を図 10 に示す．

1. 緑の画素に隣接する赤い画素を選択する．
2. 選択した赤い画素の隣にほかの赤い画素があったら，両方を線で繋ぎ，ラスタ画像を作成する．
3. 作成したラスタ画像をスムージング処理する．

4.2 壁と障害物の判別

掃除ロボットを囲んだ障害物である壁と，それ以外の障害物である家具などの物体を，図 11 に示すように区別する．

ロボットを囲んだかどうかを判断するアルゴリズムは以下の通りである．

ロボットの位置を中心として横軸と縦軸を引き，横軸のプラス部分とマイナス部分と縦軸のプラス部分とマイナス部分，四つの部分に分ける．判断しようとする障害物の輪郭線が座標の軸の各部分との交差点の数が奇数である場合は，障害物が壁と判別する．一方，交差点の数が偶数である場合は，障害物が室内物体と判別する．

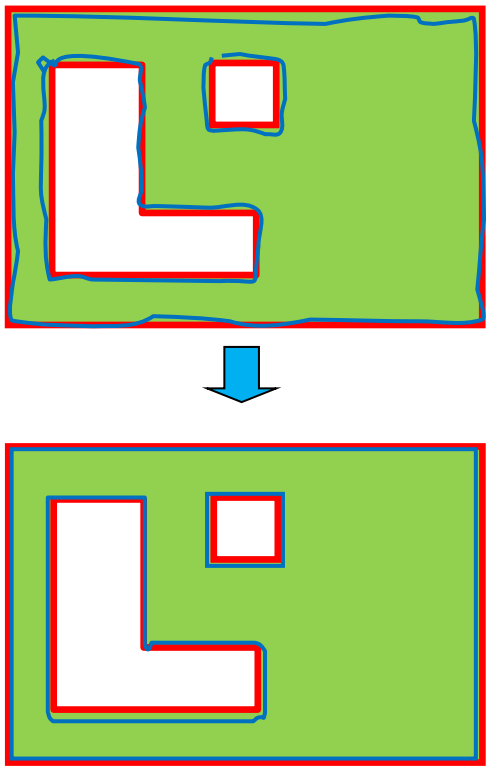


図 10 障害物の検出の例

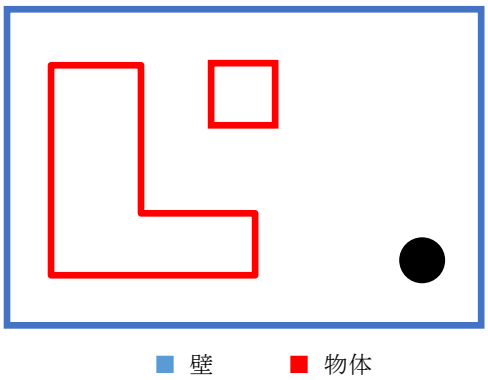


図 11 壁と障害物の判別

4.3 障害物のマッチング

過去に学習した地図情報にある障害物と，今回学習した地図情報にある障害物をマッチングし，同じ障害物が移動されたものであるかを，以下の手順で判別する．

（1）面積マッチング

マッチングする二つの物体が占める面積をそれぞれ算出し，面積の近さにより，同じ物体の可能性があるかを判断する．

（2）形状マッチング

マッチングする二つの物体の形状により，同じ物体かどうかを判断する．形状マッチングアルゴリズムは以下の通りである．処理の例を図 13 に示す．

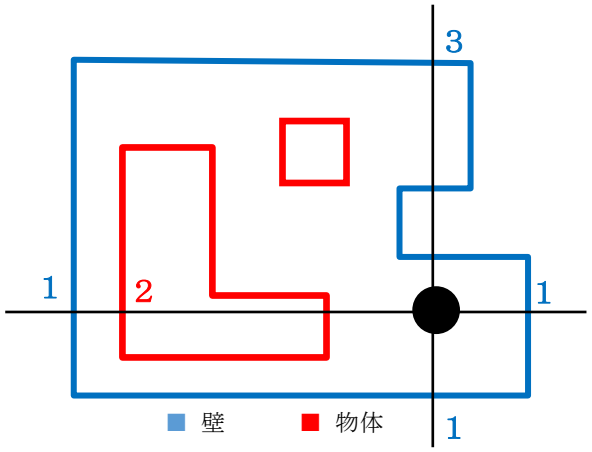


図 12 判別アルゴリズム

1. マッチングする二つの物体の重心を求める．
2. 二つの物体のある一つの角を横軸に揃える．
3. 他の角の距離が全て近い場合，同じ障害物とする．
4. そうでない場合，他の角を揃え，ステップ 3 に戻る．
5. 全ての角で一致しない場合，異なる障害物とする．

地図情報に記録されたラスタ形式の画像データに，障害物の頂点，連結順番，幾何学重心などのデータを追加し，検出した障害物に ID をつける．このとき，前回と同じ障害物と判断したら同じ ID をつける．

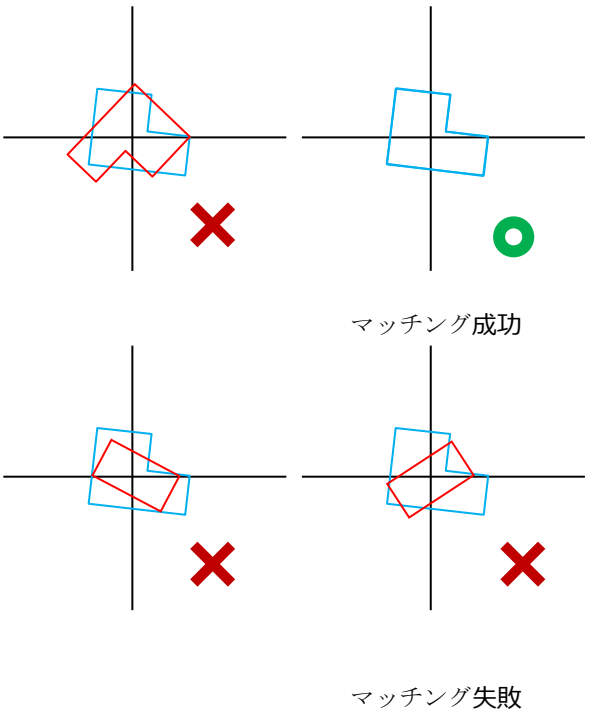


図 13 マッチングアルゴリズム

4.4 マッチング精度の評価実験

4.3 節で述べたマッチングアルゴリズムで，同一の障害

物を判別できるか確認するために，実験を行った．同じ面積で形状が異なる場合と，同じ形状で面積が異なる場合について，各 10 回を実験した．形状が異なる場合の実験結果を表 1 に，面積が異なる場合の実験結果を表 2 にそれぞれ示す．前回と同じ物体と判断したら「TRUE」と表記し，前回と違う物体と判断したら「FALSE」と表記する．

表 1　同じ面積で形状が異なる場合

三角形	四角形	五角形	六角形	円形
TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	TRUE

表 2　同じ形状で面積が異なる場合

小	中	大
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE
TRUE	TRUE	TRUE
TRUE	TRUE	TRUE
TRUE	TRUE	TRUE
TRUE	TRUE	TRUE
FALSE	TRUE	TRUE
FALSE	TRUE	TRUE
TRUE	TRUE	TRUE
TRUE	TRUE	TRUE

実験結果から，形状や面積が異なる場合には，おおむね，同一の障害物であることが判別できることが示された．

5.　掃除経路の探索（フェーズ３）

フェーズ１とフェーズ２により，掃除ロボットは，掃除するべき部屋の地図情報を保持している．そこで，保持している地図情報から，掃除する最短経路を探索する．ここでは，効率的な掃除経路として，全ての掃除空間をロボットが通過するという制約条件の中で，掃除時間の最も短い経路を探索する．

5.1　探索空間の区分

地図情報の掃除空間を，掃除ロボットと同じサイズのセルに区切る[2]．掃除ロボットが全てのセルを通過したら，全ての掃除空間を掃除したと定義する．ただし，障害物の周りのセルは 5.4 節で述べる周囲モードで掃除するため，図 14 に示すように，経路探索時の掃除空間からは削除する．

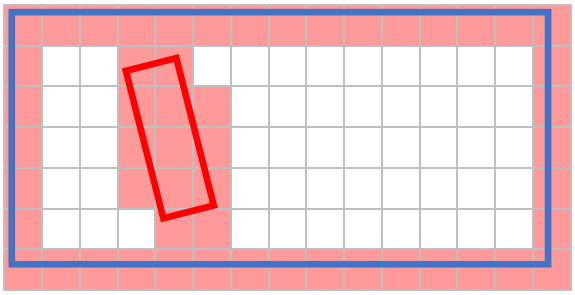
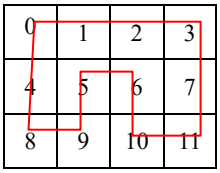


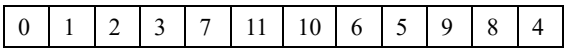
図 14 地図の区切りおよび障害物の周りの排除

5.2　経路探索に用いる遺伝的アルゴリズム

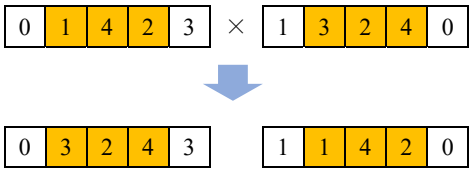
5.1 節で定義したセルをノードと捉える[3]と，本論文の経路探索問題は，全てのノードを最短経路で通過する巡回セールスマン問題と同じものとなる[4]．ここでは，以下のように設定した遺伝的アルゴリズムを用いて経路を探索する[5][6][7].

- 遺伝子：ノード（セル）番号



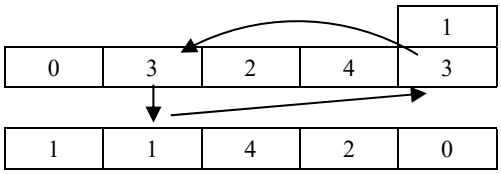


- 交叉：2 点交叉



- 重複チェック

交叉範囲外には交叉範囲と同じ遺伝子があった場合，もう一本の同じところの遺伝子の数字と入れ替える．これを重複がなくなるまで繰り返す．



- 評価値の計算

評価値を，移動距離+回転時間と定義する．

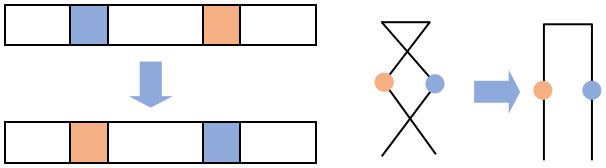
- 世帯交代

親個体数：5
子個体数：20（10 ペア，C_5^2）
次世代：親個体と子個体を全部合わせ，評価値の昇順で並べ，上位 5 個体を次世代の親個体として選択する．

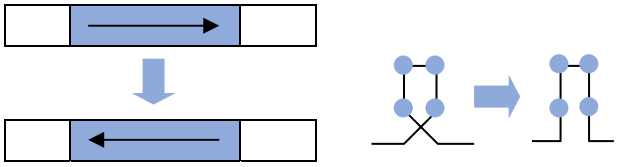
- 突然変異

個体に対して 71%の確率で，以下に示すいずれか一つの突然変異を実行する．突然変異として用いる 4 種類の手法は，以下の通りである．

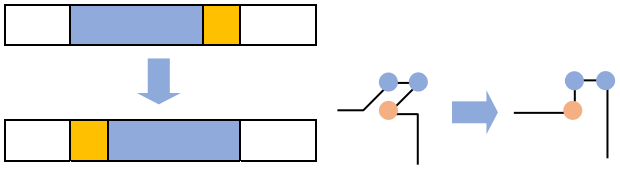
手法 1：2 ヶ所を交換する



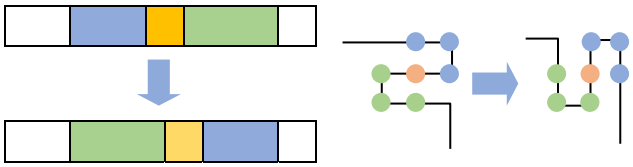
手法 2：一定区間を逆順に変更



手法 3：1 ヶ所を移動して間をずらす



手法 4：一定区間の 2 ヶ所（緑と青）を入れ替え，間の区間（黄色）をずらす



5.3 突然変異手法ごとの解探索性能の比較

まず，前節で述べた 4 つの突然変異の手法について，それぞれいずれか一つのみを用いた場合の実験結果を図 15 に示す．図 15 の通り，手法 2 が最も解探索性能が高いことが分かる．

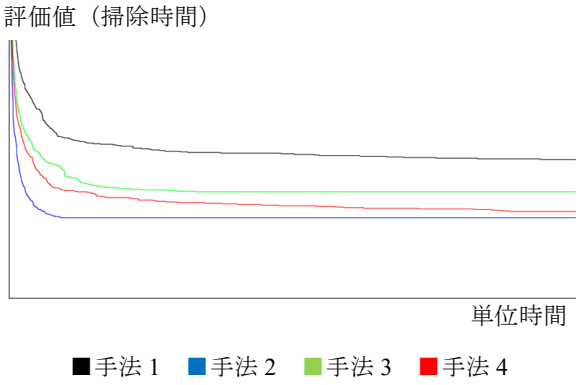


図 15　突然変異手法ごとの解探索性能

また，突然変異の手法 2 と 3 と 4 を合わせた場合の実験結果を図 16 に示す．図 16 から，複数の突然変異の手法を組み合わせることで，より解探索性能が高まることが分かる．

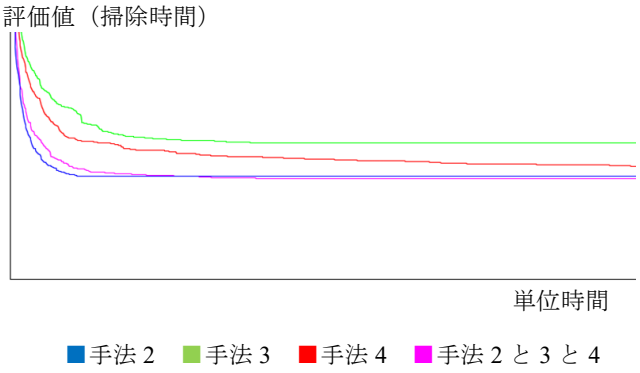


図 16　各手法と合わせる手法の対比

求められた掃除経路の例を図 17 に示す．初期状態では非効率な経路であったが，徐々に効率的な掃除経路を獲得し，最終的には最適解を得ていることが分かる．

5.4 障害物の周囲モード

障害物の周囲は，障害物の形に応じて周辺モードで得られた経路で掃除する．処理方法は以下の通りである．

1. 障害物を，ロボット半径の距離だけ膨張処理する．
2. 膨張後の障害物の周囲をベクトル化し経路とする．
3. 周囲モードの経路と最も近いセルと接続する．

これらの探索結果を図 20 に示す．掃除空間の全てを，短時間で掃除する経路を探索できていることが分かる．

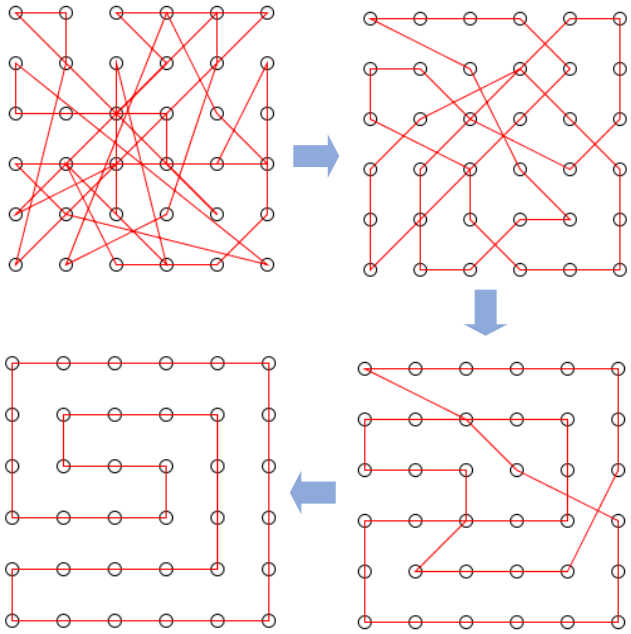


図 17　得られた掃除経路の推移の例

図 18　障害物の周囲の掃除経路

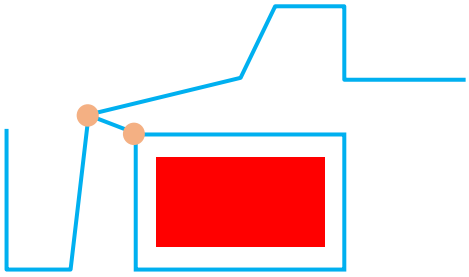


図 19 全体掃除経路への繋がり

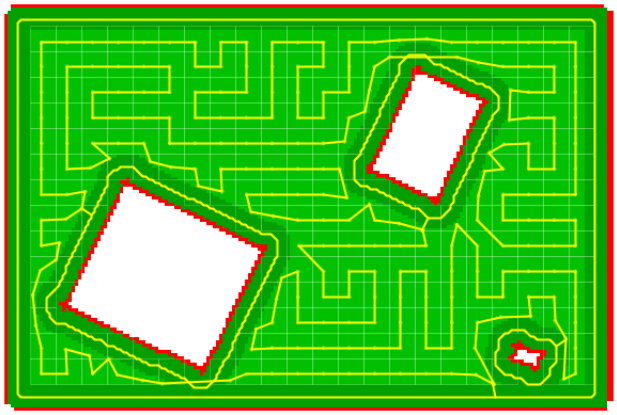


図 20　掃除経路計画

6. おわりに

本論文では，障害物との衝突センサのみを装備した掃除ロボットを想定し，室内の状況を把握して効率的な掃除経路を探索するアルゴリズムに取り組んだ．フェーズ１：地図生成では，未知の部屋においても，正しく地図が作成できることを示した．フェーズ２：障害物判定では，障害物が移動しても，おおむね正しく障害物をマッチングできることを示した．フェーズ３：掃除経路探索では，地図情報に基づいて，効率的な経路を探索できることを示した．

現時点では，従来手法との比較実験を実施しておらず，提案手法が従来手法よりどの程度，有効であるかの評価はできていない．今後，従来手法に対する有効性を明らかにしていく予定である．掃除中の部屋の状況変化に応じた掃除経路の動的な計算や，ほかの家庭用知能ロボットとの連動も，検討していきたい．

参考文献

[1] Chaomin Luo, Simon X. Yang, D.A. Stacey, Jan C Jofriet“A solution to vicinity problem of obstacles in complete coverage path planning”. 10.1109/ROBOT.2002.1013426.
[2] A. Zelinsky, R.A. Jarvis, J.C. Byrne, S. Yuta. ”Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot“.
[3] Ren Yuntian, "The Research and Design of Obstacle Avoidance Complete Coverage Path Planning Algorithm for Robot Vacuums".Huazhong University of Science & Technology, Wuhan 430074, P.R.China.
[4] Enric Galceran, Marc Carreras. “A Survey on Coverage Path Planning for Robotics”. September 24, 2013.
[5] Hiroaki SENGOKU, Ikuo YOSHIHARA. “A Fast TSP Solver Using GA on JAVA”.
[6] John J. Grefenstette, Rajeev Gopal, Brian J. Rosmaita, Dirk Van Gucht. ”Genetic Algorithms for the Traveling Salesman Problem”
[7] Hemmak Allaoua, Bouderah Brahim. ”A mono crossover genetic algorithm for TSP”. Global Journal on Technology, Issue 7 (2015) 109-115