オンライン木探索の最適性について

八神貴裕 * 山内由紀子 † 来嶋秀治 † 山下雅史 †
* 九州大学工学部電気情報工学科 † 九州大学大学院システム情報科学研究院

1 はじめに

多角形内の侵入者を探索者によって見つける問題に対して多くの研究がされてきた．多角形内のすべての点が見えるようにカメラを配置する美術館問題，動き回る侵入者を視界の限られた動かない探索者でみつけるサーチライトスケジューリング問題や，移動できる探索者で侵入者をみつける多角形探索問題などがある [1]．また、このような探索問題は多角形内だけでなく，グラフ内を逃げ回る侵入者を探索者によって捕獲するグラフ探索問題も研究されてきた [3]．これらの探索問題は身の回りの例では，建物内の監視を人間ではなくロボットに任せたいときや，動物園から逃げ出した動物を探すときに役に立つかもしれない．

本稿では複数の探索者による多角形探索のオンラインアルゴリズムを考えるための準備として，まず多角形を単純化し木とみなした．複数の探索者によるオフラインの多角形探索はすでに結果が知られていて，探索者数の上界を得る手段の一つとしてグラフ探索問題へ帰着が行われていた [2]．本来であれば，木は頂点と辺の集合であるが，ここでは辺や頂点を平面空間のようにとらえて考えた．本稿では一人の探索者と，侵入者の移動を制限するためのバリケードをいくつか用いて木内の侵入者を見つけ出すオンラインアルゴリズムの提案とその正当性を示す．

2 章では問題の定義，3 章ではアルゴリズムの紹介，4 章ではアルゴリズムの解析を行う．

2 諸定義

探索する対象となる木を $T = (V, E)$ とする．T は頂点集合 V と辺集合 E からなる連結で閉路を持たないグラフである．また，T の任意の頂点 $v \in V$ を根とした根付き木を $T(v)$ と表す．T の頂点，辺の初期状態を非クリアと呼ぶ．探索者がある頂点や辺にいるとき，その頂点や辺の状態をクリアと呼ぶ．探索者は侵入者の侵入を防ぐためのバリケードを持ち，今いる頂点にバリケードを設置することができる．非クリアな頂点や辺と接していて，探索者やバリケードが置かれていない辺や頂点は再び非クリアとなる．クリアであった頂点・辺が再び非クリアになることを再汚染とよぶ．

ある時刻において，すべての頂点・辺の状態がクリアであるとき探索成功とする．

探索者は以下の 5 個の基本動作を組み合わせて探索する．

- 辺を通過して隣接する頂点に移動する．
- 向きを変える．(頂点内でのみ)
- 今いる頂点にバリケードを設置する．
- 今いる頂点のバリケードを回収する．
- 今いる頂点のバリケードのメモリを操作する．

侵入者も探索者同様に辺を通過して隣接する頂点に移動できる．しかし，侵入者は探索者やバリケードがいる頂点・辺には侵入できない．

また，本稿では記号をそれぞれ次のように定義する．r は探索開始時に探索者の持つバリケードの個

数，b_i は i 番目に設置したバリケード，$v(b_i)$ は b_i が設置されている頂点とする．

本稿では根付き木 $T(v)$ のある任意の頂点 $u \in V$ の子である各頂点を根とした部分木を頂点 u に隣接する部分木と呼ぶことにする．

探索開始時，探索者はすべてのバリケード (r 個) を持っていて，与えられた任意の頂点 $v \in V$ から探索を開始する．

3　提案アルゴリズム

まずアルゴリズムのアイデアを紹介する．探索者は木 T の次数 3 以上の頂点にバリケードを設置する．T は 3 つ以上の部分木に分解さるので，探索者はそれぞれの部分木に対して同様にバリケードを設置し探索する．このような考え方はすでに [3] で用いられている．この考え方をもとにしたオンラインアルゴリズムを示す．このアルゴリズムの重要な部分はバリケードを設置するのに適した頂点を探索者自身で探すことである．

このアルゴリズムでは探索者と各バリケードにいくつかメモリを与えている．それらのメモリを紹介する．

探索者のメモリ

- $mode$ 探索者の動作や出力を決定する．アルゴリズム中では A, B, C の値をとるように記述しているが，それぞれ異なる整数を割り当ててもよいので整数型とする．3 つの値が区別できればよいのでサイズは 2bit.
- $flag$ それぞれのバリケードが最適な位置にいるか記録する配列．配列の大きさは r．バリケード b_i の $flag$ を $fag[b_i]$ と表す．-1, 0, 1 の 3 つの値を区別できればよいので整数型で各サイズは 2bit．全体で $2r$bit.
- $barr\_num$ 探索者が今持っているバリケードの個数．0 から r までの自然数が区別できればよいので整数型でサイズは $\lceil \log 2(r+1) \rceil$
- $deg1$ 次数 1 の頂点を訪れた回数を記録する．グラフの最高次数を調べるときのみ使われる．0 から 2 までの整数が記録できればよいので整数型でサイズは 2bit.

バリケードのメモリ

- $counter$ 未探索の部分木を順に正しく探索するための情報を記録する．T 内の頂点の最大次数を d とすると，0 から d までの整数が書き込めればよい．整数型でサイズは $\lceil \log_2(d+1) \rceil$bit.
- $name$ 設置されているバリケードを区別するためのメモリ．バリケードに 1 から r まで順番に整数で名前を付けるとしたら，整数型でサイズは $\lceil \log_2 r \rceil$bit.
- $failure$ 探索に失敗しバリケードに戻ってきたときの $coumter$ の値を書き込む．よって、型やサイズは $counter$ と同じ．

メモリを操作できるのは探索者だけである．つまり，探索中で探索者がいない頂点に設置されているのバリケードのメモリの値が変化することはない。

探索者の基本的な移動の仕方について説明する．このアルゴリズムでは頂点を部屋，辺を通路のようにとらえている．探索者には向きがあり，移動するときは進行方向を向いている．探索者が向きを変えるのは進行方向を変更するときのみである．探索者がある頂点 u から隣接する頂点 v に移動して，まだ向きを変えていないとき，探索者が 180 度向きを変えると辺 (u, v) がある向きを向く．また，この辺 (u, v) を入口となった辺と呼ぶ．次数 2 以上の頂点から隣接する頂点へ移動するとき，入口となった辺を向いている状態から，その場で反時計回りに向きを変え 1 つ目に見つけた辺に向かって進む．

メモリ $counter$ と探索者の動作について説明する．探索者はバリケード b_i を設置するとき，設置するバリケードの $counter$ に今いる頂点の次数を書き込む (ただし，アルゴリズム中で最初にバリケードを置いたときのみ次数より 1 だけ大きい値を書き込む)．

探索者が隣接する部分木の探索のために $v(b_i)$ から移動する直前に $counter$ の値を 1 減らす．このとき探索者は，上で説明した移動方法で移動することで，隣接する部分木を順に探索できる．$v(b_i)$ に戻ってきたとき，$counter = 1$ であれば探索すべき部分木をすべて探索したことになる．

また，メモリ $name$ は探索者が本来操作すべきではないバリケードを操作しようとするのを防ぐためのものである．

オンラインアルゴリズムにおける真部分木の探索とバリケードの個数に関わる関数を定義する．$T(v)$ から v 以外の任意の次数 3 以上の頂点 $u \in V$ を選ぶ．u を根とした部分木を S とする．S 内の侵入者に S 以外の領域への経路を与えず，S の探索がバリケード k で十分であるとき，$m_{flag,S}(k) = \text{true}$ と表す．S 以外の領域の再汚染が一時的に生じるが，部分木 S と S から逃げ出した侵入者がいる可能性のある領域の探索が k 個で十分であるとき，$p_S(k) = \text{true}$ と表す．

木探索のオンラインアルゴリズムを GDB(Graph Decomposition by Barricade) と呼ぶことにし，main 関数を Algorithm 1 に示す．

main ではメモリの初期化，探索の中心となる関数 search(k) の呼び出し，結果の出力を行う．ここで用いる k はメモリの $barr\_num$ を表す変数である．4 行目，最大次数を調べる（次数 3 以上の頂点に移動）について補足をする．木 T の最大次数が 2 以下の場合，探索者はバリケードを持っていなくても探索を成功することができる．このときメモリ $deg1$ を使う．T の最大次数の調べ方は Algorithm 5 に示す．

ある部分木 S に対する search(k) の動作は $flag$ の値によって変化し，$flag = -1$ または 0 のときの関数 S_flagleq0 を，$flag = 1$ のとき関数 S_flag1 を呼び出す．また，S_flagleq0，S_flag1 は S 内の部分木に対して search($k-1$) を呼び出していて，再帰的探索を行うアルゴリズムである．search(k) を呼び出したとき，search(k) が $mode = B$ で終了すれば $p_S(k) = \text{true}$，$mode = C$ で終了すれば $p_S(k) = \text{false}$ であることを意味する．search(r) の終了時の $mode$ の値によって木全体探索の結果がわかる．search(k) を Algorithm 2，S_flagleq0(k) を Algorithm 3，S_flag1(k) を Algorithm 4 に示す．

Algorithm 1 GDB の main 関数

```
1:  各バリケードの flag ← 0
2:  deg1 ← 0
3:  mode ← A
4:  T の最大次数を調べる（次数 3 以上の頂点に移動）
    最大次数が 2 以下なら mode = B になっている
5:  if mode = A then
6:     バリケード b_1 を設置する
7:     //反時計回りに v(b_1) につながっている部分
       木を S_1, ..., S_n とする。
8:     for 1 ≤ i ≤ n do
9:        S_i に対して search(r − 1)
10:       if P_{S_i}(r − 1) = false then
11:          flag[b_1] ← −1
12:          b_1 を回収して S_i へ進む
13:          search(r)
14:          break
15:       else if P_{S_i}(r-1) = true かつ i = n then
16:          mode ← B
17:       end if
18:       i++
19:    end for
20: end if
21: //結果の出力
22: if mode = B then
23:    探索成功
24: else if mode = C then
25:    探索失敗
26: end if
```

Algorithm 2 search(k)

1: //入力としてバリケードの個数を受け取る
2: $mode \leftarrow A$
3: $failure \leftarrow 0$
4: **while** $mode = A$ **do**
5: 次数 3 以上の頂点に移動する
6: バリケード b_{r-k+1} を設置する
7: **if** $flag[b_{r-k+1}] = -1$ または 0 **then**
8: S_flagleq0(k)
9: **else if** $flag[b_{r-k+1}] = 1$ **then**
10: S_flag1(k)
11: **end if**
12: **end while**
13: $flag[b_{r-k+1}] \leftarrow 0$
14: b_{r-k+1} を回収する

Algorithm 3 S_flagleq0(k)

1: //入口となった辺から反時計回りに $v(b_{r-k+1})$ につながっている部分木を $S_1, ..., S_n$ とする。（入口は含まない）
2: **for** $1 \leq i \leq n$ **do**
3: S_i に search($k-1$)
4: **if** $P_{S_i}(k-1) =$ false **then**
5: $flag[b_{r-k+1}] \leftarrow -1$
6: b_{r-k+1} を回収し、S_i の方へ進む
7: break
8: **else if** $P_{S_i}(k-1) =$ true かつ $i = n$ **then**
9: **if** $flag = -1$ **then**
10: $flag[b_{r-k+1}] \leftarrow 1$
11: b_{r-k+1}を回収し 非クリアな領域の方へ進む
12: **else if** flag=0 **then**
13: $mode \leftarrow B$
14: **end if**
15: **end if**
16: $i++$
17: **end for**

Algorithm 4 S_flag1

1: //入口となった辺から反時計回りに $v(b_{r-k+1})$ につながっている部分木を $S_1, ..., S_n$ とする。（入口は含まない）
2: **for** $1 \leq i \leq n$ **do**
3: S_i に search($k-1$)
4: **if** $P_{S_i}(k-1) =$ false **then**
5: **if** $failure = 0$ **then**
6: $failure \leftarrow i$
7: **else if** $failure \neq 0$ **then**
8: $mode \leftarrow C$
9: break
10: **end if**
11: **end if**
12: **if** $i = n$ **then**
13: **if** $failure = 0$ **then**
14: $mode \leftarrow B$
15: **else**
16: b_{r-k+1} を回収し、$S_{failure}$ の方へ進む (反時計回りに $failure + 1$ 個目の辺)
17: **end if**
18: **end if**
19: $i++$
20: **end for**

Algorithm 5 最大次数を調べる

1: //次数 3 の頂点が見つかる前に異なる 2 つの次数 1 の頂点を見つければ T の最高次数は 2 以下である
2: **while** 今いる頂点の次数が 1 または 2，かつ $deg1 < 2$ **do**
3: $deg1 \leftarrow deg1 + 2 -$ (今いる頂点の次数)
4: すでに説明した移動方法で移動する
5: **end while**
6: **if** 今いる頂点の次数が 0 または $deg1 \geq 2$ **then**
7: $mode \leftarrow B$
8: **end if**

4 アルゴリズムの解析

根付き木 $T(v)$ の v 以外の任意の次数 3 以上の頂点を選び u とする．u を根とする部分木を S とする．

S 内の u に隣接する部分木を $S_1, ..., S_l$ と表すと自然数 i を用いて，以下の 2 つの補題が成立する．

補題 1. S が部分木 $S_1, ..., S_l$ のうち $p_{S_i}(k-1) =$ false となる異なる S_i が 2 つ以上，または $m_{1,S_i}(k) =$ false となる S_i が 1 つ以上で構成されているならば $m_{1,S}(k) =$ false である．

補題 2. S が部分木 $S_1, ..., S_l$ のうち $p_{S_i}(k-1) =$ false となる異なる S_i が 3 つ以上で構成されているならば $p_S(k) =$ false である．

オンラインアルゴリズム GDB の正当性について定理 1 が成り立つ．

定理 1. GDB の出力は探索開始時の探索者の位置 $v \in V$ によって変化しない．

Proof. GDB で成功または失敗がわかったとき，探索中最後に b_1 を設置していた頂点を v とし，$T(v)$ を考える．v に隣接する部分木を $S_1, ..., S_l$ と表す．このアルゴリズムで探索が成功するときの木の構造は自然数 i, j を用いて，$p_{S_i}(r-1) =$ false かつ $m_{1,S_i}(r) =$ true となるような S_i は高々 1 つで，$\forall j(j \neq i)$, $p_{S_j}(r-1) =$ true．このアルゴリズムで探索が失敗するときの木の構造には自然数 i を用いて，$p_{S_i}(r-1) =$ false となる異なる S_i が少なくとも 3 個存在する．

失敗したときの木の構造から任意に頂点を 1 つ選び u とし，$T(u)$ を考える．変形後の木には v を根とした部分木 S' が存在し，その S' 内には $p(k-1) =$ false の部分木が少なくとも 2 つ存在する．補題 1 より，S' は $m_{i,S'}(k) =$ false．よって，失敗したときの木はどの頂点を根としても探索ができない． □

オンラインアルゴリズム GDB と他の任意のオンライン，オフラインのアルゴリズムとの関係について定理 2 が成り立つ．

定理 2. GDB で n 個 のバリケードで T の探索に失敗するならば，どのようなオンライン，オフラインのアルゴリズムでも n 個 のバリケードで T の探索に失敗する．

Proof. 数学的帰納法で証明する．

$r = 0$ のとき，GDB を用いてバリケードなしで探索に失敗した木 T を任意に 1 つ選ぶ．T には次数 3 以上の頂点が含まれているので，どのようなアルゴリズムでもバリケードなしでは T の探索に成功しない．また T の任意の次数 1 の頂点を 1 つ選び v とする．v 以外の頂点からなる T の部分木を S とすると，$p_S(0) =$ false となる．どのようなアルゴリズムでもこの部分木はバリケードなしのとき探索に失敗する．

$r = k-1$ で命題成立を仮定し，$r = k$ のときを考える．GDB で $r = k$ で失敗した木 T を任意に選ぶ．T ある特定の頂点 u を根とする $T(u)$ を考え，u に隣接する部分木を $S_1, ..., S_l$ と表す．このとき，$p_{S_i}(k-1) =$ false のとなる S_i が少なくとも 3 つ出現するような u が少なくとも 1 つ存在する．

$p_{S_i}(k-1) =$ false となる S_i 3 つを，一般性を失わず探索が終了する順に A, B, C とする．探索終了とは，ある領域のすべての頂点，辺がクリアになり，それ以降再汚染されないことをいう．探索過程において，ある領域 R の探索が終了する時刻を $f(R)$ と表す．また，R の探索にバリケードが m 個 必要かつ十分であるとき，R 内にバリケードを m 個 設置している時間が存在する．その時間の中で最後の時刻を $e(R)$ と表す．

T をバリケード k 個 で探索に成功するアルゴリズムが存在すると仮定する．このとき，
(1) $e(X) < e(Y) < f(A)$ (X, Y は A, B, C のいずれかで $X \neq Y$)
(2) $e(A) < f(A) < e(B)$
の 2 つの場合を考えれば十分である．

(1) について，時刻 $e(Y)$ で k 個のバリケードはすべて Y 内 に設置されている．このとき，X, Y 以外の領域から X 内への侵入者の経路ができるので，X 内はすべて再汚染される．$e(X)$ 以降で X 内にバリケードを k 個設置することはないので，探索は成功しない．

(2) について，時刻 $e(B)$ で k 個のバリケードはすべて B 内に設置されている．このとき C から A への

侵入者の経路ができ，A 内はすべて再汚染され，仮定に矛盾する．□

上の定理 2 の証明は，[3] ですでに証明されている以下の定理を参考にした．

定理 3. $k \geq 1$ において，木 T の探索が探索者 $k+1$ 人以上必要であるとき，かつそのときに限り，T には頂点 $v \in V$ の枝のうち探索者が k 人必要な枝が少なくとも 3 つ存在するような v が存在する．

ここで，頂点 v の枝 T' とは，v を次数 1 の頂点とみなしたときの極大の T の部分木 T' と定義されている．つまり，$T(v)$ において v の子のひとつを u とするとき，u を根とする部分木を $S' = (V', E')$ と表す．このとき v の枝 T' は $T' = (V' \cup u, E' \cup (v, u))$ と表すことができる．本稿のバリケードを探索者とみなすことで，定理 3 の k はこの問題におけるバリケードの個数（r 個）と探索者の人数（ 1 人）の和 $r+1$ とみなすことができる．

5 まとめ

オンライン木探索において，バリケードの個数が他のどのようなアルゴリズムと同じ，もしくはそれ以下のアルゴリズムを得ることができた．今後は探索者やバリケードのメモリをより少なくすることや，多角形探索への応用を課題とする．

6 参考文献

参考文献

[1] 山下雅史 ,「捜索—移動する対象を探索する」, 室田一雄編,『離散構造とアルゴリズム III』, 近代科学社, pp. 115-162, 1994.

[2] M. Yamashita, H. Umemoto, I. Suzuki and T. Kameda, “Searching for Mobile Intruders in a Polygonal Reagion by Group of Mobile Searchers”, Algorithmica, pp. 208-236, 2001.

[3] T. D. Parsons, “Pursuit-evasion in a graph”, in *TheoryandApplicationsofGraphs*, Y. Alavi and D. Lick, editors, Lecture Notes in Mathematics 642, Springer-Verlag, Berlin, pp. 426-441, 1976.