

絵画的迷路作成アルゴリズムの改善

中井亮平

岡本吉央[†]

概要

迷路のうち、解となる経路を塗ることで絵が浮かび上がるものを絵画的迷路と呼ぶ。入力として白黒 2 値画像が与えられたときに、入力画像が浮かび上がる絵画的迷路を作成する絵画的迷路作成問題に対して、岡本・上原によるアルゴリズムが知られている。本論文ではそのアルゴリズムを改良したものを紹介する。

1 はじめに

2005 年にイギリスで数独が大ブームになったことを発端に、世界中でさまざまなパズルに対する関心が高まっている。それに伴い、パズル愛好家や研究者によって、パズルの研究も盛んにされている。この研究の対象の一つが、コンピュータによるパズルの自動生成である。書店などで販売されているパズル雑誌に掲載されているパズルの大部分はパズル作家の手作業により作られたものであるが、インターネット上などでは自動生成の問題も多く見られる。

コンピュータによる自動生成の最大のメリットは、処理時間の速さであり、短時間でたくさんの問題を作成することができる。しかし生成される問題の質(解き味、見た目の美しさ、解き終わった後の達成感など)については改善すべき点が多くあり、実際、自動生成された問題よりも人の手で作られた問題の方がおもしろい、という意見も聞かれる。

そこで本論文では、質の高いパズルの自動生成、特に絵画的迷路の自動生成について考える。絵が出

ない普通の迷路の自動生成については、古くからさまざまなアルゴリズムが知られている。一方、絵画的迷路の自動生成についてはあまり研究がされていなかったが、近年岡本・上原によってアルゴリズムが提案された [1]。本研究ではこのアルゴリズムをもとに、改善されたアルゴリズムを紹介する。

2 既存アルゴリズム

まず絵が出ない普通の迷路の作成アルゴリズムを紹介する。縦 m マス、横 n マスの矩形が与えられる(図 1 左上)。各マスを頂点とみなして、縦横に隣接するマスの組を辺とするグラフを構成し(図 1 右上)、このグラフの全域木をひとつランダムに生成する(図 1 左下)。全域木の枝に対応するマス目の辺と取り除き、スタートとゴールを適当に選べば迷路が完成する(図 1 右下)。木の性質より、完成した迷路にはループが存在しない。また、スタートとゴールを結ぶ経路は一意に定まる。

つぎに絵画的な迷路の作成アルゴリズムについて考える。まず絵画的迷路構成問題を次のように定義する。

絵画的迷路作成問題

入力 縦 m ピクセル、横 n ピクセルの白黒 2 値画像

出力 解となる経路を塗ることで、入力画像が出てくる迷路

入力として与えられる縦 m ピクセル、横 n ピクセルの白黒 2 値画像の黒ピクセル全体を画像の「絵」、白黒ピクセル全体を「地」と呼ぶことにする。

東京工業大学大学院情報理工学研究所
[†]第 1 著者に同じ

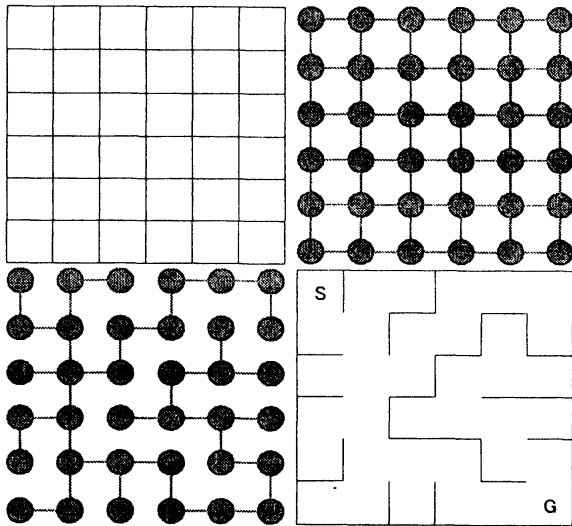


図 1: 迷路の作成方法

縦 m ピクセル、横 n ピクセルの白黒 2 値画像が与えられたとき (図 2 左)、各マス目を頂点とみなして、縦横に隣接するマスの組を辺とするグラフを構成する (図 2 右)。このグラフから全域木を構成するのだが、絵に対応するマスの頂点全体が誘導する部分グラフのみを通る経路 (ハミルトン経路) を含むことが求められる。しかし、入力画像によってはハミルトン経路が存在しない可能性がある。(図 2 はハミルトン経路が存在しない例である。) さらにハミルトン経路が存在するか判定する問題は NP 完全である。

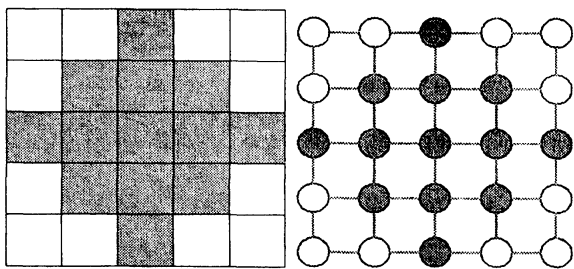


図 2: 絵画的迷路作成の難しさ

そこで、岡本・上原は、各マス目を縦横 2 等分することによってこの問題を解決している。以下アルゴリズムの概要を紹介する。

縦 m ピクセル、横 n ピクセルの白黒 2 値画像が与

えられる (図 3 上左)。この入力画像に対しては、黒ピクセル全体が 4 近傍に関して連結であることを要求しておく。各マスを頂点とみなして、縦横に隣接するマスの組を辺とするグラフを構成する (図 3 上右)。絵に対応するマスの頂点全体が誘導する部分グラフに着目し (図 3 中左)、その全域木をランダムに生成する (図 3 中右)。得られた全域木を平面的に走査し (図 3 下左)、走査した経路にしたがって縦 $2m$ マス、横 $2n$ マスの矩形上で画像の絵の部分に対するハミルトン経路が得られる (図 3 下右)。

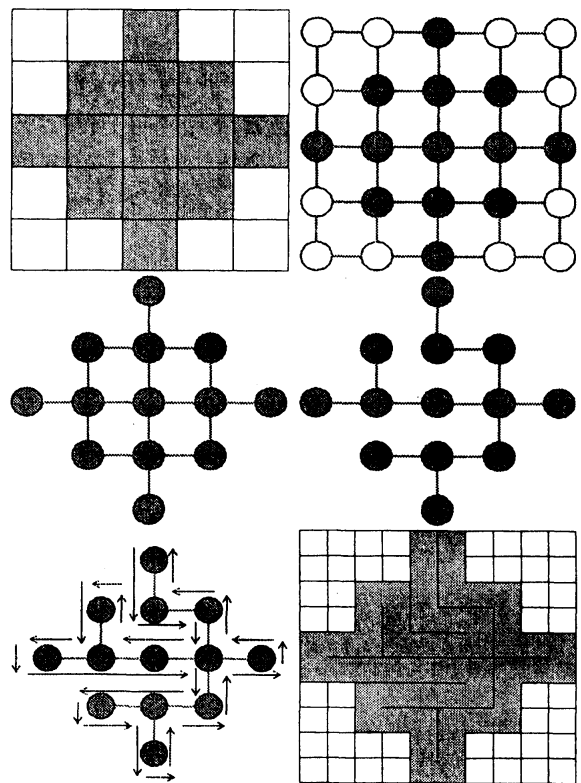


図 3: 絵画的迷路作成のアルゴリズム

次に、地の部分を構成して迷路を完成させる。ハミルトン経路で使われなかった部分の辺を取り除いたグラフを考え (図 4 右上)、解経路の辺は取り除かないように全域木を生成する (図 4 左下)。その全域木から迷路を構成すればよい (図 4 右下)。迷路のスタートとゴールはそれぞれハミルトン経路の両端点となるので、必ず隣接することになる。

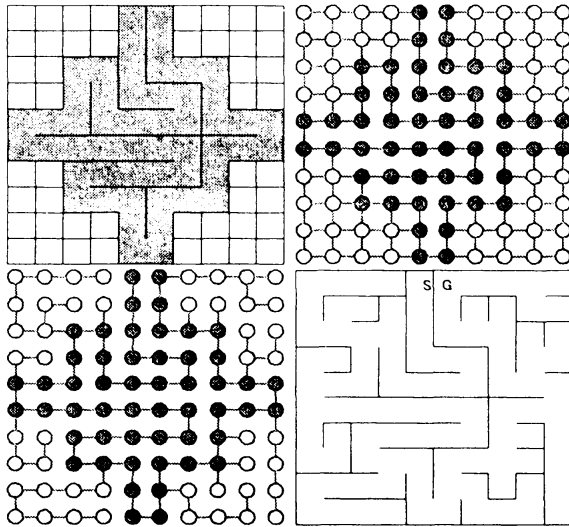


図 4: 絵画的迷路作成のアルゴリズム続き

3 提案アルゴリズム

岡本・上原によるアルゴリズムの欠点として、スタートとゴールが必ず隣り合っていないとはならず、スタートとゴールを自由に指定できないということがあげられる。しかしパズル雑誌に掲載されているような絵画的迷路で、スタートとゴールが隣り合っているものは稀である。そこで、本論文では次のような問題に対するアルゴリズムを提案する。

— 始点と終点を指定した絵画的迷路作成問題 —

入力 縦 m ピクセル、横 n ピクセルの白黒2値画像。始点と終点(ともに端にあることを求める)

出力 解となる経路を塗ることで、入力画像が出てくる迷路。ただし解経路は指定された始点から始まり、終点で終わる。

岡本・上原のアルゴリズムの主となるアイデアは、各マス目を縦横2分割することであった。しかしこの方法では、スタートとゴールを自由に指定することができない。また各マス目を縦横3等分してもうまくいかない例が存在する(5節参照)。そこで提案アルゴリズムでは、各マス目を縦横4等分すること

によりこの問題を解決する(図5)。ただし、入力と出力とでマス目の縮尺が変わってしまうので、終点と始点はそれぞれ対応する場所に置かれればよいことにする。提案アルゴリズムは次のようなものである。

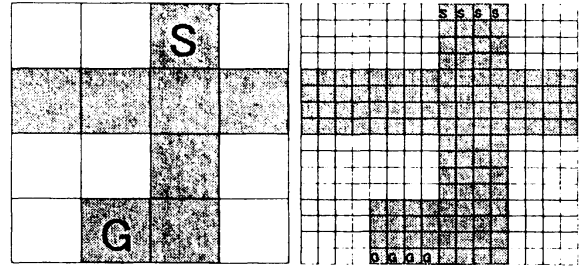


図 5: 提案アルゴリズムのアイデア

縦 m ピクセル、横 n ピクセルの白黒2値画像、始点、終点が与えられたとき(図6左上)、絵に対応するマスの頂点全体が誘導する部分グラフに着目し(図6右上)、その全域木をランダムに生成する(図の例ではそのまま)。得られた全域木で、始点から終点までの経路を見つけ(図6左下)、縦横2等分したマス目に対して経路を作る。(図6右下)

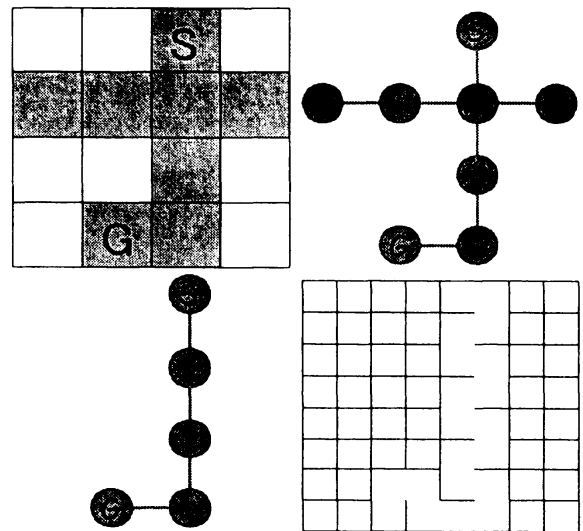


図 6: 提案アルゴリズム

縦横2等分したマス目に対して経路を構成すると先に述べたが、その構成方法について詳しく紹介す

る。図7に4つのパターンを示した。次に進む方向によって場合分けしたものである。経路は縦横2等分したマス目の左上(○のマス)からはじまるものとする。回転・反転を考慮すれば、この4つのパターンですべての可能性を網羅している。したがってこれらを組み合わせることによって、ジグザグな経路を構成することができる¹。

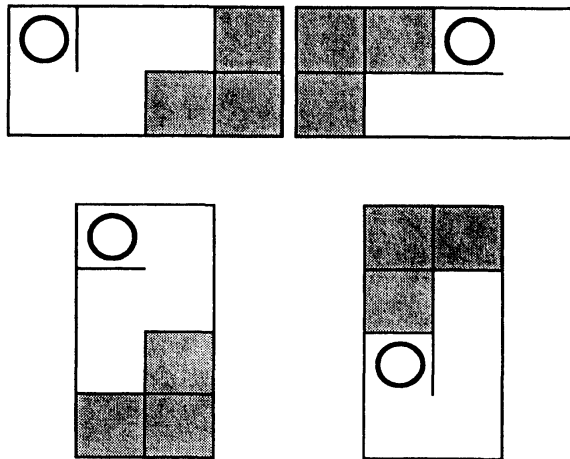


図7: ジグザグな経路

アルゴリズムの本筋に戻る。今得られた経路に残された部分を付け加えることで、絵の部分の経路を完成させたい。付け加えたい箇所が図8左上の右側の網掛け部分のようになっている場合、岡本・上原のアルゴリズムを適用することで、付け加えることが可能になる(図8右上)。始点と終点が必ず隣り合ってしまうという特徴がここでは有効活用されている。

しかし左側の網掛け部分に対して同様の手続きを行うと、ループが出来上がってしまう。そこでマス目をさらに縦横2等分することによりこの問題を解決する。これまでに構成された経路はジグザグにすることにより拡張できる(図8左下)。すると残った部分もさきほどと同様の手続きにより、付け加えることができる(図8右下)。

¹ペアノ曲線の構成法と類似している

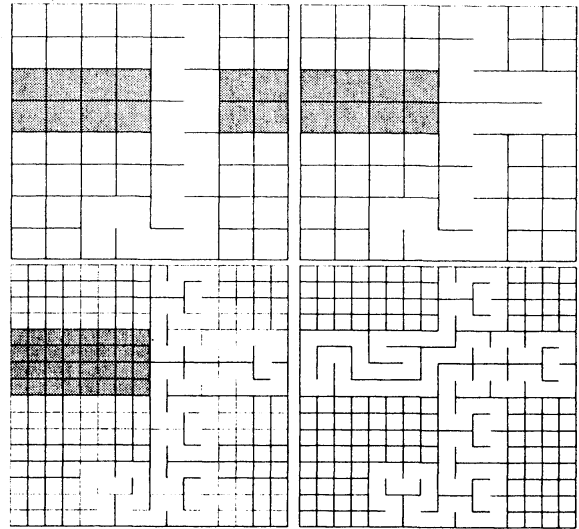


図8: 提案アルゴリズム続き

最後に地の部分を構成すれば、迷路が完成する(図9右)。

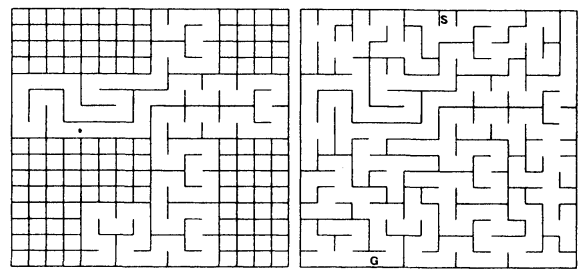


図9: 提案アルゴリズム完成

4 まとめ

本論文では始点と終点が指定された際の絵画的迷路作成アルゴリズムを提案した。アルゴリズムの長所としては、複雑な計算をしていないので処理時間が速いということがあげられる。短所としては解となる経路の単調さ、地の部分の質の悪さなどがあげられる。後者に関しては、岡本・上原によってヒューリスティックな改善方法が提案されている。

5 補足

提案アルゴリズムでは、各マス目を縦横4等分しているが、3等分では迷路を構成できない例があることを示す。

図5左の各マス目を縦横3等分すると、図10中央のようになる。これを市松模様（チェッカーボード模様）に白黒に塗った図が図10右である。ハミルトン経路が存在するのであれば、白マスと黒マスを交互に通過しなければならない。しかし図10右において、黒マスは24マス、白マスは21マスなので、ハミルトン経路が存在しないことがわかる。

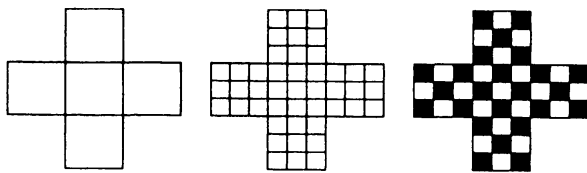


図 10: ハミルトン経路が存在しない例

参考文献

- [1] 岡本吉央、上原隆平、絵画的迷路の作り方、2008年度冬のLAシンポジウム、pp.10-1-10-8