計測自動制御学会東北支部 第190回研究集会 (2000.10.21)
資料番号 190-6

自律移動ロボットのための トポロジカルマップを用いた環境表現

Environmental representation method using topological map for a autonomous mobile robot

○及川一美*，高氏秀則**，江丸貴紀**，土谷武士**，大久保重範*

○Kazumi Oikawa*，Hidenori Takauji**，Takanori Emaru**，
Takeshi Tsuchiya**，Shigenori Okubo*

*山形大学工学部, **北海道大学大学院工学研究科

*Faculty of Engineering, Yamagata University,
*Graduate School of Engineering, Hokkaido University

キーワード： 環境地図自動生成 (Spatial learning)，トポロジカルマップ (Topological map)，
行動規範型ロボット (Behavior-based Robotics)，自律移動ロボット (Autonomous mobile robot)

連絡先： 〒992-8510　米沢市城南四丁目三番地十六号　山形大学 工学部 機械システム工学科
及川一美，Tel.: (0238)26-3246，Fax.: (0238)26-3246，E-mail: okazu@dip.yz.yamagata-u.ac.jp

1. はじめに

本稿ではロボットが自律的に作業環境の地図を作成する手法とその表現方法について述べる．ロボットの作業環境をロボットが探索し環境地図として自律的に生成することは，ロボットを利用する側の負担を減らす意味で有用であり多くの研究がある[1)]．従来の研究では絶対座標を表現方法とする手法が多く提案されているが，精度の高いセンサや慎重な計測が必要であるなど実現するのはそう簡単ではない．一方，そのような問題を解決する為に計測的な情報を持たないトポロジカルな表現手法も多く提案されており[2, 3)]，我々も環境地図の性質として「自己組織性」と「可塑性」が必要であるとし，動的に変化してしまうような環境を環境地図として表現する手法としてトポロジカルマップの自動生成手法を提案した[4, 5)]．

計測情報を持たないトポロジカルマップの生成は絶対座標を用いる手法に比べ容易であるが，自己位置を正確に特定するのは困難であるため，その実現もそう簡単なものではない．自己位置を特定する為には高度な環境認識手法が必要である．何故なら低級な認識では同じものと認識してしまう空間が多数存在してしまい，自己位置を識別することが困難になってしまうからである．環境を整備することによって，低級な認識手法でも識別を可能とすることができるようになるが，その手間が利用者側の大きな負担となれば自律的に環境地図を作らせる意義がほとんどなくなってしまう．したがって，ここでは認識を容易にする為に環境

は整備したものを利用するが，できるだけ整備しない環境をあたえ，そのような環境をロボットが探索しトポロジカルマップを生成する方法について考える．ここで問題となるのは，ロボットが得られる情報だけでは自己位置を特定できない場合が生じたときにどうするかということであるが，本稿ではロボットが幾つかの仮説を立て試行し，その仮説と現実が異なればその仮説を捨て別の仮説に基づき行動するという手法でこの問題の解決を図った．以降，提案するトポロジカルマップの表現方法並びに生成手法について述べ，シミュレーションによりその有効性を示す．

2. 作業環境の設定

ロボットの作業環境は従来と同様，転回点に挟まれた枝路と片側が行き止まりとなる枝路で構成される通路状環境とする．転回点同士は高々一本の枝路で繋がっており，どの枝路の両端も同じ転回点にならないものとする．転回点には識別を容易にするためのランドマークが用意されている整備された環境とする．ランドマークはそれぞれ識別番号を持ち，電波を用いて識別信号を発信するタイプのものを想定している．これは以下の理由からである．

- ロボットに転回点を自律的に認識させるためには視覚センサなどの技術を要するが，それは容易に実現できるものではないということ．
- 保守が容易であること．
- 識別番号を使うことにより目的地等の指示が容易になるということ．

ロボットはこのランドマークを発見しながら環境地図を生成する．

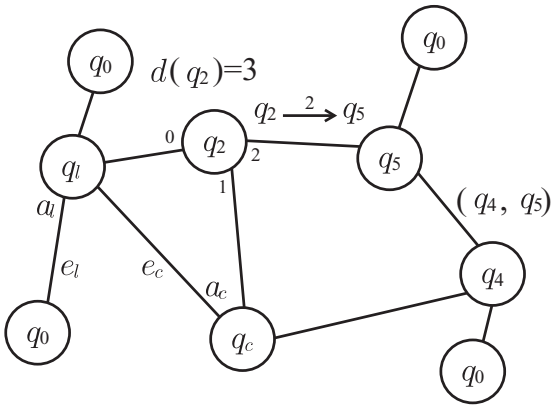

Fig. 1 Topological map

3. トポロジカルマップ

環境地図は**Fig. 1**に示されるような転回点と行き止まりを節点とし枝路を辺とする単純無向グラフで構成されるトポロジカルマップを用いる．ここでトポロジカルマップを次のような$G = (V, F)$で表すことにする．ただし，Vは節点の集合であり，Fは辺の集合で$F \subset V \times V$である．転回点を表す節点にはランドマークの識別番号iを用いてq_iと表し，節点q_i, q_jを結ぶ辺を(q_i, q_j)と表す．単純無向グラフなので$(q_i, q_j) = (q_j, q_i)$ $(i \neq j)$である．識別番号“0”は特別な意味を持たせ行き止まりを表すことにし，その節点をq_0と表す．各節点q_iと隣接している行き止まり以外の節点の集合をV_{q_i}と表す．節点の次数を$d(q_i)$と表すとすると，左回りに辺が結ばれている順にラベル$(0, \ldots, d(q_i)-1)$をふり，このラベルを節点における，もしくは転回点における方向と呼ぶことにする．また，節点q_iとq_jがq_iのa番目の方向の辺(q_i, q_j)で結ばれているとき，$q_i \xrightarrow{a} q_j$と表し，次の関数

$$\alpha(q_i, (q_i, q_j)) = a \quad (1)$$

$$\delta(q_i, a) = q_j \quad (2)$$

$$\epsilon(q_i, a) = (q_i, q_j) \quad (3)$$

を用意しておく．式(1)はq_iに接続している辺(q_i, q_j)の方向を返す関数である．式(2)はq_iの方向aに隣接する節点を返す関数である．式(3)はq_iの方向

aに接続されている辺を返す関数である．以降どのように自律移動ロボットが作業環境を探索して，トポロジカルマップを生成していくのか述べる．

3.1 トポロジカルマップの生成

初期状態ではどの節点にも到達したことがない状態$V = \phi$で探索を始める．ここで，ロボットが現在いる節点をq_c，前回いた節点をq_lと表し，q_cに到達したときに通ってきた辺をe_cとしてその辺が結ばれているq_cにおける方向をa_cとし，q_lに到達したときに通ってきた辺をe_l，q_lにおける方向をa_lと表すことにする（**Fig. 1**）．そして，ある節点で次にある方向へ移動した結果どの節点に行けるかをこのトポロジカルマップから予想した節点をq_gと表す．ここで，前回いた節点が分からない場合や予想ができない場合などを$q_l = \emptyset, q_g = \emptyset$などのように表すことにする．

環境の探索戦略を**Fig. 2**に示す．基本的には知識を用いて予想し現実と照らし合わせて知識を修正することを行う．本手法は環境地図をトポロジカルマップで表しているので，ロボットはマップの辺に相当する枝路に沿って移動する．ロボットが転回点に到達すると，前回の転回点で予想した転回点に到達したかを調べる．その結果に合わせて環境地図を修正する．そして次にどの枝路に進入するか選択する．現在の転回点と選択した枝路から次に来る転回点を予想し，その枝路に沿って移動することを繰り返す．

今回は簡単のため，静的障害物や移動障害物に

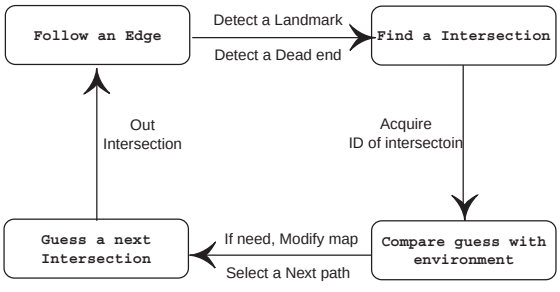

Fig. 2　Basic exploration strategy

行く手を阻まれるなど，何らかの原因で枝路を進行中に振り返ったり，転回点で意図する枝路に進入できないなどの失敗を起こさずに，環境を探索しトポロジカルマップを生成できるものとする．

3.1.1 予想と比較・環境モデルの修正

ロボットが予想した転回点と実際に来た転回点を比較し，どのように環境地図を修正するのかについて述べる．まず知識が乏しくて予想ができなかった場合と知識から予想できた場合で分けて考える．

$q_g = \emptyset$ 予想ができなかった場合

更に以下の場合で分けて考える．

- $q_i \notin V$のとき．つまり，識別番号iの転回点に来るのは初めてで，現在の節点q_iがトポロジカルマップに登録されていない場合は，q_iを到達したことのある節点の集合V及び前回の節点q_lに隣接している節点の集合V_{q_l}に，q_lをq_iに隣接している節点の集合V_{q_i}に加え，(q_l, q_i)を通ったことのある辺の集合Fに加える．また，$q_l \xrightarrow{a_l} q_i$をセットする．つまり，$q_l$における方向$a_l$に$a_i$が隣接していることを記憶する．そして，$a_c = 0$として$q_i \xrightarrow{a_c=0} q_l$もセットする．つまり，初めて来た転回点の現在の方向は0と決めておく．ただし，初期状態のように前回の節点が分からないの場合はq_lに関する処理を行わない．

- $q_i \in V$で，$q_l \notin V_{q_i}$，$q_i \notin V_{q_l}$　（$q_l \notin V_{q_i}$と$q_i \notin V_{q_l}$は同値なので以下どちらかのみを示す）であるとき．つまり，識別番号iの転回点に来たことがあるが前回の節点q_lから来るのは初めての場合．q_lをV_{q_i}にq_iをV_{q_l}に加え，(q_l, q_i)をFに加える．また$q_l \xrightarrow{a_l} q_i$をセットする．これだけの情報では現在の節点$q_i$のどの方向にいるのかが分からないので，

考えられる全ての仮説を立て式(1)を用いて$a_c = \alpha(q_i, (q_i, q_l))$にいると仮定する．仮説の立て方は後で述べる．

- $q_i \in V$，$q_l \in V_{q_i}$であるとき．つまり，節点q_iには来たことがあって，尚且つ前回の節点q_lとは隣接することが分かっている場合．隣接する事が分かっているということは，$q_l \xrightarrow{a_l} q_i$がセットされているということであるから，式(2)を使って$\delta(q_l, a_l) = q_i$と現在の転回点に到達できることを予想できたはずである．にもかかわらず予想ができなかったということは，前回の節点での仮定で$\delta(q_l, a_l) \neq q_i$となっており，その仮定が間違っているということである．そこでまず，現在のq_iにおける方向を$a_c = \alpha(q_i, (q_i, q_l))$により求める．そして$q_l$に対して誤りのある仮説を捨てる為，仮説の枝刈りを行う．仮説の枝刈りについては後で述べる．

$q_g \neq \emptyset$ 予想ができた場合

更に以下の場合に分けて考える．

- $q_g = q_i$であるとき．つまり予想した節点と実際の節点が等しいとき．予想通りではあるが仮説がまだ待ち行列の中に2個以上残っている場合は，誤りのある仮説を取り除く為に枝刈りを行う．

- $q_g \neq q_i$，$q_i \notin V$であるとき．つまり，別の節点を予想した上に，現在の節点は今までに来たことのない節点であった場合．まず，q_iをVに，q_lをV_{q_i}に加え，(q_l, q_i)をFに加える．q_iには初めて来たので，現在の方向$a_c = 0$として$q_i \xrightarrow{a_c=0} q_l$をセットする．予想と異なったのは前回の転回点での仮定が間違っていたということであり$q_i \notin V_{q_l}$であったから，q_lのそれぞれの仮説に$q_l \longrightarrow q_i$をセットするために仮説を修正する．修正の方法は後で述べる．そして，q_iをV_{q_l}に加える．

- $q_g \neq q_i$，$q_i \in V$，$q_l \notin V_{q_i}$であるとき．つまり，予想を立てることはできたが，以前来たことのある節点でq_lから来るのは初めての節点q_iに到達したとき．まず，q_lをV_{q_i}に加え，(q_l, q_i)をFに加える．現在の方向が分からないので仮説を立て，現在の方向を$a_c = \alpha(q_i, (q_i, q_l))$と仮定する．予想が異なったのは前回の転回点での仮定が間違っていたということであり，$q_i \notin V_{q_l}$でもあるから仮説の修正を行う．そして，q_iをV_{q_l}に加える．

- $q_g \neq q_i$，$q_i \in V$，$q_l \in V_{q_i}$であるとき．つまり，q_lから来たことがあるにもかかわらず，別の節点を予想した場合．この場合は前回の転回点での仮定が間違っていたということである．まず，現在の方向を$a_c = \alpha(q_i, (q_i, q_l))$により求める．そして誤りのある仮説を取り除くため仮説の枝刈りを行う．

仮説を立てる　仮説を立てるというのは，その転回点で考えられる全ての枝路の並び方の組合わせを考えることである．仮説を立てる処理は，人工知能で言う推論における展開そのものである．

各節点q_iは待ち行列Q_iと方向でラベル付けされた辺の組合わせ，つまり左回りにどの順番で枝

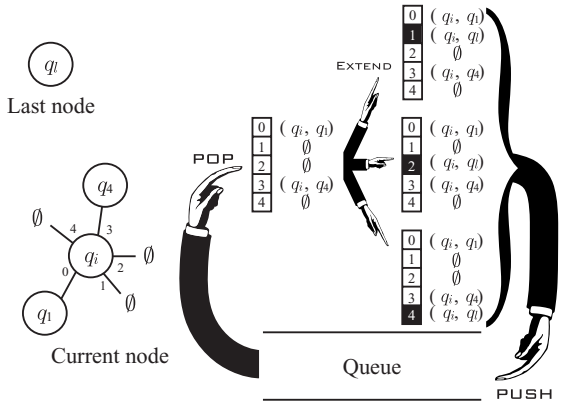

Fig. 3　Make assumptions

路が繋がっているかの情報を持っている．繋がっている辺の数が$d(q_i)$である節点$q_i \in V$が$q_l \notin V_{q_i}$であるとき．つまりq_iには来るのは初めてではなく，q_lから来るのは初めてであるため，辺(q_i, q_l)がどの方向に繋がっているか特定できないとき仮説を立てる．Q_iには幾つかの辺の組合わせが入れられており，それをここでは仮説と呼ぶことにする．初期状態ではQ_iには現在のq_iの辺の組合わせのみが入っている．Q_iの全ての仮説に対して次のような処理を行う．

Q_iから仮説を取り出して$\epsilon(q_i, a) = \emptyset$であるような方向$a$が$n < d(q_i)$個あるならば，その$\epsilon(q_i, a_j) = \emptyset$ $(j = 1, \ldots, n)$となる方向に$q_i \xrightarrow{a_j} q_l$とする$n$個の仮説を立て$Q_i$に入れる．仮説の入れ方によって探索方法が変わるが，ここでは幅優先探索になるように入れるようにした．全ての仮説に対して処理を行った後，Q_iの先頭の仮説を現在のq_iの辺の組合わせと仮定する．

仮説の修正　仮説の修正は節点q_lで予想した次に来る節点とは異なる節点q_iに到達し，その節点q_iにはq_lから初めて来た場合，q_lの全ての仮説に対して$q_l \xrightarrow{a} q_i$をセットすることである．しかし，セットできないような仮説は捨てられる．

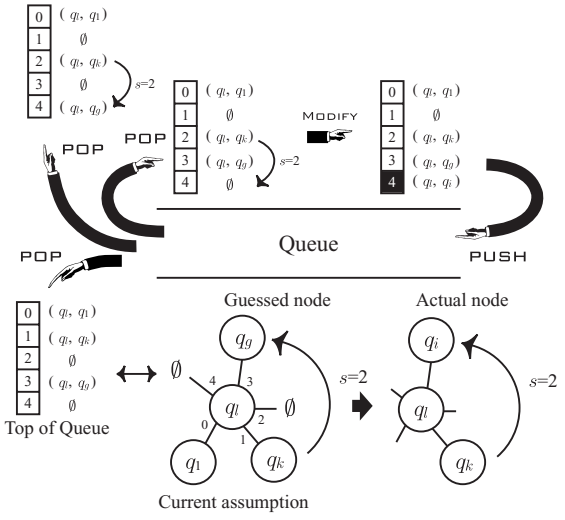

Fig. 4　Modify assumptions

$x + y$をpで割った余りを$x \oplus_p y$と表すとすると，結ばれている辺の数が$d(q_l)$である節点q_lに方向aから左回りにs番目の方向$a \oplus_{d(q_l)} s$（以下，簡単のため$a \oplus s$と表記する）の辺に沿って移動すれば仮定から$q_g = \delta(q_l, a \oplus s)$と予測できるが，実際には$q_i \in V$，$q_l \notin V_{q_i}$ $(i \neq g)$である節点に到達した場合，q_lの仮定が間違っていたということであるから，待ち行列Q_lの先頭の仮説を捨て残りの仮説に対して次のような修正を行う．

q_lにはq_kから来たものとする．Q_lの仮説を1つ取り出して$a' = \alpha(q_l, (q_k, q_l)) \oplus s$で尚且つ$\epsilon(q_l, a') = \emptyset$である方向に$q_l \xrightarrow{a'} q_i$を設定し$Q_l$に入れる．このとき$\epsilon(q_l, a') \neq \emptyset$であるならばこの仮説は捨てる．最後に$Q_l$の先頭の仮説を$q_l$の辺の組合わせと仮定する．

仮説の枝刈り　仮説の枝刈りは明らかに誤りのある仮説を捨てる処理であり，解を早く見付けるためであるとともに，不必要な仮説を捨てることによりメモリを効率よく使うためでもある．

仮説の修正の場合と同様に予想したq_gに対して$q_i \in V$，$q_l \in V_{q_i}$ $(i \neq g)$である節点q_iにq_lから到達したとする．この場合もq_lの仮定が間違っていたということであるから，待ち行列Q_lの先頭の仮説を捨て残りの仮説に対して次のように枝刈り

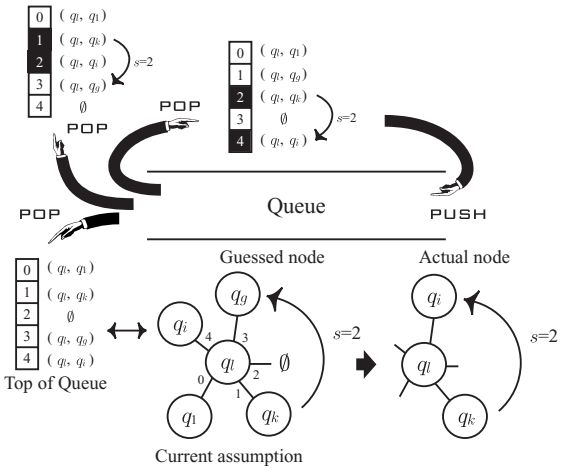

Fig. 5　Pruning assumptions

を行う．また，予想が当たった場合においても，上記の理由により，先頭以外の仮説に対して枝刈りを行う．

q_l には q_k から来たものとする．Q_l の仮説を1つ取り出してこの仮説が $a' = \alpha(q_l, (q_l, q_k)) \oplus s$ で $\epsilon(q_l, a') = (q_l, q_i)$ であるならば Q_l に戻す．そうでなければこの仮説は捨てる．最後に Q_l の先頭の仮説を q_l の辺の組合わせと仮定する．

3.1.2 次の枝路の選択と予想

効率よくトポロジカルマップを構築する為に，転回点で次の枝路を以下のように選択する．

節点 q_i の待ち行列 Q_i に含まれている仮説の総数を $d(Q_i)$ と表したとき，この $d(Q_i)$ の値により分けて考える．

$d(Q_i) > 1$ である場合　q_i の現在の辺の組合わせは仮説に基づく仮定であるから，その仮定が正しいかどうかを調べる為に，$\epsilon(q_i, a) \neq \emptyset$ となる方向 a の中から任意に一つ選ぶ．つまり，行ったことのある方向を選んで，予想通りの結果になるかを調べる．

$d(Q_i) = 1$ である場合　q_i の現在の辺の組合わせは仮定ではないから，$\epsilon(q_i, a) = \emptyset$ となる方向 a の中から任意に一つ選ぶ．つまり，まだ行ったことのない方向を選択する．

もし，全ての方向 a が $\epsilon(q_i, a) \neq \emptyset$ であれば，q_i の辺の繋がりが全て分かったということであるから，隣接する節点 q_j の中から $d(Q_j) > 1$ となる全ての方向の中から任意に一つ選ぶ．つまり，仮説を複数個持っている節点を積極的に選んで正しい解を早く見付ける．

また，隣接する節点 q_j の全てが $d(Q_j) = 1$ であるならば，どの方向を選んでも構わないので，行き止まりでない方向を任意に一つ選ぶ．

上のアルゴリズムを用いて方向 a を選択した後，$\delta(q_i, a)$ により次に現われる転回点を予想する．

4. シミュレーション

今回は簡単に本手法の有効性を確認する為に，環境として単純無向グラフで用意した．ロボットが節点に到達するとその節点の識別番号と結ばれている辺の数が分かるようにしてある．

Fig. 6の環境は同じ識別番号を持つ q_0 の節点，つまり行き止まりが一つの節点に多数結ばれている場合においても，本手法でマップを生成できるかどうかを調べる為に用意した環境であるが，様々な初期値に対して何の問題なしに環境地図を生成することができた．これは今回のシミュレーションで使用した言語がJavaであり，各節点や各辺に対してそれぞれ一つのオブジェクトを生成するようにプログラムしたからで，行き止まりは節点で表せば全て q_0 となるにもかかわらず，各節点でどの方向にいるのかを判断できるのかは，各節点 q_i において今どの方向に自分がいるのかを結ばれている節点ではなくて，$\alpha(q_i, (q_i, q_0))$ のように結ばれている辺で判断していることと，(q_i, q_0) で表せば q_i とその節点に隣接する全ての行き止まりの節点

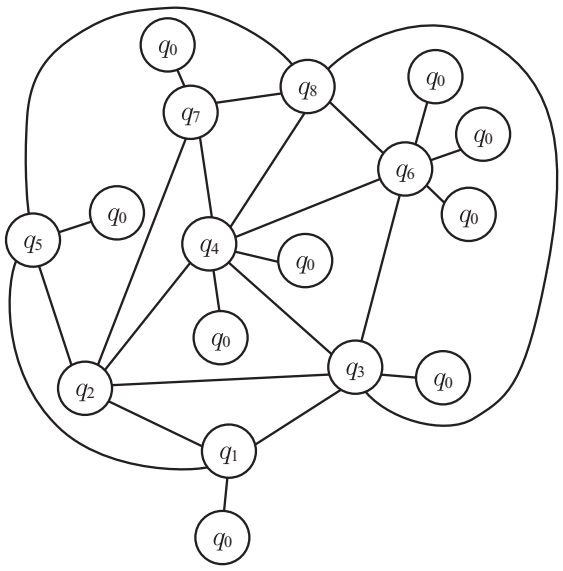

Fig. 6　Environment (1)

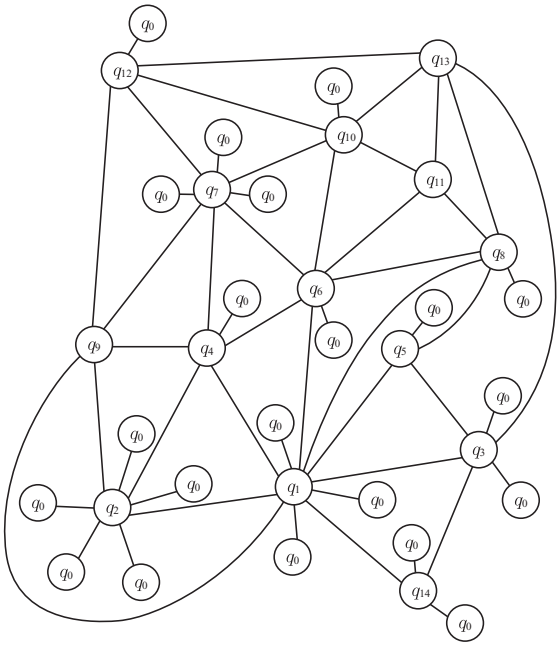

Fig. 7　Environment (2)

q_0 に結ばれている辺は同じものに見えるが，Java 上のオブジェクトとしては異なるので区別ができている．これは Java のようなオブジェクト指向の言語を用いたからできたというのではなく，オブジェクトのアドレスが辺のラベルとなって結局はラベル付きの無向グラフを作っていたからだけのことであるが，プログラムを組む上で各辺のラベルを気にする必要がなく，また動的にメモリを確保したり開放したりするのが容易に行えるので，今回のようなプログラムにはオブジェクト指向言語の方が便利である．そして，動的にメモリを確保・開放する手法を今回取りいれたことによって，今まで提案していた人工神経回路を用いた手法に比べ，無駄なメモリの利用も減り，記憶領域が固定であったという問題も解決された．

Fig. 7の環境は更に多くの節点と行き止まりがある環境であるが，自己位置が特定できない場合がかなり増えるので完全に生成するまで多少の時間がかかるが，この場合も当然のごとく様々な初期値に対して問題なく地図を生成した．細かい結果については当日述べることにする．

5.　仮説に対する考察

本手法ではある転回点において現在どの枝路にいるのか特定できないことになっても，考えられる全ての可能性から仮説を立て，それを試してみることにより環境地図を生成する手法である．ところで，もしジャイロを搭載しある転回点でのどの方向に自分かいるのかを特定することができるならば，このようなアルゴリズムは不要かもしれない．もっと言えば自己位置を特定できるセンサを装備していれば，このようなアルゴリズムは不要であるといえる．しかし，常に自己位置を正しく特定できるのであればである．

我々人間のセンサはロボットのセンサと比較して格段に性能が高い．環境認識に関しては比較にならないほどである．それでも，我々は自己位置を特定できず道に迷うことがある．ロボットであれば尚更自己位置の特定に失敗するであろう．我々が道に迷ったとき，全く知らない土地である場合を除いて，大抵は大体どのあたりにいるのか想像する．そして，その想像から行動を決め，その行動の結果からその想像が正しかったのかを判断し，想像を修正するなどを行う．

このように一つの地図の中に様々な可能性を混在させ，それぞれを試すことによって，例え自己位置を特定できなくても，もしくは誤った特定をしたとしてもそれを修正し，正しい知識を獲得することが可能になる．それを実現するためには地図に冗長性が必要となる．従来の手法[4, 5]では人工神経回路が持っている冗長性によって実現していたわけであるが，グラフ表現にしたことによりそれが失われることになった．しかし，待ち行列に複数の仮説を用意することにより，冗長性を実現することができたのである．

以上の理由により，仮説のような冗長性を表現する手段は，エラーリカバリの見地から必要であると考える．

6. おわりに

今回は自律移動ロボットのための環境表現並びに環境地図の生成手法について述べた．この問題において難しいのは，単なるグラフ探索ではなく枝路の並び方を知らなければならないということと，枝路にはランドマークが無い為正確にどの枝路を移動しているのか分からないということである．それでも上手く生成できたのは，特定できない場合は仮説を立てさせたことと，生成した地図から次に来る転回点を予想させ実際と比較し仮説を修正できるようにしたためで，今ある情報で考えられる全ての事象を考えて，Trial and Errorで不適切と分かった事象を捨てていくことにより実現できたのである．しかし，全ての事象といっても地図全体で考えるのではなく，局所的に節点での事象を考えているので大量のメモリを必要としていないところが本手法の長所であると考えている．つまり，大域的な冗長性ではなく局所的な冗長性を実現したことにより，メモリの使用量を抑えている．今後はグラフではなく通路状の環境に対して本手法が有効かを調べていくことにする．

参考文献

1) 松本勉，油田信一：“経路地図に従った移動ロボットの自律走行システム”，日本ロボット学会誌，vol. 5，no. 5，pp. 351–359，1987.

2) Benjamin J. Kuipers and Yung-Tai Byun: “A Robust, Qualitative Approach to a Spatial Learning Mobile Robot,” SPIE Sensor Fusion : Spatial Reasoning and Scene Interpretation, vol. 1003, pp. 366–375, 1988.

3) Maja J. Mataric: “Integration of Representation Into Goal-Driven Behavior-Based Robots,” IEEE Transaction on Robotics and Automation, vol. 8, no. 3, pp. 304–312, 1992.

4) 及川一美，土谷武士：“行動規範型自律移動ロボットの世界像獲得及びナビゲー ション手法”，日本ロボット学会誌，vol. 16，no. 1，pp. 65–73，1998.

5) 及川一美，土谷武士：“手書き地図を用いた通路状環境のオフライン教示”，日本ロボット学会誌，vol. 17，no. 5，pp. 704–713，1999.