騎士巡歴問題の最大解の記録更新に挑む

奥野 政勝* 宮野英次**
(*(有)ゾマ **九州工業大学)

1 はじめに

無向グラフを $G = (V, E)$ とする. ハミルトン閉路とは, 頂点集合 V の全ての頂点を1度ずつ通る閉路のことである. 入力グラフ G がハミルトン閉路を持つか否かを問うハミルトン閉路問題は代表的な NP 完全問題の1つである [4]. しかし, 実用的な応用も多い重要な問題であるため, 多くのヒューリスティック手法が提案されている. 本稿では, ハミルトン閉路問題の特殊ケースである**騎士巡歴問題**(Knight Tour Problem, KT 問題) [1] について扱う. KT 問題は, チェスを使った経路探索問題の 1 つであり, 8×8のチェスボードで全てのマス目を通るような騎士(ナイト)の経路を求める問題である. さらに, より一般の大きなボード上のナイト経路を求める問題が考えられている [1].

KT 問題は古くからあるパズルで, コンピューターの出現を待つ以前に手動で多彩な解が得られてきた. その主眼は主に美しい解の追及にあり, 手動での大きなボードの解は困難であった. その後, コンピュータによる深さ優先探索を基本とするバックトラックアルゴリズムの教材とされてきたが, 提示されるプログラムは8×8のボードの解をなんとか得る程度で、大きな解への挑戦の形跡はほとんどない.

本研究では, 出来るだけ大きなボード上の解を高速に求める手法の開発を目指す. 本稿では各頂点における隣接頂点の訪問順序ついて異なる 2 つのヒューリスティック手法を考える. (1) 1 つ目は, すべての頂点について, 隣接する 8 つの頂点の訪問順序を同一とする方法である. (2) 2 つ目は, それぞれの頂点について, 異なる訪問順序を用いる方法である. ただし, いずれの場合も, 経路探索を始める前に訪問順序については固定するものとする.

本稿では, 上記の 2 つの探索順序を考え, それらの計算機実装による結果を示す. (1)の訪問順序の場合, 304×304の大きさのボードまでナイト経路を求めることができる(具体的な経路は発表時に示す). (2)の訪問順序について, 個々の頂点の訪問順序をランダムに選択した場合, 48×48 までのナイト経路を求めることができる.

2 提案アルゴリズム

本稿では, ある偶数×偶数の頂点からなる格子ボードから, ナイトが動くことができる隣接頂点間に辺を考えた無向グラフ $G = (V, E)$ を入力とする. また, G の 2 点を Start(始点)と End(終点)として, すべての頂点を正確に 1 度訪問するような Start から End への経路を探索するためのアルゴリズムを考える. ただし, 今回提案するアルゴリズムは任意のグラフについても動作することに注意してほしい.

2.1 アルゴリズムの基本戦略

入力グラフの頂点次数(頂点に隣接する頂点数)は, アルゴリズムを開始する前には最大で 8 である. 経路選択では, (1) どの隣接頂点に移動するかを選択して, (2) 深さ優先順で移動を行い, (3) 経路選択が失敗の場合はバックトラックする. また, 頂点 u から頂点 v, 続いて頂点 w に移動した場合には頂点 v については辺 uv, 辺 vw 以外の辺をグラフから削除する. つまり, 経路探索アルゴリズムの基本は, 終点までの経路候補を減らしながら移動を行う.

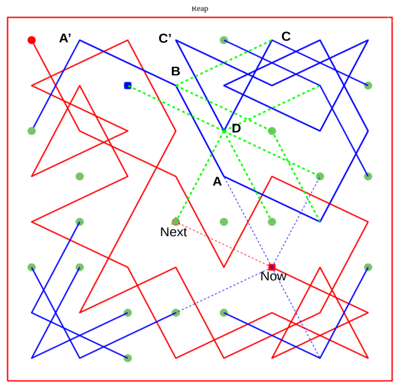

図 1: 8×8ボード上でのアルゴリズムの動作例

提案するアルゴリズムは枝刈りによるバックトラック法に基づくが, さらに, 次の 2 つの主要戦略を用いる. **(隣接頂点訪問順序戦略)** 各頂点における隣接頂点への訪問順序は, 開始時に決定するが, (1) 全ての頂点について, 同一の訪問順序とする方法と, (2) それぞれの頂点について, 異なる訪問順序とする方法を考える. **(辺削削除戦略)** 上述のように, u, v, w の順に頂点を訪問した場合, 経路上の頂点 v が持つ辺 uv, 辺 vw 以外の辺を候補から削除する. さらに, 経路上以外の頂点の辺も候補から削除できる場合がある. 頂点 p とそれに接続する 2 つの未訪問頂点 q, q' があり, q と q'の次数が2であると仮定する. もし p から q と q'以外の頂点に移動した場合, q と q'の経路候補になる辺がそれぞれ1となり, 最終的なナイト経路を形成することができない. そのため, p の次数が 3 以上であっても, この時点で辺 pq と辺 pq'以外の辺を削除することができる.

ここで, アルゴリズムの動作例を示す(図1を参照). アルゴリズムは深さ優先のバックトラック法を基本として, 経路の行き止まりである Dead End を発見して枝刈りを行う. 図1では Start を左上角の赤丸で, End を青丸で表す. 探索は, ある時点での訪問頂点を表す Now(赤四角)から次の訪問候補頂点 Next(赤三角)への移動の有効性を調べ, 実際に移動するか, 移動せずに次の訪問候補頂点を調べるかを決定する. 図1において Now から Next へ移動を進めようとすると、Now の隣接頂点の辺が削除される。Now からの青い破線が辺削除を表している. その結果, 次数が3であった頂点 A の次数が2となる. 頂点 A と A' の次数が2なので, 間にある B は, 上述の辺削除戦略により次数2にできる. すなわち、残りの辺は削除できる。削除した辺が緑の破線である. さらに, C の次数が 3 から 2 となり, CとC' の間にある D も辺削除戦略により次数2とできる. この辺削除は連鎖的に起こり, 緑の破線の 7 辺が削除される. これにより, 分岐の無い青い線で表した単純道が増えていく. 単純道上における頂点の隣接頂点訪問順序は一択であるため, 探索の効率は加速度的に上がることになる. 辺削除の結果 Dead End が生じる場合がある. Dead End とは, 以降どのような経路を選んでも end に到達できない状態である. 提案アルゴリズムでは3種類の Dead End を調べて深さ優先探索でのバックトラックを行う. 図2で大きい青丸が Dead

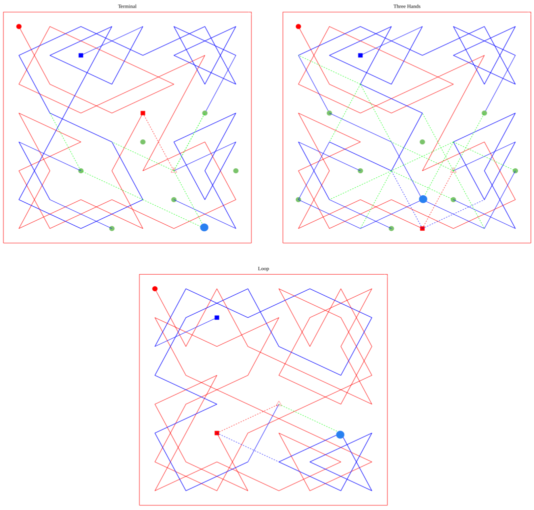

図 2：（左上）Terminal，（右上）Tree，（下）Loop

End を表す．（Terminal）編削除により次数 1 の頂点が出来ると経路探索の失敗となる．（Three Hands）ある頂点から 3 本以上の単純道ができると，少なくとも1本の単純道はナイト経路の一部にはならない．（Loop）孤立した閉路ができた場合は探索を打ち切りバックトラックすることができる．

計算機実装実験により，10×10程度のボードの場合は Dead End が起きず，バックトラックをすることがほとんどない．14×14ボードの場合のバックトラックも比較的早い段階で起こり，高速にナイト経路を発見できる確認できる．

2.2 連結成分戦略とオイラーグラフ戦略

グラフ$G = (V, E)$の任意の頂点部分集合$S \subseteq V$について，Gから Sを削除することで得られる連結成分数を $k(G-S)$ で表すとする．Gがハミルトン閉路を持つための十分条件として，$|S| \geq k(G - S)$が成り立つことが知られている [2]．これは各連結成分間を移動するための条件を表しており，この関係に動機づけられた枝刈り戦略を提案アルゴリズムは用いる．また連結成分の出入りに着目し，いわゆる一筆書きのための条件であるオイラーグラフの定理 [3] に基づく枝刈り戦略を利用する．本稿では，誘導部分グラフで，かつ，全ての頂点の次数が3以上となるものを**島**と呼ぶことにする．また，異なる島に含まれる 2 つの頂点を両端点とする単純路を**橋**と呼ぶことにする．図3では，赤い線が通過済み，緑点，青点，橙点，紫点が異なる 4 つの島を表し，青線が単純道を表す．島は連結成分であり，Next，End，島および橋について以下の(I)～(III)を満たす必要がある．

(I) 全ての島について橋の数が偶数とする．このとき，Next と End は同じ島に含まれていなければならない．

(II) ある島について橋の数が奇数であるとする．このとき，奇数となる島は正確に 2 つであり，1 つの島は Next を，もう 1 つの島は End を含んでいなければならない．

(III) 橋の無い島は存在しない．

(I)は Next としてある島に入った場合，橋の数が偶数でなれば End まで到達できないことを示している．また，Next が現時点での Start であり，各島は入ったら出る必要があり，最後の End を持つ島のみ入ることでナイト経路の終端となることを示している．(III)は島への出入りができないため経路を持たないことを示している．

アルゴリズムの計算量は増えるが，各段階で上記の戦略を用いながらバックトラックの際の早い段階での枝刈りを行い，アルゴリズムの高速化と大規模なボードまでの経路探索を可能にしている．

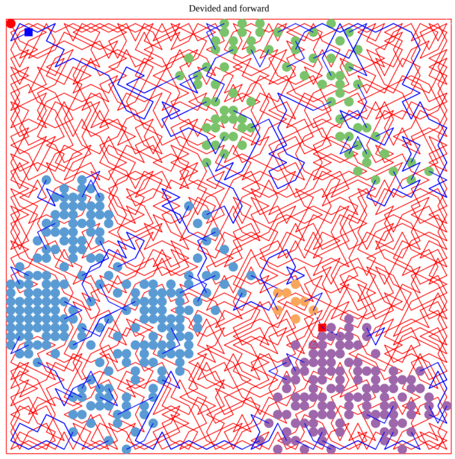

図 3： 48×48ボード上の島と橋

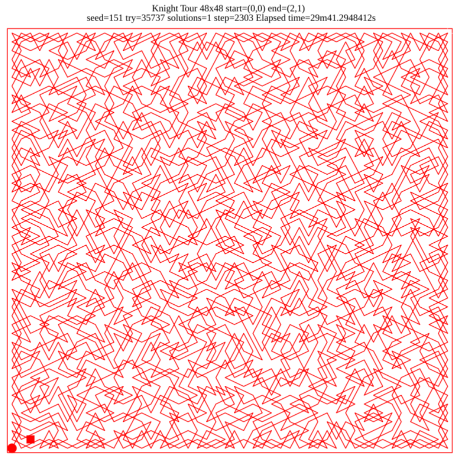

図 4： 48×48ボードのナイト経路

3 高速化の工夫と実装結果

計算機への実装において，2 つの並列化・スレッド化による高速化を行っている．(1) 1 つ目は start からいくつかの隣接頂点への移動については直列計算を行い，その時点での経路選択に従って入力グラフからの辺削除を行う．ある程度の経路選択が終わった時点からスレッド化による並列的な経路探索を行う．直列計算と並列計算にはトレードオフがあるため，効率の良い切り替えのタイミングを試みている．(2) 2 つ目は，まずすべての隣接頂点訪問順序を乱数に従って準備する．この乱数に従った順序を P 個用意する．次に P 個のそれぞれにスレッドを準備して完全非同期に並列探索を行う． 48×48や304×304の解はこれらの高速化によって得られたものである（図4参照）．

謝辞． 本研究は科研費 JP24K02902 の助成による．

参考文献

[1] Knight's tour notes (complied by George Jelliss). https://www.mayhematics.com/t/t.htm

[2] V. Chvatal. Tough graphs and Hamiltonian circuits. Discrete Math., vol.5, pp.215 – 228 (1973)

[3] L. Euler. Solutio problematis ad geometriam situs pertinentis. Comment. Acadmiae Sci. I. Petropolitanae, 8, 128–140 (1736)

[4] M.R. Garey and D.S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman (1979)