環境認識の動的な変化に応じて プランを変更する BDI エージェント

奈良女子大学理学部情報科学科 片山 寛子

提出年月日：2010 年 1 月 30 日
指導教員：新出 尚之

概要

ロボット制御への自律エージェントの応用は、知的ロボットの実現のために有効と考えられる。これまでに我々は、ロボットを BDI エージェントにより制御し、現実世界における BDI エージェントの有用性および問題点を検証した。その結果、行動の際に発生する誤差が誤認を引き起こし、エージェント内部に間違った認識を与える問題点が確認された。

そこで、本研究ではエージェントにプランナを搭載し、行動の動的な再選択を可能にした。これにより、エージェントが自身の信念の誤りによって行動ができなくなる事態を回避することができた。また、異なる 2 つの戦略を状況に応じて使い分けることで、誤認した箇所を発見し、行動を修正することが可能となった。

1 はじめに

BDI エージェント [4] とは、信念 (B)、願望 (D)、意図 (I) の三つの心的状態パラメータを持つ、代表的な合理的かつ自律的なエージェントの一種である。BDI エージェントは人間の思考をモデル化しエージェントとして実現したものであり、熟考しながら自律的に行動選択を行うものである。

BDI エージェントは成し遂げたい願望を持つと、その目標に向けて達成すべく意図を形成する。意図は目標を達成するための大まかな行為列から成るものであり、エージェントは意図を形成することにより目標達成に向けてある程度大まかな行動列を形成することができる。これらの行動列に対しての行動方法については、複数のプランの中からその行動が実現可能であると判断されるプランを選択することによって形成される。

従って、意図が形成されると目標達成までの指針が決まり、エージェントはそれに従って行動を行う。

合理的エージェントは他に保持している意図と衝突を起こすような意図は生成せず、無矛盾性を持つ。従ってエージェントは意図を保持することによって目標までの行動にある程度一貫した行動を取ることができる。

現状では BDI エージェントは主にシミュレータ上で扱われることが多く、実世界上でのロボットの実装における応用は少ない。2.1.2 節に挙げるように、ライントレースによってロボットの動作制御を行う例はあるが、あまり現実的なロボットの動作制御とは言い難いものである。また、意図の生成にも柔軟性を欠くものである。

従って本論文では、現実世界のロボットを BDI エージェントにより制御し、より複雑で柔軟な動作を行わせ、これにより現実的なロボット制御における BDI エージェントの有用性および問題点の検証を行う。

実世界での制御は、行動の誤差やシステム間の通信のタイミングなど、シミュレーション段階では起きない問題を考慮することになることが予想される。よって、実世界での現実的な行動に基づく実験による検証を積

むことは、今後 BDI ロボットの制御を具体的に応用していくにあたって有用であるといえる。

これまでに我々は、2.1 節で述べるように、ロボットを BDI エージェントで制御して迷路を探索させる実験を行った。その結果、エージェントの知覚の誤認によってエージェントが自身の信念の誤解に基づき行動ができなくなるという問題点が判明した。本論文ではその問題に焦点を当て、一度目標達成が失敗した場合でも再考できるように BDI エージェントにプランナを搭載し、再プランニングのメカニズムを取り入れることで目標達成を図った。同時に、誤認による再プランニングで用いられる戦略として、通常の探索方法、つまり探索で得られた知覚を正確な信念として用いる 4.1.1 節の search とは別に、誤認したかもしれないという疑いを持ちながら探索する 4.1.2 節の research という新たな探索方法を設計し、実装した。

エージェントは、プランナと通信することでこの二つの探索方法を使い分け、状況に応じて戦略を変えるという手法を取ることで、誤認による目標達成の失敗を回避することができた。

本論文では、はじめに研究背景、基本知識と基本設計、BDI エージェントの設計を説明した後、実験と考察について述べる。

なお、本研究は、奈良女子大学院博士後期課程 1 年藤田と、奈良女子大学 4 回小島との共同研究である。

2 背景

2.1 従来の実験

これまでに我々は簡易なロボットに BDI エージェントを搭載し、シミュレータ上では確認できない実世界上での問題点を探り、その有用性を検証した。

そのために取り扱った例題として、エージェントに迷路内のオブジェクトを発見するという願望を持たせ、迷路探索をさせるという実験を行った [6]。

環境設定は、今回の実験と同様であり、4.1 節で後述する。また、BDI エージェントはプランナを用いず、今回の研究の中で取り扱った、4.1.1 節と 4.1.2 節に述べる 2 つの戦略のうち、search の方のみを用いる。すなわち、エージェントは自身の信念の信憑度に対して絶対的な価値感をもって行動する。

実験では、迷路にあたるマップ空間を実際に用意して、そこでロボットは実際に壁を知覚しながら探索作業を進め、オブジェクトを発見すると元の探索ルートを辿って元の位置に戻るということを行った。

2.1.1 システム概要

ロボットには LEGO MINDSTORMS NXT を採用した。

このロボットは比較的操作が扱いやすく、安価であるためロボットを実際に扱って実験をすることには適したものであると言える。

NXT の制御には、Python 言語で書かれた既存の制御ライブラリを用いた。ロボットは PC から Bluetooth で通信を行うことにより制御命令を与えられ動作を行うようになっている。またロボットは行動の際に超音波センサによる知覚を行い、これを PC の方へ逆に通信することにより超音波センサで得られたロボットと壁との距離を送信するようになっている。

一方、BDI エージェントの構築には、BDI エージェント構築用言語 AgentSpeak に基づいて設計された Jason というインタプリタを採用した。BDI エージェントはそのアーキテクチャに則って行動選択を行い、それを実世界に反映させるため、前述した Python プログラムとネットワークによる通信を行うよう設計した。

これにより、エージェント側での Jason 内部で決定された行動は、Python プログラムが動いている PC と

通信を行うことにより伝達され、Python 側の PC は NXT と通信することによりロボットはエージェントの指令通り行動を行うことが可能となっている。

2.1.2 関連研究との比較

BDI エージェントによる NXT の制御例としては、1 節に述べた [3][2] などがある。

[3] の NXT の制御は、格子状に張られたテープの色を光センサで知覚し、その度合を見ることによって進んでいる。また、経路探索には、A-star 探索アルゴリズム [1] を使用し、予め与えられたマップから最短経路を割り出して探索を行っている。

我々の実験では、逐一迷路の壁を知覚して、その場その場で進むべき方向を選択しながら探索することで、目標を達成させている。この探索手段は、A-star アルゴリズムと比較して効率的ではないが、より現実的な問題を解決するのに適した方法であるといえる。

また BDI エージェントの観点から見て、[3] や [2] では、与えられた願望に対して持つ意図が一意的であり、その意図もひとつの原始的行動を指している。意図とは本来エージェントの行動の方針として扱われるべきであり、その中身は大まかなプランから構成される行動列から形成されるものである。また、願望に対しての意図が固定されていると柔軟な行動決定ができない。これに対し、今回の我々の実験では、単純な行動列による意図ではなく、行動の方針として意図を保持し、階層的に行動列を形成することにより、[3] や [2] のような BDI エージェントよりも BDI アーキテクチャの特質を有効に反映できる。

2.1.3 この実験による考察と問題点

実験の結果、ロボットは、エージェントの渡す行動命令を忠実に遂行した。

しかし、前進や回転などの行動命令は、床との摩擦や、NXT の電池の消耗によって行動に誤差が生じてしまう結果となった。最初はダンボールの床を使用していたが、あまりにも誤差がひどいため、摩擦の少ないプラスチックの床上で実験を行うことにしたが、一つの行動によるわずかな誤差が、行動を繰り返すことで大きな誤差につながるという結果になった。

この実験により誤差が原因で発生する問題の一つとして、知覚の誤認が確認された。

知覚の際、ロボットは前方に超音波を流し、その測定値で壁の有無を判断するが、ロボットが知覚するべき方向に対してある程度直角に向いていない場合、その方向に壁がない場合でもその前方に壁があると認識してしまう場合があった。

その誤差によって生じた誤認は、そのままエージェントの誤解として信念に反映されてしまう形となった。そしてその結果、信念をもとに行動選択を行っているエージェントは、壁に囲まれて、オブジェクトを発見するという目標を達成することができず、無限ループに陥ってしまうという問題が発生した。

3 基本設計

3.1 BDI エージェント

3.1.1 BDI アーキテクチャ

BDI エージェントのインタプリタは一般的に次のようなループで実装される [4]。

BDI-initerpreter
initialize-state();

do

option := option-generator(event-queue, B, D, I);

selected-options := deliberate(options, B, D, I);

execute(I);

get-new-external-events();

drop-successful-attitudes(B, D, I);

drop-impossible-attitudes(B, D, I);

until quit.

ここで、B は信念、D は願望、I は意図である。

初期条件を与えられると、このループの中で event-queue を読み、行動のリスト、option を返す。次に deliberate を用いて、最適な行動を option から取り出し、それを意図構造体 I に加える。そして、意図 I が原始的な論理式であれば execute で実行する。この後 get-new-external-events により外部の変化を受け取る。ここで、達成された目標と意図を捨てて意図と願望を修正し、次に不可能となった目標や実現不可能となった意図を捨ててまた修正し整合性を保つ。これを繰り返すことにより最終的に実現したい願望を達成するという仕組である。

最終的な願望を達成するまでエージェントは多数の副願望を達成していかなければならない。しかも、エージェントは最初から詳細なプランを立てるのではなく、最初は大まかなプランを立案し、実行の進行に従って、サブプランをより詳細なプランで埋めていく方式をとる。このようにして、最初は大まかなプランであった意図が、行動を繰り返すことによってより細密にされていき、最終的に願望を達成するという形式をとる。

このとき、エージェントは達成を試みる間は一貫してその目標を達成することに重きをおくという立場をとる、この姿勢を我々は「意図」と呼び、意図は目標を達成するための指針というものにあたる。

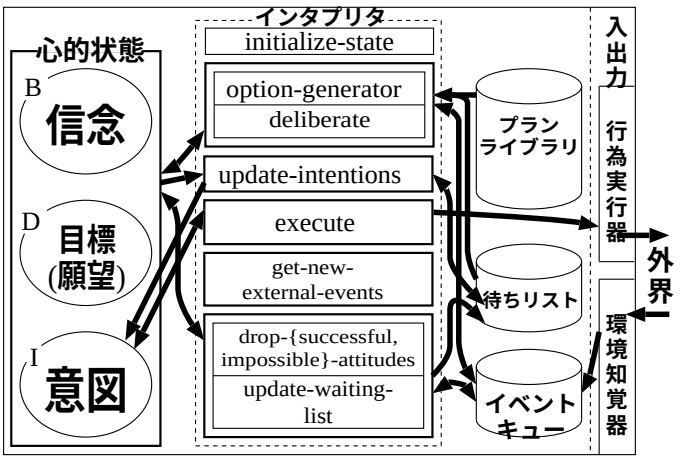

図 1　BDI アーキテクチャの概要図

3.1.2　Jason とは

Jason とは、BDI アーキテクチャを基礎とした AgentSpeak を拡張した言語のインタプリタおよび開発環境である。

Jason では、エージェントを定義し、行動決定を下すエージェントプログラムを AgentSpeak で実装し、そのエージェントが置かれる環境モデルを Java を用いて実装する。

環境設定プログラムによって環境プログラムを指定することにより、その環境とエージェントの相互作用が可能となる。我々は、環境シミュレートプログラムを連動させることで、計算機上での環境の可視化を行った。

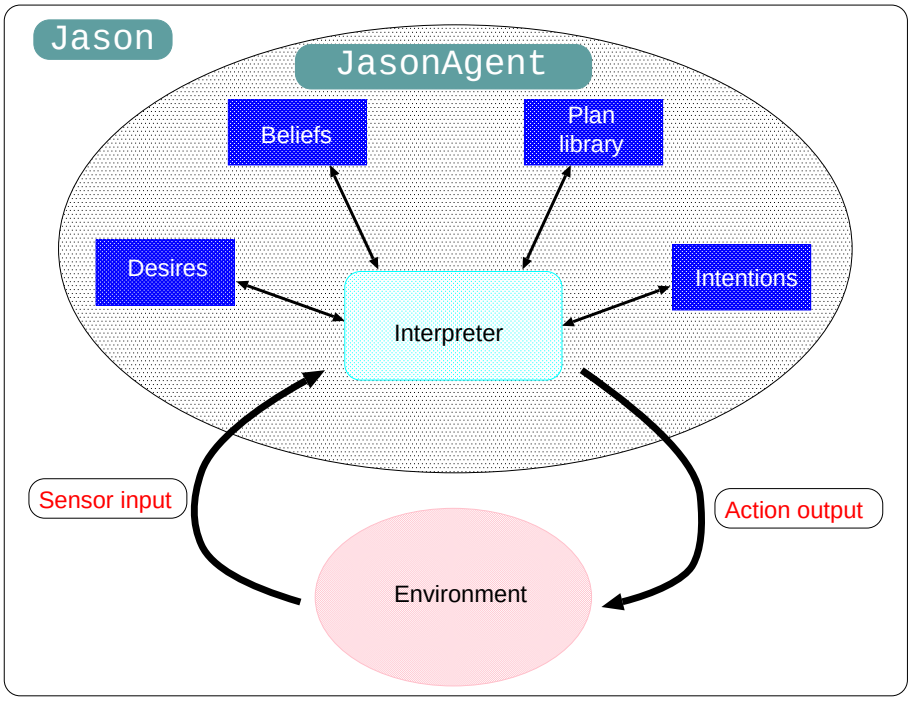

図 2　Jason と環境との相互作用

3.1.3　AgentSpeak による BDI エージェントの設計

我々が設計したエージェントはプランナと結合しており、上位の行動決定はエージェントによって行われ、最上位の行動決定 (search や research 等) はプランナによって行われる。

エージェントプログラムには AgentSpeak を使用した。

AgentSpeak は計算機上で BDI システムを実現するプログラミング言語であり、信念ベース、願望ベース、プランライブラリで構成される。

特定の問題に対しエージェントを設計する場合、初期設定として信念ベースには初期信念、願望ベースには達成すべき目標である初期願望を与えることができる。

プランライブラリには、予想されるイベントを駆動条件としたプランの集合を記述する。プランの中身に記述されるものには、これ以上分割されない基本行為や、このプランを達成するためのプロセスの一部であるサブプランがあり、これらの組合せによりそのプランが選択された場合の行動列が決定する。

プランの選択は、駆動条件と前提条件により行われる。イベントはある信念、または願望を獲得したときに発生し、それがプランの駆動条件となる。

例えば、ある願望を達成するために新たに副願望が発生した場合、副願望がイベントとしてサブプランの駆動条件となる。また、初期信念や初期願望はプログラムが起動して初めて発生するイベントであり、そこから

プランの選択が始まる。

発生したイベントがプランの駆動条件とマッチした場合、さらにそのプランの前提条件を満たすかを判定する。前提条件は、駆動条件を満たしたプランに対し、環境についての最新の情報を元に実行可能かをチェックする役割をもつ。つまり現在の信念から前提条件が導かれれば前提条件は満たされたことになり、この二つの条件を満たしたプランのみが選択可能となる。

このようにしてプランを選択することにより、状況に適した意図が形成され、エージェントはその意図に沿って願望を達成すべく信念を元により細密なプランを立てる。これを繰り返してエージェントは目標を達成するための一貫した行動をとる。

3.2 LEGO MINDSTORMS NXT およびその制御

LEGO MINDSTORMS NXT は LEGO 社が開発した、商用教育用のロボットである。安価で比較的制御が簡単であり、実験で用いるには適したロボットである。

今回作成したロボットは NXT の組み立てカタログどおりに作成したもので、タイヤが前方に二つ、後方に一つあり、前方のタイヤを回転させることにより、前進したり回転したりすることが可能である。

また、超音波センサを NXT の正面に設置し、前方に超音波を送ることにより、その方向に壁がないかを感知することが可能である。

NXT の制御用には、制御ツールや制御言語が数々存在するが、その中でも本研究では Python によるライブラリを使用することにする。Python 言語で NXT を制御するためのライブラリが配布されており、それをもとに制御プログラムを作成し、タイヤのトルク数を与えることによって前進、左回転、右回転といった行動の制御を行った。左回転と右回転は 90 度回転と 180 回転を行えるように設計し、実装した。

3.3 プランナの搭載

プランナとは、エージェントの行動列自動生成プログラムである。エージェントは自身の初期状態と達成願望をプランナに与え、それをもとにプランナは願望を達成する手段を組み立てる。

今回プランナは、共同研究者の藤田が作成したものを使用した [5]。このプランナは、SHOP2 をプラン探索のエンジンとして使用し、動的環境に対応可能なものである。

3.3.1 今回のプランナの使用法

エージェントの目標は、迷路の中を探索し、オブジェクトを発見し、元の道を辿り迷路を抜けることである。これを達成するため、エージェントはプランナに問い合わせ、プランナはこれを、(1) 迷路を探索し、オブジェクトを発見する、(2) バックトラックをする、という 2 つのサブゴールに分割する。

プランナは、これらのサブゴールを達成するために、上位の意図、つまり大まかな行動の指針を選択し、それにより全体の行動制御の流れを決定する。

(1) のサブゴールを達成する場合、そのための手段として今回は search と research という戦略を用意した。

エージェントは、プランナに問い合わせることで、この二つの探索方法を使い分け、状況に応じて戦略を変えるという手法を取ることで、誤認によりエージェントが目標達成を失敗するという事態を回避した。

3.4 全体の実装

2.1.1 節で述べたように、ロボットには NXT を使用し、BDI エージェントは Jason で構築した。そして、Jason と、Python の制御プログラムの間で通信を行うことにより、BDI エージェントによる NXT の制御を行った。

通信はソケットを用いた TCP プロトコルによる通信で行われる。Jason 側が送信する行動命令は、forward や right 90 など、行動を視覚的に表すテキスト文字列で与えられ、また Python 側が NXT に送信する制御命令は実際にロボットが動くためのモータのトルク数を扱ったレベルの命令を与えられる。このように、エージェントは高位のレベルによる命令で統一し、NXT と接続されている PC に対しては低位のレベルによる命令に統一し、レベル別に命令を統一し管理するような設計を施した。

さらに今回の実験では、BDI エージェントにプランナを搭載した。プランナは前述の [5] により実装された動的環境に適応するプランナを使用し、Jason とはプロセス間通信を行うことで、文字列の送受信を行った。Jason は、プランナに現在の状態を表す記号を文字列として送信し、プランナはそれをもとに最上位行動命令を生成する。そして、その命令を Jason 側に返し、Jason はそれを意図として、さらに綿密なプランを立てるよう設計した。

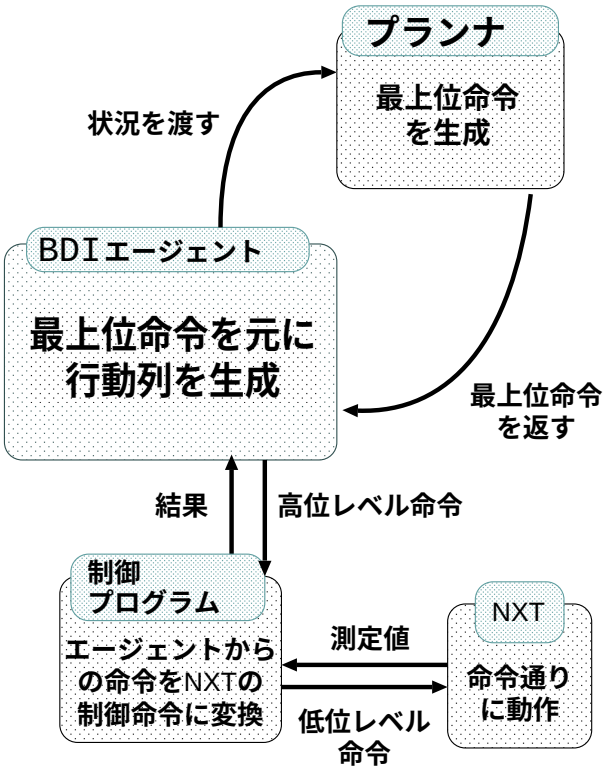

図 3　全体の実装

4　本研究における BDI エージェントの設計

本節では、本研究のうち著者が担当した部分である、BDI エージェントの構築について述べる。

4.1　迷路探索用 BDI エージェントの設計

今回我々が採用した例題は、図 7 のような 4×5 マスの迷路の中をエージェントがたどり、迷路内に配置されたオブジェクトを発見するというものである。

エージェントは、迷路内を上下左右に 1 度に 1 マス移動することができる。

エージェントに与える環境としては、捜索範囲を xy 平面上の $(0,0)$ から $(3,4)$ までの格子点で表現する。

壁は座標間に設置し、その座標間の移動はできないものとする。

初期設定として、ロボットの位置を $(0,0)$、ロボットの方向を y 軸の正方向、オブジェクトの位置を $(3,4)$ とする。

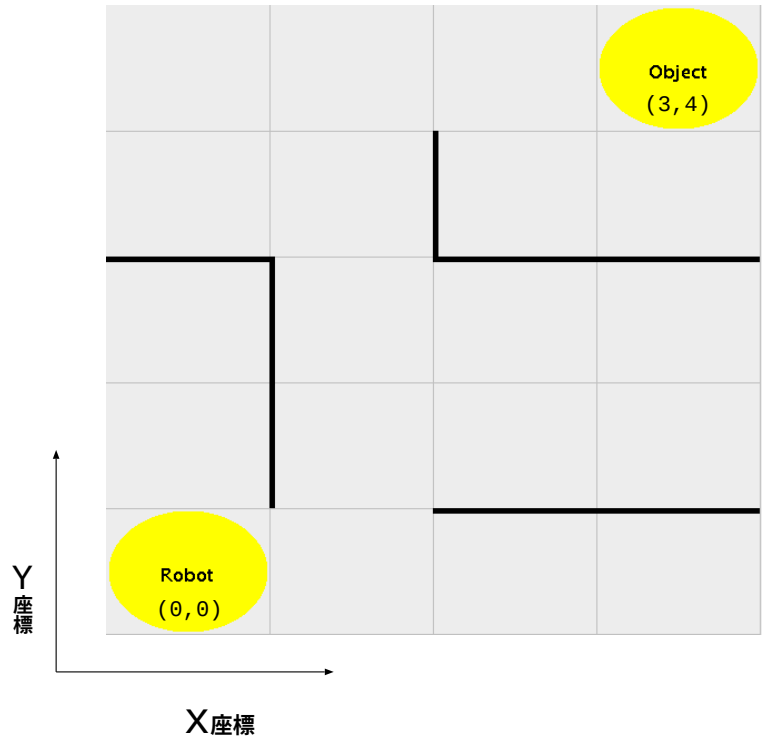

図 4　シミュレーションの図

エージェントは信念として捜索範囲のみ与えられ、最初の時点では壁がどこにあるかは知らない。地図は捜索作業の中で壁を知覚し、その情報を信念に追加することで徐々に完成されていく。

また知覚はロボットの位置に対し $+x$ 方向、$+y$ 方向、$-x$ 方向、$-y$ 方向の最大 4 方向に対して行われる。そこで得られる情報は隣り合うマス目間の壁の有無とオブジェクトの有無に限定する。

オブジェクトの探索は、知覚と移動の繰り返しにより行われる。知覚は超音波センサがロボットの正面に設置されていることから、知覚する方向に回転、前方に超音波を送信、センサの測定値から壁の有無を推測す

る、という手順で行う。

これらを基本行為とし、Python の制御プログラムに命令を渡すことでロボットは実世界上で知覚を行う。移動も同様に、移動する方向に回転し 1 マス分前進する、という手順を踏み、これらを Python に送ることにより行われる。

ロボットの探索方法として search と research という二つの戦略を用意した。

search は、環境から得た信念を完全に信用し再度知覚して確認することは行わない。

これに対して research は、信念に誤りがあるかもしれないという疑いを持って再度知覚を行い、信念ベースと照し合せながら捜索を行う。この異なる戦略はエージェントがプランナに問い合わせることにより使い分けられる。

4.1.1 search

search は、4 方向のうちオブジェクトのある可能性のある方向に対し知覚を行う。

それは、壁の信念のある方向や直前に通った座標への方向を含まない。

この条件のもと知覚を行うのであれば、当然以前通過したことのある座標上にオブジェクトが存在していないのは明らかであるが、この実験の次の段階として、迷路上でランダムに動くロボットの捜索を行うことを予定しており、対象が移動することを想定した設計となっている。

よって以前通過した座標であったとしても、その方向は知覚の対象に含まれる。

また、探索範囲は壁で仕切られているものとし、予めエージェントにその信念を与えているため、エリア外は知覚しない。

このような条件のもと一通り知覚が終了し、なおオブジェクトを発見できなかった場合、進路選択が行われる。

進路選択は、知覚によって得た壁の信念や、以前通過した座標の信念をもとに行われる。

壁の信念のある方向への移動は、選択肢から除外し、また既に通過した座標より、まだ通過していない座標への移動を優先することで、無限ループに陥ることなく全探索を行えるようにした。

このような進路選択を行い、なお選択肢が複数ある場合は、$+x$ 方向、$+y$ 方向、$-x$ 方向、$-y$ 方向という順に優先度をつけた。

エージェントはオブジェクトの位置やその方向、また壁の位置について全く情報がない状態で捜索を行うため、自身でその場での最良の進路選択を行うには限界がある。よって今回は優先度を固定することで、一つの道を選択できるようにした。

進路が決まりそこへ移動する際に、エージェントは通過した座標をパスとして信念に追加する。この信念には番号が割り振られており、トラックバックする際にこの番号を辿ることで以前通過した座標へ戻ることが可能となる。行き止まりや無限ループ回避などでバックトラックする場合、バックトラックした座標の信念に、番号の代わりに fail という記号を付けることで、オブジェクト発見には至らなかった道であることを表現した。

また、知覚と移動を繰り返し、すべての道を通過したがオブジェクトを発見できなかったと判断された場合、エージェントはプランナに問い合わせ、プランナの返答を待つように設計している。

以下の図は、search で探索した場合のロボットの動きを表している。

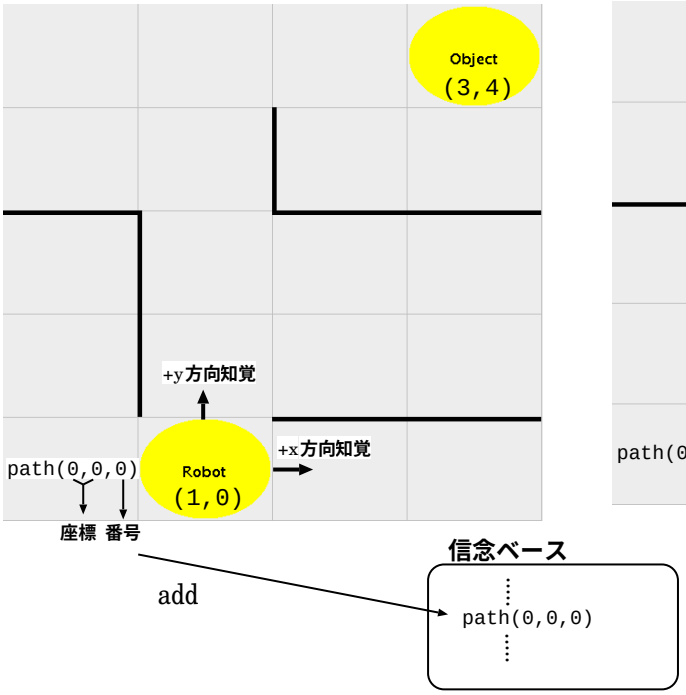

図 5　cycle1

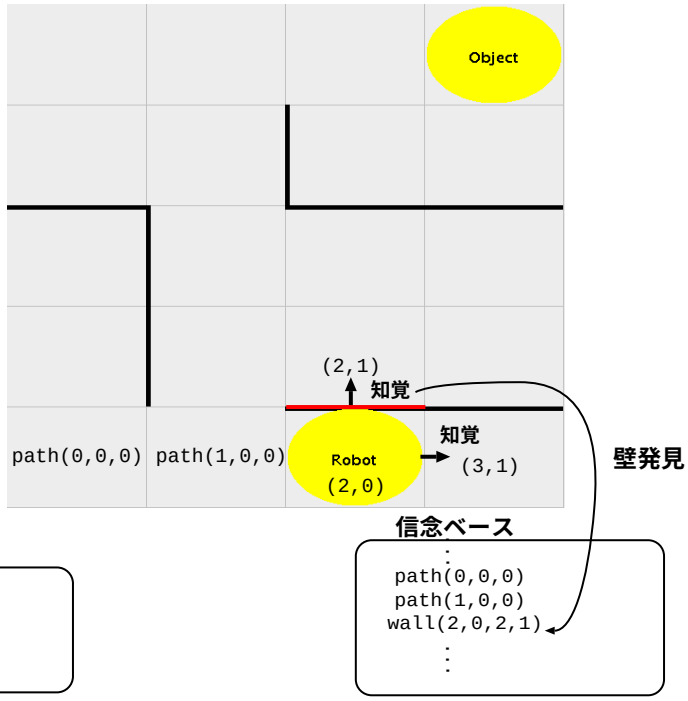

図 6　cycle2

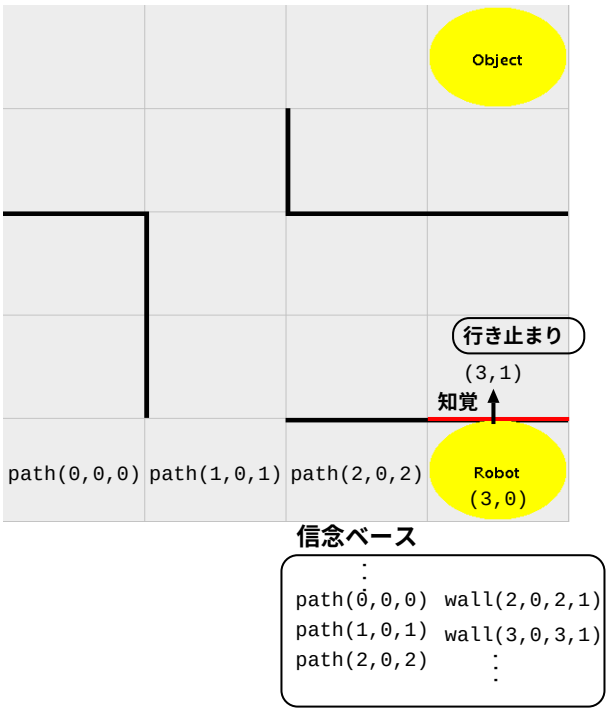

図 7　cycle3

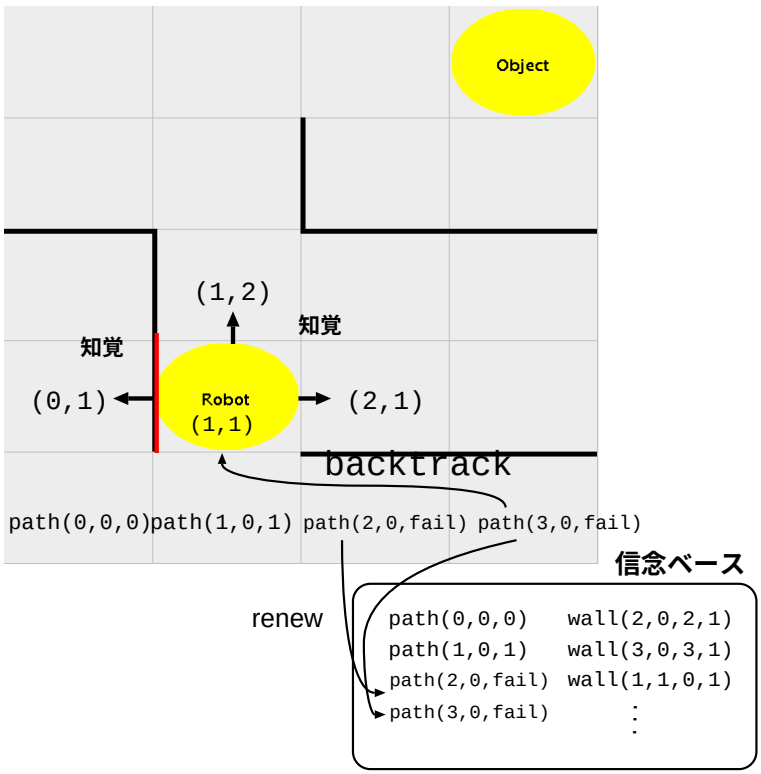

図 8　cycle4

4.1.2 research

research は、2 章で述べた問題を解決すべく設計されたもう一つの探索方法である。

全ての道を探索したがオブジェクトを発見できなかった場合でも、research に戦略を切替えることで、誤認したかもしれないという疑いを持って探索を続けることを可能にした。

research も search と同様、オブジェクトを発見することを目標とし探索を行うが、探索作業の中で得た壁の信念を疑いながら探索するため、知覚の対象となる方向が異なる。

research の知覚は、捜索範囲として仕切られている壁や、直前に通過した座標への方向を除いた、すべての方向に対し行われる。つまり、以前知覚によって壁の信念を得た方向も、知覚の対象に含まれることになる。そして、ある方向への知覚を行った後、知覚の結果とエージェントが持っている壁の信念リストを照し合せ、その方向に対して誤認があったかどうかを確認する。

誤認が確認された場合、壁の信念リストは更新される。また、それによって未開拓の道が発見された場合、エージェントは、プランナに新ルートを発見したことを伝え、プランナから指示が出るまで待機する。誤認ではなかったり、誤認していたが新ルート発見までには至らなかった場合、research は続行される。

知覚方向の選択において、今まで得てきた壁の信念を選択条件から外すのであれば、壁の信念をすべて削除して一から探索し直す方法と research は大差ないように思える。しかし、我々は誤認が頻繁に起こることは想定しておらず、ほんの一部の誤りのために壁の信念全てを削除するのは非効率的であり、それはマップの規模が大きい程言えることである。よって、知覚する度に保持している信念と照合し、必要な箇所だけ更新を行うという手法をとった。

search と同様、research も 1 マスにつき一通り周囲を知覚したら進路選択に移る。進路選択は、壁の信念をもとに行われ、壁があるとされる方向は選択肢から除外される。また、research モードで移動する場合も、通過する座標に research 特有のパスをつけることで、新ルートを見つけるまで移動可能な全ての道を再探索することが可能である。

research によって全ての道を再探索しても新ルートが発見できない場合、プランナに問い合わせ、プランナの返答を待つよう設計している。

5 実験

3.4 節で述べた設計によるロボットで、実際に迷路を探索し、オブジェクトを発見するということを目指した。

マップは、4×5 マスに仕切ったものを使用し、マス目間に壁を設置した。ロボットには、その中で超音波センサによる壁認識を行わせ、それをもとに探索を行わせた。

今回の実験では、前回の実験で確認された、壁の誤認によりエージェントが目標を達成できなくなる事態に対し、プランナを搭載することによる問題点の解決に焦点を当てる。

5.1 実験手法

実験では、ロボットが壁のない方向を知覚しているときに、その方向を塞ぐことでそこに壁があると認識させ、意図的に誤認を発生させた。そして、知覚した後塞いだ箇所を取り外し、ロボットの動向を観察した。

実際には、図 9 の赤線部分に一時的に壁を設置し、エージェントがその方向を知覚した後、取り外すことを

行った。このマップの場合、塞いだ箇所はオブジェクトを発見するために通過しなければならない道であり、オブジェクトへの唯一の経路を誤認させることで再プランニングを引き起こし、戦略を research に切替えさせた。

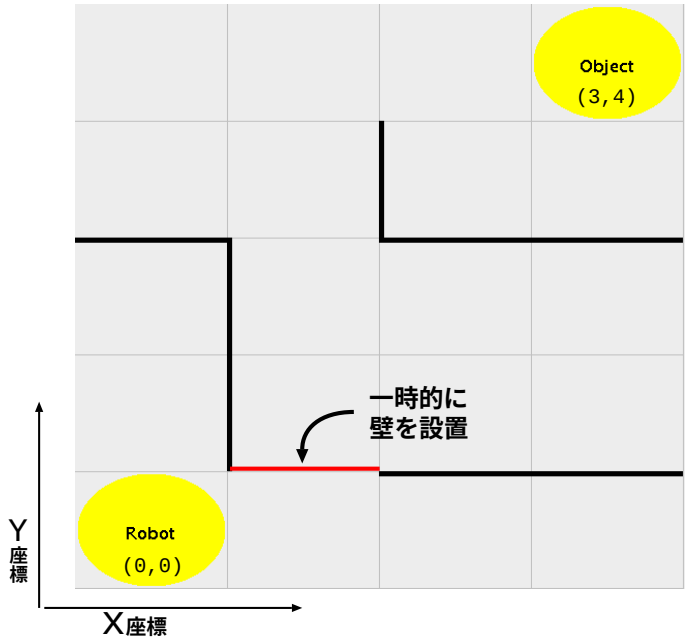

図 9　実験

5.2　実験結果

ロボットは初め、search モードで探索を開始し、故意に塞いだ箇所を、壁があると認識した (図 10)。その後、正確に壁を知覚し、行き止まりまで行き着いた。

この時点で、塞がれていた壁は取り外されていたが、まだ通過していない道があったので、ロボットは search モードのまま、バックトラックを行い、壁が取り外された箇所は通過する際に知覚されなかった (図 11)。

そして、再度行き止まりにつきあたって、もう通過していない道がなくなると、この時点でエージェントはプランナに問い合わせ、research モードに切り替わった (図 12)。

ロボットは再び探索を始め、取り外した箇所を再度知覚し、壁がないことを認識した。そして、発見した道が未開拓であることをプランナに伝達し、それによって再び search モードに切り替わった (図 13)。

そこから search モードで探索を行い、オブジェクトを発見し、プランナに問い合わせて return モードに切り替わった (図 14)。

return モードのロボットは、もと来た道を通過して元の場所まで戻った (図 15)。

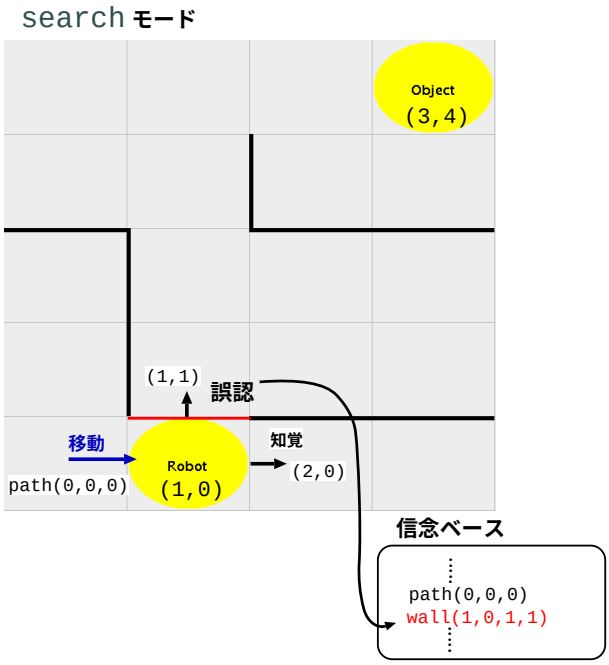

図 10　cycle1

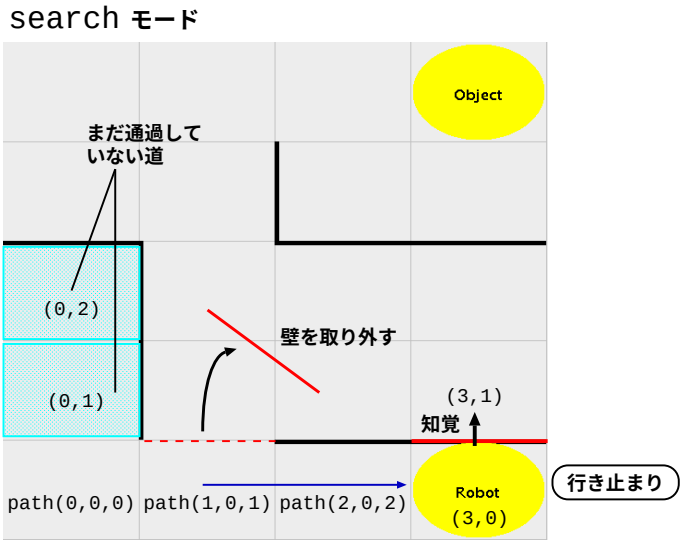

図 11　cycle2

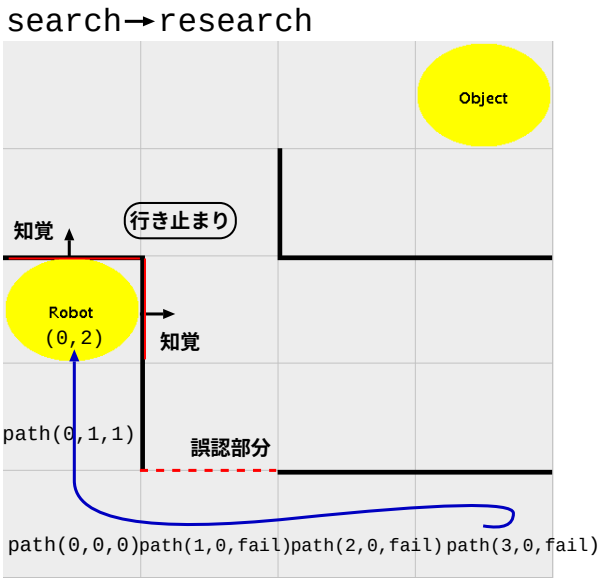

図 12　cycle3

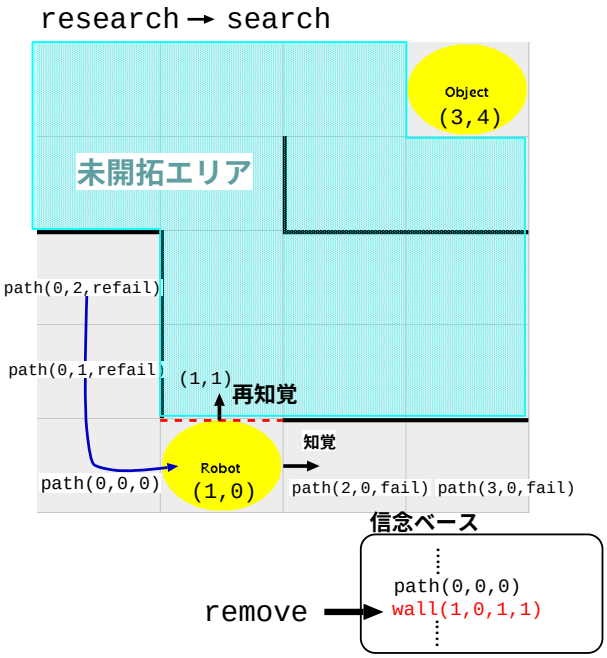

図 13　cycle4

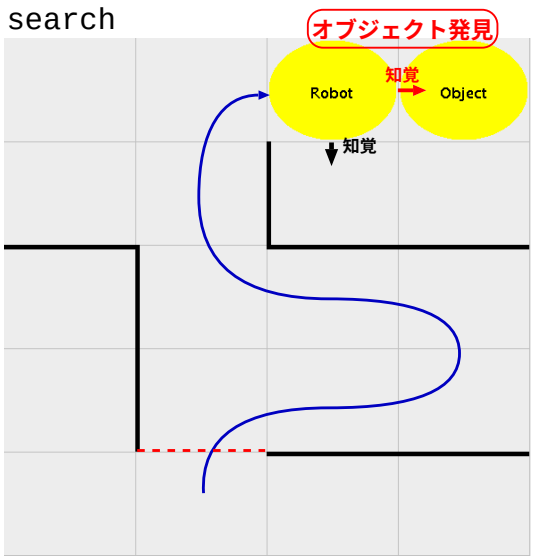

図 14　cycle5

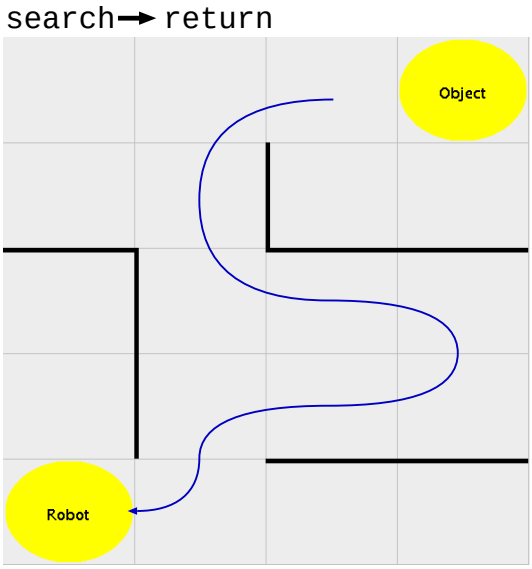

図 15　cycle6

5.3　考察

この実験結果より、誤認によって八方塞がりという事態に陥っても、プランナに問い合わせ戦略を切替えることで、必ず誤認した箇所を発見し、行動が修正されることが確認された。

今回は行動の誤差によって生じる誤認に焦点を当て、エージェント側で問題解決を図った。しかし、誤差による問題がすべて解決されたわけではなく、回転や前進の際に生じる誤差によってロボットの向きや位置がずれ、エージェントの認識する世界と現実世界との連携がとれず、誤差を手動で修正しなければ制御ができなくなるという問題が、前回の実験に引き続き残った。

誤差の対処として、エージェントは、現実世界をマス目空間とみなし、行動補正を行うなどしてできる限りエージェントの世界に近づけるという方法と、マス目という概念を無くし、誤差を許容して現実世界をエージェントに認識させるという方法が考えられる。

しかし、マス目を無くすことはエージェントの世界での基準位置を無くすことであり、この迷路の問題ではエージェントが自身や壁の位置情報を把握することができなくなるという問題が発生する。

現実世界とエージェントの世界認識を正しく対応させるという問題は今後の課題である。

5.4　将来展望

今回の研究では、プランナを搭載したことにより、失敗した場合も再行動を行わせることが可能となった。今後の課題としては、行動補正を行い、行動の誤差を小さくすることが挙げられる。また、使用したのプランナは、動的環境にも対応しており、次の段階として、オブジェクトの代わりにロボットを用意し、迷路内を彷徨わせて、それを発見するという問題設定に変更し、動的環境下でのプランナの有用性を確かめたい。

参考文献

[1] Hart P. E, Nilsson N. J, and Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, pp. 100–107, 1968.

[2] Soren Andreas Juul, Kaspar Henrik Moss Lyngsie, Michael Vandborg, Kim Fiedler Vestergaard, Torsteinn Sævar Hjartarson, and René Bach Gustafson. Agent Programming — Robot Traffic. *Aalborg University*, 2008.

[3] Anders Lemke, Johan Laidlaw, and Lars Zilmer-Pedersen. Developing Multi-Agent Lego Robotics. *Technical University Of Denmark*, 2007.

[4] Munindar P. Singh, Anand S. Rao, and Michael P. Georgeff. Formal Methods in DAI. *Multiagent systems: a modern approach to distributed artificial intelligence*, pp. 331–376, 1999.

[5] 藤田恵, 新出尚之. 動的な環境に対応する BDI エージェントのためのプランナの設計と実装. In *Proc. of Joint Agent Workshops and Symposium 2008*, 2008.

[6] 藤田恵, 小島侑子, 片山寛子, 新出尚之. 実世界での BDI ベースのロボットの柔軟な行為決定の実現に向けて. In *Proc. of Joint Agent Workshops and Symposium 2009*, pp. 354–361, 2009.