コンフィグレーション空間構造に基づく知恵の輪の難易度評価

岩瀬　亮† 　鈴木　茂樹† 　中　貴俊† 　山田　雅之‡ 　遠藤　守‡ 　宮崎　慎也‡

†中京大学大学院情報科学研究科　‡中京大学情報理工学部

Difficulty Rating of Puzzle Rings Based on Configuration Space

Akira IWASE† 　Shigeki SUZUKI† 　Takatoshi NAKA† 　Masashi YAMADA‡
Mamoru ENDO‡ 　and 　Shinya MIYAZAKI‡

† Graduate School of Computer and Cognitive Sciences, Chukyo University
‡ School of Information Science and Technology, Chukyo University
E-mail: 　† {iwase | shigeki | naka}@om.sist.chukyo-u.ac.jp, 　‡ {myamada | endoh | miyazaki}@sist.chukyo-u.ac.jp

あらまし　知恵の輪を解く際の難易度は，輪を外す手順の組み合わせの複雑さや，輪が移動可能な経路全体の構造の複雑さが主に関連していると考えられる．本論文ではこれら複雑さを表す量を仮定し，知恵の輪の難易度との関連性を明らかにすることにより，知恵の輪の難易度を定量的に評価する方法を提案する．これを実現するためには，知恵の輪の解を計算機処理により求めるためのアルゴリズムや，探索空間の構造化を実装する必要がある．本研究ではまず，実在する知恵の輪を対象としてこれを実現した．次に，シミュレーションで得られた諸量と実際に人が感じる難易度との関連性を被験者実験を通じて評価し，それらの諸量の知恵の輪の難易度としての妥当性を検証した．

Abstract　The difficulty of puzzle rings is mainly due to the complexity of the solving procedures and the dimensions of the possible moving paths. This paper aims at quantitative evaluation of the puzzle ring difficulty. We propose some feature values to express the difficulty. Then, effectiveness of those values is evaluated by comparing with how difficult we feel when we solve real puzzle rings.　To be solved puzzle rings by the computer, we have implemented data structure to express puzzle rings and algorithms to manipulate them. The search algorithm to find the optimal path and a method for characterizing the configuration space are presented.

キーワード　動作計画, 高次元コンフィグレーション空間, 知恵の輪, 難易度評価

Keyword　Motion Planning, High –Dimensional Configuration Space, Puzzle Rings, Evaluating Difficulty Rating

1. はじめに

人は試行錯誤しながら知恵の輪を解こうとするが，その過程は，同じ状態を行き来し，必ずしも効率的ではない．これは，知恵の輪の取り得る状態全体がどのような構造になっているかを理解できないことが原因と考えられる．また，多くの知恵の輪が市販されているが，現在は主観的な評価に基づき難易度レベルを設定している．そこで本研究では，知恵の輪が取り得る状態の構造を分析し,それと知恵の輪の難易度との関連性を調査し，難易度を客観的に評価する方法について検討した．

知恵の輪を解くという問題は，高次元の経路探索問題に属す．これはロボットの動作計画問題と関連するため，その分野で広く研究されており，様々な探索手法が提案されている[1][2][3]．その中には，提案された手法を評価するため，Alpha Ring と呼ばれる最も単純な知恵の輪を評価実験に用いたものも幾つかあるが[4][5][6]，とくに知恵の輪を効率良く解こうと狙ったものではなく，また，Alpha Ring 以外のより複雑な知恵の輪を解いたという報告はない．一般に知恵の輪を解くという問題は目標状態を与えないのが普通であるが，それらは初期状態と目標状態を与え，それらを結ぶ経路を求めることを行っている．これは問題を易しくしており，知恵の輪の問題を解いたとは正確にはいえない．さらに，知恵の輪が取り得る状態を調べ，その状態空間の構造を調査したものはなく，知恵の輪が経路探索問題としてどの程度難しい問題なのか明らかになっていない．

本論文では，知恵の輪の問題を，絡みを外すという観点から定義する．次に，高次元経路探索問題の観点から知恵の輪の特徴を明らかにし，効率的に解く方法について述べる．さらに，人の解決プロセスを考慮し，知恵の輪を解く過程で探索した空間の特徴量と知恵の輪の難しさとの関連性について述べる．

1.1. 知恵の輪の分類

知恵の輪には様々な種類があり,材質,人文学,幾何学など様々な観点からの分類が試みられている．ここでは幾何学的観点からの分類を紹介する[7]．知恵の輪で扱う個々の輪の形状を単純化した場合,2 つの輪の関係は図 1 のように表現できる．図 1(a)は両方の輪が閉じていて絡みが外れないもの，(b)は両方の輪に隘路（狭い隙間）があり，はじめは絡んでいるもの，(c)は一方の輪は閉じてお

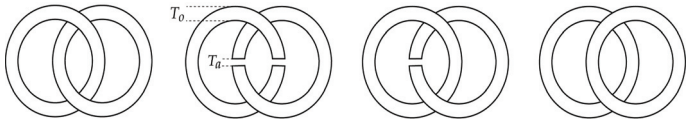


図 1　幾何学的観点からの知恵の輪の分類

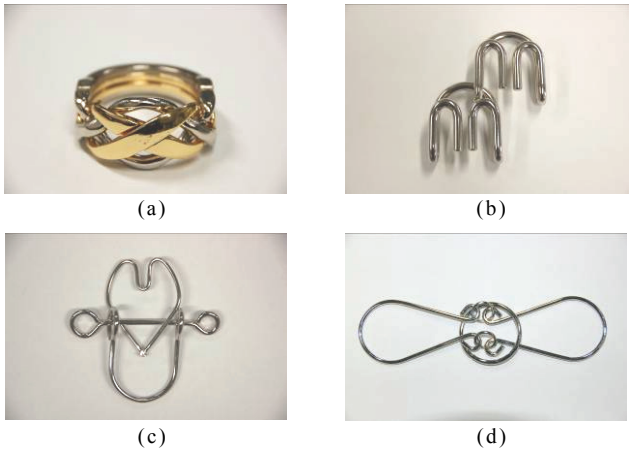


図 2　図 1 の分類に対応する実際の知恵の輪の例

り，もう一方に隘路があるもの，(d)は両方の輪が閉じており，絡みのないものを表す．それぞれに対応する知恵の輪の例を図 2 に示す．

このなかで，図 1(b)のタイプは種類も豊富で，また，比較的形状が単純であるにも関わらず，解くのが難しい．本研究ではこのタイプの知恵の輪を対象とし，以降では，このタイプを両隘路型とよぶ．両隘路型の知恵の輪の特徴は，円柱形の剛体を変形して輪が作られていること，および，輪の太さ T_o は均一で，かつ，隘路の幅 T_a より大きいことである(図 1(b))．$T_o > T_a$ であることは，円柱部分を隘路に通して外すことは不可能であり，外すためには 2 つの輪の隘路をうまく重ね合わせスライドさせる必要があることを意味する．したがって，隘路どうしが重なるように知恵の輪を移動させるという戦略が有効であり，実際，人が解く場合もこのような戦略をとっていることが観察できる．また，計算機により解く場合もこのような戦略を用いることにより，効率的に解けることが予想できる．

2. 知恵の輪の表現方法

ここでは知恵の輪を計算機上で表現するための方法を述べる．

2.1. 形状表現と交差判定

ここで対象とする両隘路型の知恵の輪は，その輪が円柱状の剛体を変形したものである．計算機上では，直径 $T_o > 0$ を持つ円柱 c_i の連結 $c_0, c_1, \ldots, c_n$ により輪の形状を表現する．これは近似表現であるが，円柱の数を十分多くすれば，知恵の輪の本質を損なうことはない．また，2 つの輪 O' と O の交差判定はそれぞれの輪を構成する円柱 c_i' と c_j の全ての組みについてその中心軸間の距離 d_{ij} を求め，$d_{ij} \le T_o$ となる組みが 1 つでも存在したら交差しているとする．

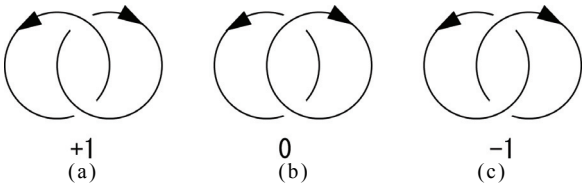


図 3　絡み数

2.2. 絡み数による判定

知恵の輪を計算機で解かせる場合，停止性を保証するためには，解けた状態か解けていない状態かを判断できることが必要である．両隘路型の知恵の輪の場合，この判断に絡み数というものが利用でき，ここでは知恵の輪の問題を，絡み数を用いて定義することを試みる．絡み数とは，2 つの輪に向きを仮定し，一方の輪が他方の輪を何回転しているかを表す数である[8]．図 3(a)(b)(c) に絡み数の例を示す．絡み数は符号をもち，一方の輪に沿って左ねじを回したとき，ねじが他方の輪の矢印の方向に進むとき＋とし（図 3(a)），逆方向に進むとき－とする(図 3(c))．また，一方の輪が他方を回転していないとき絡み数は 0 とする(図 3(b))．絡み数は複雑な閉曲線や折れ線によって近似した閉曲線についても計算する方法が知られている[9]．ここでは両隘路型の知恵の輪を対象とするが，それぞれの輪について向きと両端を結ぶ線を仮定することにより，2 つの輪の絡み数を計算し，その値により解けているか解けていないかを判断する．両隘路型の知恵の輪の場合，取り得る絡み数は＋1，0，－1 の何れかであり，＋1 または－1 ならば解けていない状態，0 ならば解けている状態となる．

2.3. 位置姿勢の表現

知恵の輪では 2 つの輪の相対的な位置関係のみが重要なので，知恵の輪の位置姿勢は一方の輪 O' を固定して考えればよい．また，3 次元空間では 1 つの剛体の自由度は 6 であることから[1]，知恵の輪の自由度は 6 である．固定しない方の輪 O については，基準位置での姿勢回転と，そのあとの平行移動との合成により，任意の位置姿勢を表現できる．ここでは基準位置での姿勢回転量をオイラー角 (a,b,c)，平行移動量を (x,y,z) とする計 6 個のパラメータの組み (x,y,z,a,b,c) で輪 O の位置姿勢を表現する．一般に，このようなパラメータの組みをコンフィグレーションとよぶ．また，各パラメータを軸とし，全てのパラメータ値が 0 となる点を原点とするパラメータ空間を，コンフィグレーション空間とよぶ[1][2][3].

2.4. 知恵の輪のコンフィグレーション空間

ここでは知恵の輪のコンフィグレーション空間の特徴について述べる．図 4 にコンフィグレーション空間の概念図を示す．

コンフィグレーション空間全体を C，輪 O' と O が交差するコンフィグレーションの集合を $C_{obstacle}$，交差しないコンフィグレーションの集合を C_{free} とする．また，絡み数が＋1，0，－1 となるコンフィグレーションの集合をそれぞれ $C_{l=+1}$，$C_{l=0}$，$C_{l=-1}$ と表すと，$C_{free}=C_{l=+1}\cup C_{l=0}\cup C_{l=-1}$ である．知恵の輪を解くという問題は初期コンフィグレーション $p(\in C_{l=+1}\ \mathrm{or}\in C_{l=-1})$ を与え，$C_{l=0}$ の任意のコンフィグレーションに至る経路を求めるという問題に対応する．図 5(a),(b)には初期コンフィグレーション，(c)には解けた状態すなわち $C_{l=0}$ に属すコンフィグレーションの例を示す．知恵の輪のコンフィグレーション空間には以下のような特徴がある．

① ここで対象とする知恵の輪は，隘路があることから，初期コンフィグレーション p から $C_{l=0}$ に至る経路は $C_{obstacle}$ で挟まれた非常に狭い道を通る必要がある．以降，この狭い道を C_{narrow} と表す．

② $C_{l=+1}$ のコンフィグレーションから $C_{l=-1}$ のコンフィグレーションへ至るには，必ず $C_{l=0}$ のコンフィグレーション，すなわち解けた状態を経由する必要がある．逆にいえば，もし初期コンフィグレーション p が $C_{l=+1}$ に属すならば知恵の輪を解くのに $C_{l=-1}$ に属すコンフィグレーションは考慮する必要はなく $C_{l=+1}$ 内での連続な移動のみ考えればよい．

③ $C_{l=+1}$，$C_{l=-1}$ はそれぞれ必ずしも全てが連続ではなく $C_{obstacle}$ に囲まれ孤立した領域も存在する．図 5(d)に示すコンフィグレーションは絡み数が＋1であるが，実際にはこのような状態は取りえない．

④ 知恵の輪の場合，$C_{l=0}$ は開集合であり，$C_{l=+1}$ と $C_{l=-1}$ は閉集合である．

上記①より，知恵の輪を解く経路を見つけるためには，狭い道 C_{narrow} におけるコンフィグレーションを見つける必要がある．また，②，③から初期コンフィグレーション p から連続的に遷移可能で，かつ，p と同じ絡み数をもつコンフィグレーションを探索すれば解に至る経路が見つかることが分かる．また，④より，目標コンフィグレーションが陽に与えられない場合，すなわち，解けた状態のコンフィグレーションが分かっていない場合，$C_{l=0}$ 側から初期コンフィグレーションへ至る経路を見つけるより，初期コンフィグレーションから $C_{l=0}$ の任意の点へ至る経路を見つける方が効率がよいと考えられる．

3. 知恵の輪を解くアルゴリズム

知恵の輪の難易度を評価するためには，その解経路の特徴を調べることが必要である．また，解経路のみでは人が解く際に試行錯誤する現象を解明できないので，探索空間を調べることも必要である．ただし，その探索空間とは初期状態から遷移可能な状態の集合であれば十分ある．これらを考慮し，ここでは Rapidly-Exploring Random Trees (RRT)構成アルゴリズム[4][10]を知恵の輪の解経路探索手法として採用した．

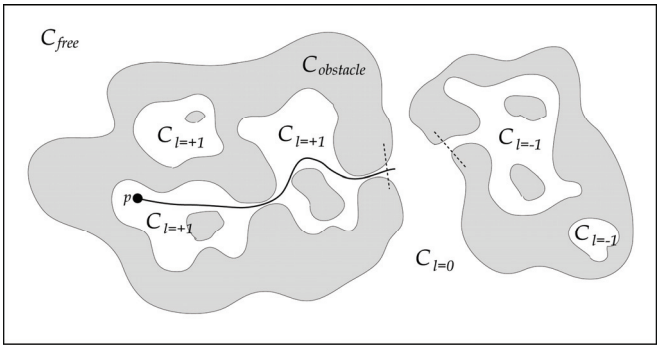


図 4 コンフィグレーション空間の概念図

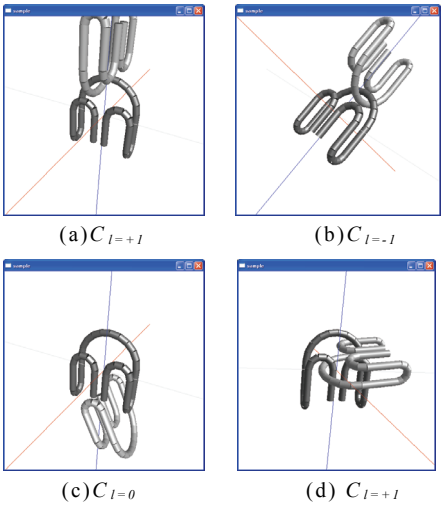


図 5 (a) (b)初期コンフィグレーションの例，(c)解けた状態の例，(d) 不可能状態の例

3.1. RRT 構成アルゴリズム

RRT 構成アルゴリズムは初期状態から遷移可能な状態を逐次調べていきながら，その範囲を一様に広げる性質があり，知恵の輪を解くのに向いている．また，インプリメントも易しい．

RRT 構成アルゴリズムは以下の処理を，木を構成する頂点数が上限値 K に達するか，木が目標条件を満たすまで行う．

(1) 初期コンフィグレーション p_{init} のみからなる木 T を作る．

(2) コンフィグレーション空間の 1 点 $p_{rand}\in C$ をランダムにサンプリングし，木 T の中で p_{rand} に最も近い頂点 p_{near} を見つける．次に，p_{near} から p_{rand} の方向へ p_{near} からの距離が ε となる点 p を求め，p_{near} と p の間に p_{rand} があれば p_{new} を p_{rand} とし，そうでなければ p_{new} を p とする（図 6）．

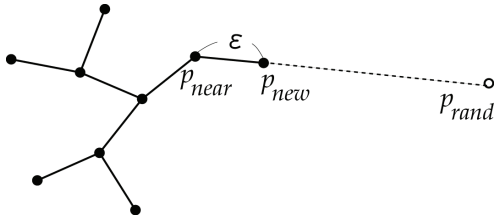


図 6 RRT 構成の様子

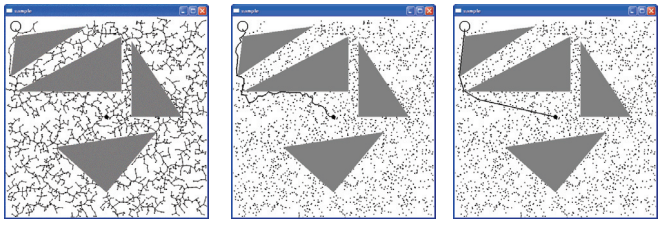
(3) さらに，p_{near}とp_{new}が接続可能ならば木に点p_{new}と枝(p_{near}, p_{new})を追加する．ここで，2つのコンフィグレーションが接続可能とは2点を結ぶコンフィグレーション空間中の直線が$C_{obstacle}$と交差しないときをいうものとする．
(4) 追加されたp_{new}が目標条件を満たすなら処理を終了する．そうでなければ(2)へ戻る．

このアルゴリズムにより構成される木Tの頂点は，C_{free}全体に一様に広がる性質がある[4]．また，初期コンフィグレーションから各頂点への経路が木の枝として記録される．理解を容易にするため，図 7(a)には，2次元コンフィグレーション空間において，●で示す初期コンフィグレーションから○で囲まれた目標領域に至る経路をRRT構成アルゴリズムにより求めたとき，生成された木を示す．なお，処理は目標領域に至った時点で停止するものとした．図 7(b)には目標領域に入ったコンフィグレーションから初期コンフィグレーションまで木を遡って得られる経路を示す．木から直接得られる経路は滑らかではないので，経路長が最小になるよう最適化したものを図 7(c)に示す．この最適化では(b)で示す経路上の全ての点からなる完全グラフを作り，その完全グラフの全枝について距離を求め最短経路を求めている．なお，$C_{obstacle}$をまたぐ枝の距離は∞とする．以降ではこのように最適化した解経路を最適経路とよぶ．

3.2. RRT構成アルゴリズムのカスタマイズ

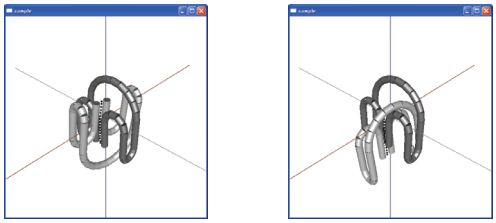
上で述べたように，RRT構成アルゴリズムは知恵の輪を解く経路を探索するのに都合がよい．しかしながら，このアルゴリズムは各ステップで最近点を求める必要がある．単純な最近点アルゴリズムはO(N)なのでRRT構成アルゴリズムは頂点数が増えるにつれ処理が急速に遅くなる．よって，複雑な知恵の輪を現実的な時間内で解くためには，生成する頂点数をできる限り抑え，スピードダウンしない工夫が必要である．以下では，知恵の輪の特徴を考慮し，生成頂点数を抑える工夫について述べる．

先に述べた形状の性質（隘路より輪が太いこと）より，知恵の輪を解くには，2つの輪の隘路を重ねながらスライドさせる必要がある．そこでこのような状態のコンフィグレーションpを予め用意し，木Tがそのコンフィグレーションにある



(a) (b) (c)

図 7 RRT構成アルゴリズムの実行例:(a)生成される木，(b)目標領域から初期コンフィグレーションに至る枝，(c)最適化した経路



(a) (b)

図 8 (a) (b)隘路コンフィグレーションの例

程度近づいたらTとpが接続可能かどうか調べ，可能なら，pをTに追加するようアルゴリズムを拡張した．これより，生成頂点数を抑えることができ，実際実験では現実的な時間内で解けなかったものが解けるようになった．また，これは人が解く場合に用いる戦略に対応していることから，この方法で得られた結果のほうが精度よく難易度評価できると考える．

2つの輪の隘路が重なるような状態のコンフィグレーション（以降，これを単に隘路コンフィグレーションとよぶ）の例を図 8 に示す．隘路コンフィグレーションは輪の形状から自動的に求めることができるが，紙面の都合上その方法の説明は省略する．

知恵の輪を解くためにRRT構成アルゴリズムをカスタマイズしたものを図 9 に示す．なお，Aはあらかじめ用意した隘路コンフィグレーションの集合を表す．

アルゴリズムの終了条件は木を構成する頂点数が上限値に達するか，または，新しく木に追加される頂点の絡み数が0となった場合とする．関数CONNECT_NCは新しく追加された点p_{new}に最も

```
BUILD_RRT for PUZZULERING(p_init, A)
1.  T.init(p_init);
2.  CONNECT_NC(p_init, T, A);
3.  while T.size < K do
4.      p_rand ← RANDOM_CONFIG();
5.      p_near ← NEAREST_NEIGHBOR(p_rand, T);
6.      if NEW_CONFIG(p_rand, p_near, p_new) then
7.          T.add_vertex(p_new);
8.          T.add_edge(p_near, p_new);
9.          if LINKNUMBER(p_new) = 0 then
10.             return true;
11.         else
12.             CONNECT_NC(p_new, T, A);
13. return false;

CONNECT_NC(p_new, T, A)
1.  p_narrow ← NEAREST_NEIGHBOR(p_new, A);
2.  if (DIST(p_new, p_narrow) < r) and
    FREE_PATH(p_new, p_narrow) then
3.      T.add_vertex(p_narrow);
4.      T.add_edge(p_new, p_narrow);
5.      A.delete(p_narrow);
6.      CONNECT_NC(p_narrow, T, A);
```

図 9 知恵の輪を解くアルゴリズム

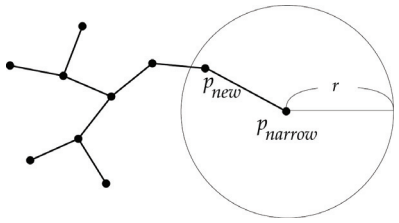


図 10 木に隘路コンフィグレーションを接続する様子

近い隘路コンフィグレーション p_{narrow} を A から見つけ，その距離が r 未満で，かつ，接続可能なら木 T に点 p_{narrow} と枝(p_{new}, p_{narrow})を追加し A から p_{narrow} を削除する処理を再帰的に行う(図 10)．関数 NEW_CONFIG は 3.1(2)の処理を行い，p_{near} と p_{new} が接続可能ならば真を返す．

4. 知恵の輪の難易度評価

ここでは知恵の輪の難易度評価方法について述べる．まず，被験者実験を行い，知恵の輪の難しさの要因に関する考察を述べる．次に，その要因と解経路探索により得られた探索空間の特徴量との関連性について述べる．

4.1. 被験者実験

本研究では 9 人の被験者に 3 種類の知恵の輪を実際に解いてもらい，解くまでにかかった時間を数回に渡り計測した．以下に実験方法の詳細と結果について述べる．

10 分の時間制限を設け，難易度の異なる 3 種類の知恵の輪（alpha,dalpha,devil）を任意の順番で解いてもらい，解けるまでの時間を計測した．計測は 1 週間の間隔をあけながら計 3 回行い，各回終了後には，全ての被験者に解き方を見せた．

結果を図 11 のグラフに示す．なお，便宜上解けなかった場合は 600 秒としてプロットしてある．

1 回目の結果を見ると，alpha は全ての被験者が比較的速く解けたのに対し，dalpha と devil は解くのに時間がかかり，制限時間内で解けない被験者もいた．この結果は，経路の候補が多く存在し，どれが解ける経路なのかを予測することが難しいことが要因と考えられる．

2 回目，3 回目の結果を見ると，devil は回を重ねるごとに解く時間が短くなる傾向がある．一方，dalpha は回を重ねても解く時間が短くなっていない．毎回終了後に解き方を教えているにもかかわらず，dalpha のように回を重ねても時間が短くならないのは，大まかな経路は知っていてもその経路にしたがって移動するのが難しいことが要因と推測する．

表 1 探索した頂点数と最適経路上の頂点数

Sv: 全頂点数， Pv: 最適経路の頂点数

alpha			dalpha			devil		
	Sv	Pv		Sv	Pv		Sv	Pv
1	466	9	1	11759	18	1	41604	19
2	377	11	2	1325	16	2	25463	22
3	1000	9	3	3238	15	3	41366	15
4	2104	8	4	8267	17	4	61799	23
5	581	9	5	16354	15	5	33825	24
6	7315	9	6	3866	21	6	24401	15
7	412	10	7	15725	18	7	32703	27
8	599	10	8	30725	18	8	42474	23
9	503	8	9	19032	20	9	34830	17
10	8380	7	10	6964	17	10	30237	26
平均	2174	9	平均	11726	18	平均	36870	21

以上の結果より，知恵の輪の難しさの要因として，次の 2 種類が存在すると考えられる．

(E1) 大まかな経路を予測する難しさ

(E2) 予測した経路に従って移動することの難しさ

本研究では，これらに関連する探索空間の特徴量を調査した．以降では，E1 に関連する「最適経路頂点数」，「探索空間のクラスタ数」および E2 に関連する「解経路の可視率」について述べる．

4.2. 最適経路頂点数

3 章で述べたアルゴリズムを用い，3 種類の知恵の輪について，乱数値を変え 10 回解かせた結果を表 1 に示す．表 1 において，Svは生成された木の頂点数，Pvは最適経路における頂点数である．図 12 は 3 種類の知恵の輪について，それぞれ 10, 5, 6 回目に得られた最適経路における頂点の状態を表示したものである．ここで用いたアルゴリズムは確率的手法であることから，表 1 に示すように，解けた時点での木の頂点数にばらつきが見られるが，平均的に，alpha，dalpha，devilの順に多くなっている．これは初期コンフィグレーションから遷移可能なコンフィグレーションの多さに対応し，実質的な探索空間の広さを意味する．最適経路の頂点数は経路の非線形性の高さを表し，その数が多いほど複雑な経路といえる．10 回解かせて得られた最適経路のうち最も頂点数が少ないものに着目すると，alpahは 7，dalphaとdevilはともに 15 であることから，alphaは他の 2 つに比べ経路が単純で，dalphaとdevilの経路はほぼ等しい複雑さと考

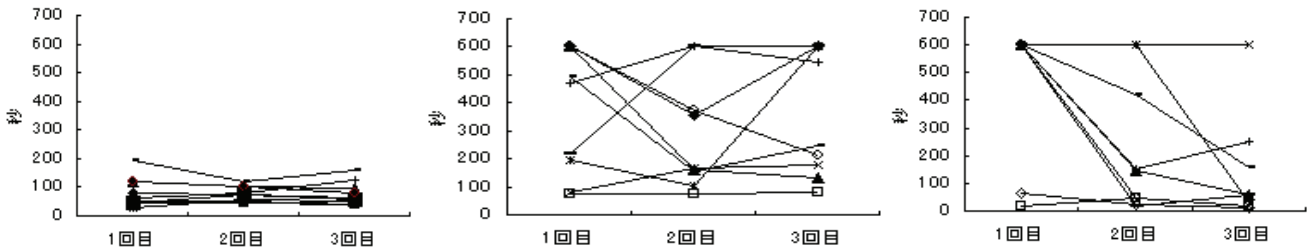


図 11 被験者が解くのにかかった時間：alpha(左)，dalpha(中)，devil(右)

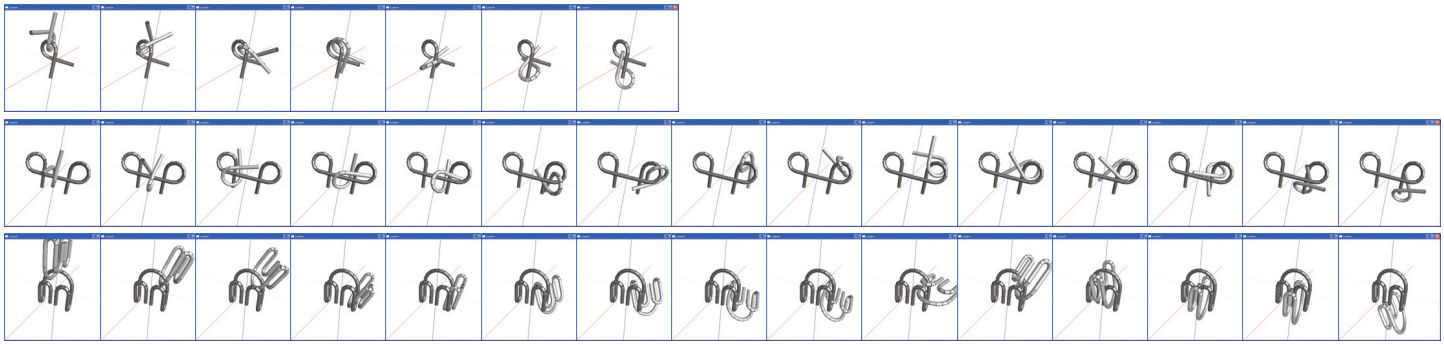


図 12　最適経路の表示：alpha(上段)，dalpha(中段)，devil(下段)

えることができる．複雑な経路ほど予測するのは難しいので，最適経路頂点数はE1 に関連する特徴量と考えられる．なお，知恵の輪を解くのにかかった平均の時間は，CPU:Athlon-3200+，Memory: 1024MByte，OS:Windows XP Professional を用いた場合，alpha，dalpha，devilはそれぞれ約 18 秒，646 秒，2760 秒であった．

4.3. コンフィグレーション空間の構造化

知恵の輪の難易度を評価するとき，迷い込む道の有無や数を考慮する必要がある．このためには，知恵の輪の取り得る状態の集合を構造化する必要があり，ここではコンフィグレーション空間を対象とした構造化を行う．人間はある程度，状態を構造化（抽象化）して考えていると思われるが，構造化の基準が人の直感に反するものではいけない．これらを考慮し，ここでは，接続可能なコンフィグレーションどうしをクラスタ化し，さらにクラスタ間の隣接性を調べ，構造化を行った．以下にその方法について述べる．

クラスタ：クラスタはクリークとする[11]．すなわち，同一クラスタ内の任意の 2 点は接続可能であるものとする．

クラスタリングアルゴリズム：上記のようなクラスタを求めるには，クラスタ間距離が最長距離法に従えばよい[12]．すなわち 2 つのクラスタ C，C' の距離 $D(\mathrm{C},\mathrm{C}')$ は次のようにする．

$$D(\mathrm{C},\mathrm{C}') = \max D(x_i, x_j) \quad (x_i \in \mathrm{C}, x_j \in \mathrm{C}')$$

クラスタリングアルゴリズムは次のような一般的な階層的手法を用いた．

1. 1 個の点だけを含む N 個のクラスタからはじめる．
2. クラスタ間距離 D が最小となるクラスタの組み (C,C') を求める．ただし，$D(\mathrm{C},\mathrm{C}') = \infty$ なら処理を終了する．
3. 2 で求めた 2 つのクラスタを併合して 1 つのクラスタとし，2 へ戻る．

クラスタの隣接関係：クラスタ間の隣接関係は次のように定義する．クラスタ C と C'が隣接するとは，その要素間の最小距離 $\min D(x_i, x_j) \quad (x_i \in \mathrm{C}, x_j \in \mathrm{C}')$ が閾値以下であるか，または，RRTの木の枝でその両端が x_i，x_j であるような各要素の組みが存在するときをいう．

理解を容易にするため，図 7(a)の頂点を構造化したものを図 13 に示す．図 13 では，各クラスタ

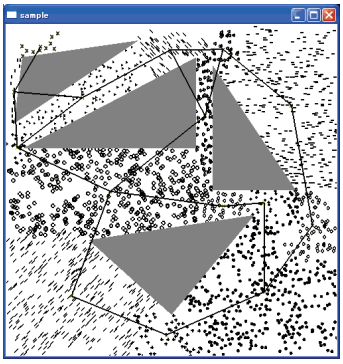


図 13　図 7(a)の C 空間の構造

表 2 探索空間構造に関するデータ

	alpha	dalpha	devil
全クラスタ数	505	884	10482
3 点以上のクラスタ数	488	853	1149
5 点以上のクラスタ数	450	807	736
10 点以上のクラスタ数	337	637	422
平均隣接クラスタ数	3.9	4.2	2.2
解経路が通るクラスタ数	7	14	13
解経路クラスタに隣接するクラスタ数	14	41	55

に含まれる点を，クラスタ別に異なる記号で表す．また，クラスタの隣接関係を枝で表現している．この例では 15 のクラスタが得られている．

3 種類の知恵の輪について，探索した点集合を構造化した結果を表 2 に示す．今回の構造化では，知恵の輪の取り得る状態をクラスタという単位で分類している．よって，クラスタ数の多さは，取り得る状態の種類の多さに対応する．クラスタはその成分数によって大小があるが，3 点以上を含むクラスタ数を見ると，alpha に比べ，dalpha や devil は大きい値になっている．また，解経路に隣接するクラスタ数も大きい値になっており，これは迷い道が多いことを意味する．また，これらは被験者実験の 1 回目の結果とも対応することから，E1 と関連する特徴量と考えられる．

4.4. 解経路の可視率

解経路上のある点から次の点への移動のしやすさを表す量として解経路の可視率を定義する．ここで可視とは，3 章で述べた接続可能と同意である．図 14 において，○は最適経路上の頂点を表し，

●は探索過程で生成された解経路以外の頂点を表す．また，破線矢印は可視，実線矢印は不可視を表す．解経路上の頂点V_iについて，V_iから可視な点を求め，その数をNとする．また，それらの点の中から頂点V_{i+1}が可視な点の数を調べ，Mとする．可視率はM/Nとする．可視率が小さい場合，移動可能な範囲が多くあり，かつ，その中で解経路に戻れる移動が少ないことに対応することから，大まかな経路が分かっていても，解経路をたどるのが難しいことに関連すると考えられる．

3 種類の知恵の輪について，解経路上の各点の可視率を求めた結果を図 15 に示す．dalpha は可視率が平均的に小さく，さらに可視率が極端に小さい点が多くある．被験者実験では dalpha の場合，2 回目，3 回目も解く時間が改善されなかったが，この結果と可視率が対応している．以上より解経路の可視率は E2 に関連する特徴量と考えられる．

5. おわりに

本論文では，知恵の輪の難易度を定量的に評価するための方法について述べた．この評価には知恵の輪を計算機により解く必要があり，そのためのアルゴリズムを示した．また，被験者実験の結果を考察し，知恵の輪の難しさの要因として【大まかな経路を予測する難しさ】,【予測した経路にしたがって移動しようとする難しさ】をあげ，前者については,「最適経路頂点数」や「探索空間のクラスタ数」，後者については「解経路の可視率」が関連していることを示した．なお，これらの実験で用いた 3 種類の知恵の輪は，上記の「難しさの要因」に関する仮説を裏付ける結果が顕著に現れるものを我々が所持する約 20 種類の知恵の輪の中から選んだものであり，ここではこれら 3 種類の知恵の輪に関する実験結果のみを示したが，他の知恵の輪についても上記仮説および難易度評価方法は有効であると考える．しかしながら統計的有意性を示すためには，より多くの知恵の輪について，解経路やコンフィグレーション空間構造を調査すると共に，被験者実験の人数を増やし，長期的に学習効果を調べる必要があり，これらを今後の課題とする．また，知恵の輪を解くアルゴリズムや難易度評価方法は知恵の輪のデザインへ応用できることから，知恵の輪の自動デザインなども今後研究したい．その他，高次元コンフィグレーション空間構造の可視化方法を検討し，知恵の輪などのコンフィグレーション空間構造を直感的に理解できるようにすることも興味深いテーマであり，今後検討したい．

謝　辞

第 22 回 NICOGRAPH 論文コンテストならびに本論文の論文審査において貴重なコメントを下さいました査読者の方々に感謝いたします．また本研究にご意見を頂いた，株式会社ハナヤマに感謝します．本研究の一部は，文部科学省科学研究費補助金および，私立大学ハイテク・リサーチ・センター補助金による．

文　献

[1] 太田順,倉林大輔,新井民夫,知能ロボット入門 ー動作計画問題の解法ー,コロナ社，東京, 2001.

[2] J.C.Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, MA, 1991.

[3] M.ドバーグ, M. フォン.クリベルド, M.オーバマーズ, O.シュワルツコップ, コンピュータ・ジオメトリ，近代科学社，東京, 2000.

[4] J. J. Kuffner, S. M. LaValle, RRT-connect: An efficient approach to single-query path planning. In Proc. IEEE Int'l Conf. on Robotics and Automation, pp.995-1001, 2000.

[5] Nancy M.Amato, O.Burchan Bayazit, Lucia K. Dale, Christopher Jones, and Daniel Vallejo, Choosing Good Distance Metrics and Local Planners for Probablistic Roadmap Methods. IEEE Transactions on Robotics and Automation, vol.16, no.4, pp.442-447, August 2000.

[6] P. Isto, A Parallel Motion Planner for Systems with Many Degrees of Freedom, International Conference on Advanced Robotics, pp.339-344, 2001.

[7] 秋山久義,知恵の輪読本,新紀元社，東京, 2003.

[8] 河内明夫(編著), 結び目理論，シュプリンガー・フェアラーク東京，東京, 1990.

[9] T.Banchoff, Self Linking Number of Space Polygons, Indiana University Mathematics Journal, vol.25, no.12 , 1976.

[10] S. M. LaValle, Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[11] 惠羅博，土屋守正，グラフ理論， 産業図書，東京, 1996.

[12] 神嶌敏弘,“データマイニング分野のクラスタリング手法(1), ”人工知能学会誌，vol.18, no.1, pp.59-65, 2003.

[13] 岩瀬亮，鈴木茂樹，中貴俊，山田雅之，遠藤守，宮崎慎也:コンフィグレーション空間構造に基づく知恵の輪の難易度評価，第 22 回 NICOGRAPH 論文コンテスト予稿集（CDROM）．

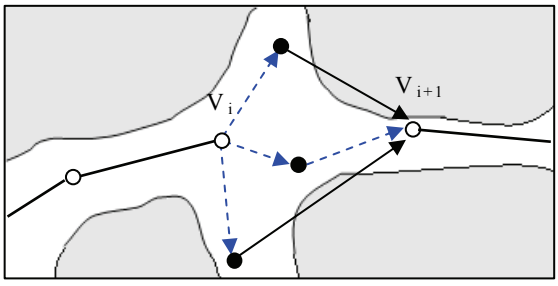


図 14　可視率の概念図

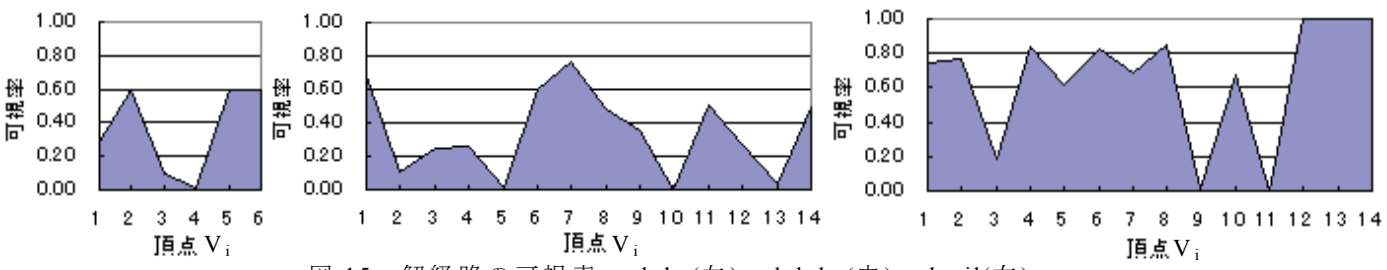


図 15　解経路の可視率：alpha(左)，dalpha(中)，devil(右)