2005－AL－103（1）
2005／11／11

接触インベーダーゲームに対するオフラインアルゴリズム

藤村幸代，伊藤 暁

山口大学工学部

あらまし　接触インベーダーゲームとは，各インベーダーが侵入してくるのみで玉を打ってこない，また玉を打つのでなく迎撃機をインベーダに接触させることで，自陣への侵入を防ぐようなインベーダーゲームである．本稿ではオフラインアルゴリズム，すなわちゲーム開始から終了までの間に飛来する各インベーダーに関する情報が前もって与えられる状況のもとで，インベーダーの侵入数を最小化（迎撃可能インベーダー数を最大化）するような戦略を求める問題を考察し，そのような最適戦略が $O(n \log n)$ の時間計算量で求まることを示す．ここに，n はインベーダー総数である．

An Off-line Algorithm for Contact Invader Game

Sachiyo Fujimura, Akira Ito

Faculty of Engineering, Yamaguchi University

Abstract　Contact invader game is an invader game in which invaders could move into defence area but cannot shoot bullets and one interceptor can avert the incoming invaders only with its direct collisions to them. In this paper, we consider an offline algorithm for the intrusion minimization problem (interception maximization problem) of this game and show that such optimal strategy can be obtained in $O(n \log n)$ time, where n is the total number of invaders.

1.　まえがき

画面の上端から現れ画面下端の自陣に侵入しようとするインベーダーに対し，自陣の直前で待ち構える迎撃機を左右に動かしてインベーダーを迎撃し、その侵入数をできるだけ少なくするようなゲームに対し，本稿では各インベーダーが侵入してくるのみで玉を打ってこない，また玉を打つのでなく迎撃機をインベーダに接触させることで，自陣への侵入を防ぐような制限されたインベーダゲームを扱う．また本稿では，このような接触インベーダゲームに対するオフラインアルゴリズム，すなわちゲーム開始から終了までの間に飛来する各インベーダーに関する情報が前もって与えられる環境のもとでの，侵入インベーダー数を最小化する（迎撃数を最大化する）問題について考察する．

ところで，オフラインアルゴリズムでは入力データが一度に与えらるが，オンライン環境では入力が逐次的に 1 つずつアルゴリズムに到着する．従って，オンラインでは未来の入力データを用いることなくを過去のデータのみに基づいて，各場面での戦略を設計する必要がある．例えば，OS におけるページング問題を一般化したオンライン問題として k サーバ問題がある [1]．そこではオンラインで次々と到着するリクエスト要求に対し，k 台のサーバーの最終的な総移動距離をできるだけ最小化するようサーバの割り当て戦略を決めることが求められる．オンラ

インアルゴリズムの性能評価としては，事前に入力情報が知らされているオフラインアルゴリズムのうち最良のものとの性能比（競合比と呼ばれる）が評価尺度として採用されている．k サーバ問題においてはこれまでに知られている最良性能のオンラインアルゴリズムはその内部でオフラインアルゴリズム自体が用いられており [2]，更にそのオフライン版の問題そのものの計算量についても詳しく調べられている [3].

以上の観点から，本稿で扱う迎撃機最大化問題は攻撃機をリクエスト，迎撃機をサーバと考えることで，サービスが受けられるリクエスト数を最大化するような 1 サーバ問題（あるいは救援物資配給問題）のオフライン版と捉えることができる．

まず第 2 節では，議論に必要となる諸定義を与える．そこでは，迎撃数最大化問題がインベーダー間のある順序関係をグラフで表したときに，そのグラフの最長パスを求める問題と等価であることを示す．また，この事実に基づいて素朴なアルゴリズムを設計すると $O(n^2)$ の計算量が必要となることを示す．これに対し第 3 節では，順序関係を表すグラフを前処理しておくことでその枝数を $O(n)$ まで減少させることができ，最終的な計算量を $O(n \log n)$ に落とすことが出来ることを示す．ここに，n はインベーダー総数である．

2. 準備

本稿ではインベーダゲームに関する諸定数を以下のように定義する（図 1 参照）.

- ゲーム開始時刻，終了時刻：$0, T$,
- 防御すべき自陣の幅＝迎撃機（自機）の移動可能範囲＝画面水平幅：$[0, 1]$,
- レーダー視野範囲＝画面垂直幅：$[0, 1]$,

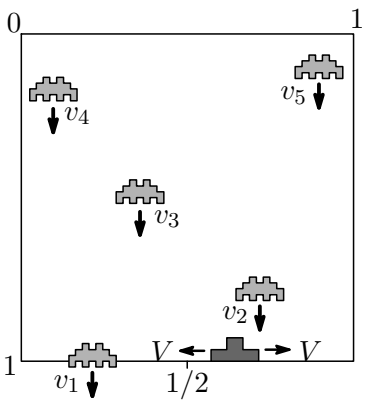


図 1. インベーダゲームの座標設定

- インベーダー総数：n,
- 各インベーダーが画面上端に現れる時刻と水平位置：s_i, x_i,
- 各インベーダーの垂直移動速度：v_i,
- 自機の水平移動速度：V,
- ゲーム開始時点での自機の水平位置：中央 $(0, 1/2)$.

各インベーダーが迎撃可能線（画面下端）に達する時刻は $t_i = s_i + 1/v_i$ と計算できることに注意．オフラインでは，各インベーダーが迎撃可能水平線に達する時刻 t_i とその位置 x_i のみが問題になる（オンラインでは，s_i, v_i も問題に含まれる）.

以下が本稿で扱う最適化問題である.

定義 2.1 (迎撃数最大化問題)

入力： 各インベーダー $i (1 \le i \le n)$ の到達時刻と到達位置 (t_i, x_i).

出力： 迎撃インベーダー数が最大（侵入インベーダーが最小）となるような，迎撃対象インベーダー系列.

上述の設定ではインベーダーは全て同一としている．もしインベーダーの種類により防御し

た際の得点（通常は1点）が異なるように設定したい場合には，移動速度が同じ複数のインベーダーを同一時刻・同一地点に与えることで対応できる．

次に，あるインベーダーを迎撃したときの状況を考える．もしその場所が画面の端であったならば，同じ時刻にもう一方の端から侵入しようとするインベーダーは迎撃できない．このように，あるインベーダーをある場所で迎撃すると，その後（あるいは同時に）迎撃可能なインベーダー数は限られてくる．

インベーダー j がインベーダー i を迎撃後に（あるいは同時に）迎撃可能であるとき，$i \preceq j$ と記す．以後，迎撃機 i_0 ならびにインベーダー $i_1, i_2, \ldots, i_n$ の集合を I と記す．

事実 2.1 関係 $\preceq$ は集合 I 上の半順序関係である．

（証明）(1) 反射律 $i \preceq i$, (2) 反対称律 $i \preceq j, j \preceq i \Rightarrow i = j$, (2) 推移律 $i \preceq j, j \preceq k \Rightarrow i \preceq k$ の各性質が成り立つことは明らか． □

上記の事実より，次が得られる．

命題 2.1 有向グラフ $G = (I, E)$, $E \stackrel{\triangle}{=} \{(i, j) \mid i \preceq j\}$ は非巡回グラフである（G を**連続迎撃可能グラフ**と呼ぶ）．

以後に示すアルゴリズムは次の幾何学的な事実に基づく．

事実 2.2 各 $i, j \in I$ について，

$$i \preceq j \iff (x_j, t_j) \in C_i.$$

ここに，C_i は時空間図2の領域 $[0,1] \times [0,T]$ 内において，点 (x_i, t_j) から下方に出る傾き $V, -V$ の2直線で囲まれる円錐領域である．

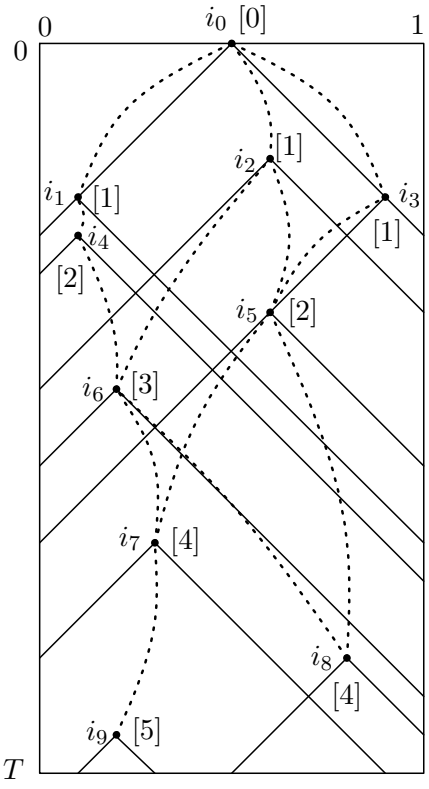


図 2. 接触インベーダゲームの時空間図例

このように，一つの比較 $i \preceq j$ 結果（真偽値）を求めるのに必要な時間は $O(1)$ で済み，すべてのペア $i, j (0 \leq i, j \leq n)$ についての総計時間は $O(n^2)$ となる．生成された連続迎撃可能グラフ G に対してトポロジカルソートを行えば，始点 i_0 からの最長路が求められる．その長さが最大可能得点である．図2において各インベーダーの横に書かれた鉤カッコ内の数値は始点からの最長路を通過してそのインベーダを迎撃したときに得られる点数を表す．

よく知られれているように，このソートは G の枝数 $|E|$ に比例する時間で済む．

以上の議論により，素朴なアルゴリズムの性能について以下が成り立つことがわかる．

定理 2.1 迎撃数最大化問題に対する素朴なアルゴリズムの時間計算量は $O(n^2)$ である．

素朴なアルゴリズムにおいて計算量上一番のネックとなっているのは，G を生成する際に行う比較回数が $O(n^2)$ である点である．

ところで，連続迎撃可能グラフ G において自己ループならびに推移律を表す枝を省略するとハッセ図 $G_{\rm H} = (I, E_{\rm H})$, $E_{\rm H} \stackrel{\triangle}{=} \{(i,k) \in E \mid \forall j \in I[(i,j) \notin E$ または $(j,k) \notin E]\}$ が得られるが，$G_{\rm H}$ には G の全ての情報が含まれている（図 2 の例ではハッセ図の枝が点線で示されている）．

そこで次節では，（G の生成過程を経ないで）$G_{\rm H}$ を入力データから直接生成し，それに対してトポロジカルソートを行うようなアルゴリズムを提案する．その際，$G_{\rm H}$ の枝数が $O(n)$ で抑えられることが示される．

3. 提案アルゴリズム

本節では，素朴なアルゴリズムよりも効率的なアルゴリズムの存在について考察する。見通しを良くするために，まず時空間図 2 の座標系 (t,x) を，迎撃機の初期位置 i_0 を起点とする傾き $V, -V$ の 2 直線に沿った単位ベクトルを基底ベクトルとする座標系 (X,Y) へ変換する．この座標変換 f は線形写像（アフィン写像）であり，I の各点について $O(1)$ 時間で行える．

図 3 において，各インベーダー i から出ている 2 本の点線の矢印は，その点よりも原点に近い領域にあるような点のうち，Y 座標，X 座標が最も大きい他のインベーダ $MaxY(i)$, $MaxX(i)$ をそれぞれ指している．例えば，i_6 は i_8 と i_7 から指されているが，i_9 からは指されていない．i_9 は i_7 を迎撃後に迎撃可能だからである．このように，各点 i について $MaxY(i), MaxX(i)$ を求めれば，連続迎撃可能性に関するハッセ図 G_H を構成できることがわかる．

提案アルゴリズム

1. 座標変換：各 $i(0 \le i \le n)$ について，$(X_i, Y_i) = f(t_i, x_i)$ を計算する．
2. 区間最大値探索： 各 $i(1 \le i \le n)$ について，$MaxY(i) = \max_{0 \le k \le n}\{Y_k \mid 0 \le X_k \le X_i, 0 \le Y_k \le Y_i, k \ne i\}$ ならびに $MaxX(i) = \max_{0 \le k \le n}\{X_k \mid 0 \le X_k \le X_i, 0 \le Y_k \le Y_i, k \ne i\}$ を求める．
3. ハッセ図 $G_{\rm H} = (I, E_{\rm H})$ の作成： 各 $i(1 \le i \le n)$ について，$MaxY(i) = Y_k$ なる k に対して，$(k,i) \in E_{\rm H}$ とする．同様に，$MaxX(i) = X_k$ なる k に対して, $(k,i) \in E_{\rm H}$ とする．
4. トポロジカルソート： $G_{\rm H}$ において，点 i_0 からの最長路 $\mathcal{P}$ を求め，最適戦略とする．

上記アルゴリズムで $MaxY(i)$ を求める際に，もし $Y_k = Y_{k'}$ なる 2 つ以上の点が $i_k, i_{k'}$ が存在する場合には，$\max X_k$ なる Y_k を $MaxY(i)$ とする．$MaxX(i)$ についても同様に計算する．

系 3.1 $|E_{\rm H}| \le 2|I|$.

（証明）$G_{\rm H}$ において，各点 $i \in I$ の入次数は高々2 である． □

$MaxY(i)$ と $MaxX(i)$ を計算するためには，ヒープ探索木 (priority search tree) を利用する．ヒープ探索木とはヒープ（順位付きキュー）と 2 分探索木を組み合せたデータ構造であり，計算幾何学の分野において区間列挙等のために使

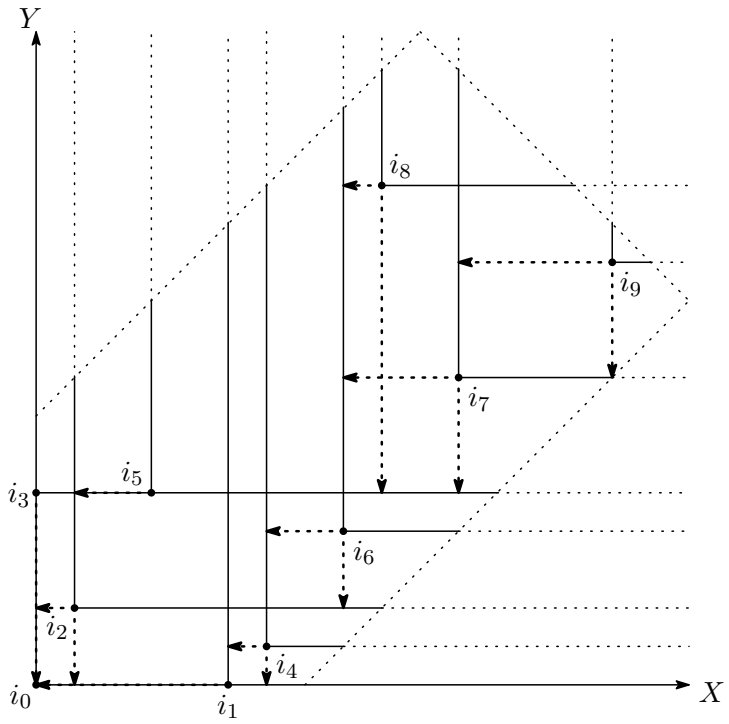


図 3. 座標変換後の時空間図

用される．ヒープ探索木ではヒープ条件

点 v の X 値 $\leq v$ の子孫の X 値

が満たされるとともに，準対称順

点 v の左部分木の子孫の値
$\leq v$ の右部分木の子孫の値

が満たされる（対称順［点 v の左部分木の子孫の Y 値 $\leq$ 点 v の Y 値 $\leq v$ の右部分木の子孫の Y 値］自体は満たさない）．

まず，入力データから Y 軸，X 軸に関するヒープ探索木をそれぞれ構成する．以下は，Y 軸に関するヒープ探索木 $\mathcal{T}_Y$ の再帰的構成手続きである．

ヒープ探索木 $\mathcal{T}_Y$ の構成 [4]

1. Y 座標に関してソートされた点系列 $S = \langle i_0, i_1, \ldots, i_n \rangle$ を引数として，以下の手続きを実行する：

(1) $S = \emptyset$ ならば，何もしないで戻る．

(2) S から X 座標の値が最小（複数あれば Y 座標が最小）の点 i_{xmin} を選び，現在の部分木の根とする．

(3) 点系列 $S' = S - \{i_{\mathrm{xmin}}\}$ を，前半部 S_1 ($|S_1| = \lceil |S'|/2 \rceil$) と後半部 S_2 ($|S_2| = \lfloor |S'|/2 \rfloor$) に分割する．

(4) S_1, S_2 をそれぞれの引数として，左部分木，右部分木の作成を再帰的に行う．

ステップ 1.(b) によりヒープ条件が，ステップ 1.(c) により準対称順が満たされることに注意されたい．$\mathcal{T}_X$ も同様にして構成する．

次に，区間 $[0, X_i] \times [0, Y_i]$ 内にある点 i 以外の点でその Y 座標が最大のものを求める手続き $MaxY(i)$ の計算法をを示す．通常の領域最小値探索では半無限領域が対象とされるが，本アルゴリズムでは有限領域が探索対象であることに注意されたい．

$MaxY(i)$ の計算アルゴリズム

ヒープ探索木 $\mathcal{T}_Y$ の根から開始する．

1. 現在の点 p が空もしくは X 座標の値 X_p が $Xp > X_i$ ならば，$-\infty$ を返す（範囲がオーバーしたか点が存在しない）．

2. 右部分木の区間最大値 Y_{m} を調べる．もし $-\infty$ だったならば，左部分木の区間最大値 Y_{m} を調べる（右部分木を優先する）．

3. $p \neq i$ かつ $Y_{\mathrm{m}} < Y_p \leq Y_i$ が成り立つならば，Y_p を返す．そうでなければ，Y_{m} を返す．

$MaxX(i)$ も同様にして計算する．

図 3 の入力データに対するヒープ探索木 $\mathcal{T}_Y$, $\mathcal{T}_X$ を図 4 に示す．

定理 3.1 迎撃数最大化問題に対する提案アルゴリズムの時間計算量は $O(n \log n)$ である．

(証明) ヒープ探索木 $\mathcal{T}_Y, \mathcal{T}_X$ の構成は $O(n \log n)$ 時間，各 $MaxY(i), MaxX(i)$ の計算は $O(\log n)$ 時間で済む． □

なお，$\Omega(n \log n)$ 時間が必要かどうかは未解決である．

4. むすび

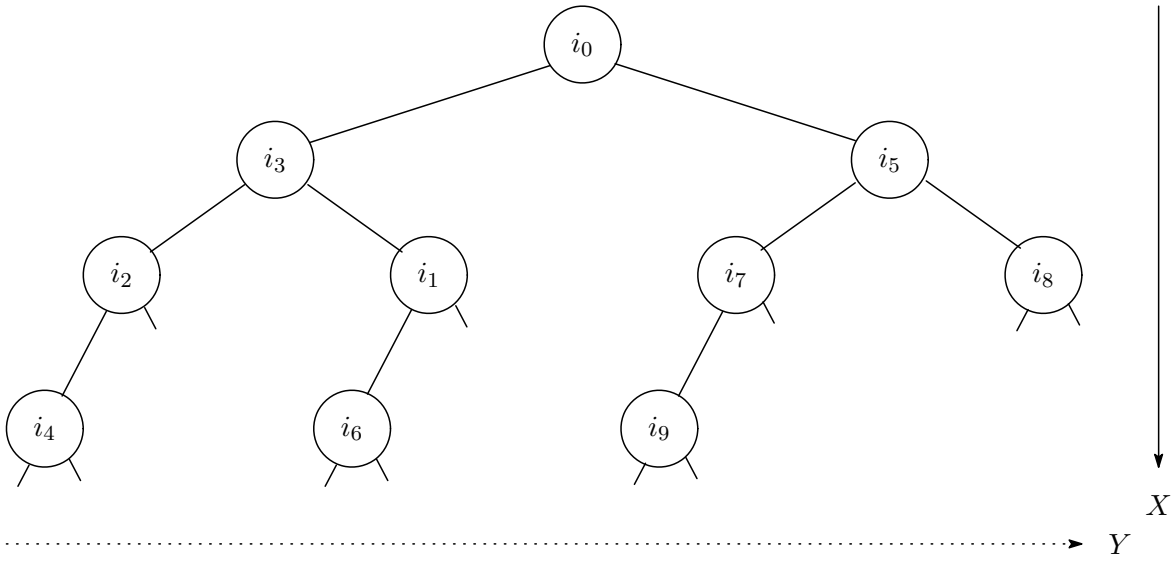
本稿では現実のインベーダーゲームのオフライン計算モデルについて考察した．

今後はオンラインアルゴリズムに関する考察とともに，基本モデルを拡張して更に自然なゲームに近づけることが課題として挙げられる．拡張の第 1 段階としては自機が攻撃機に接触して迎撃するだけではなく遠方から玉を打って迎撃できるような玉打ち迎撃モデル（すなわち非触モデル）が挙げられる．この場合には，時空間図においてインベーダーを点ではなく垂直線分と捉えることができる．インベーダー線分の下端はインベーダー本体の存在点を表し，その長さはインベーダーが画面上に現れている時間すなわち画面の高さに比例することになる．また，玉打ち迎撃モデルの変種として，インベーダーが点ではなくある一定のサイズ（幅と奥行き）を持つ矩形インベーダーモデルが考えられる．この場合は，インベーダーの中心点から一定の水平距離内に近づくだけで迎撃できるようになる．いずれにせよ，基本モデルでは連続迎撃可能性を表す順序関係が比較不能であったインベーダーどうしが，これらの場合には条件付きで比較可能となり，ハッセ図の構成がより困難なものとなる．

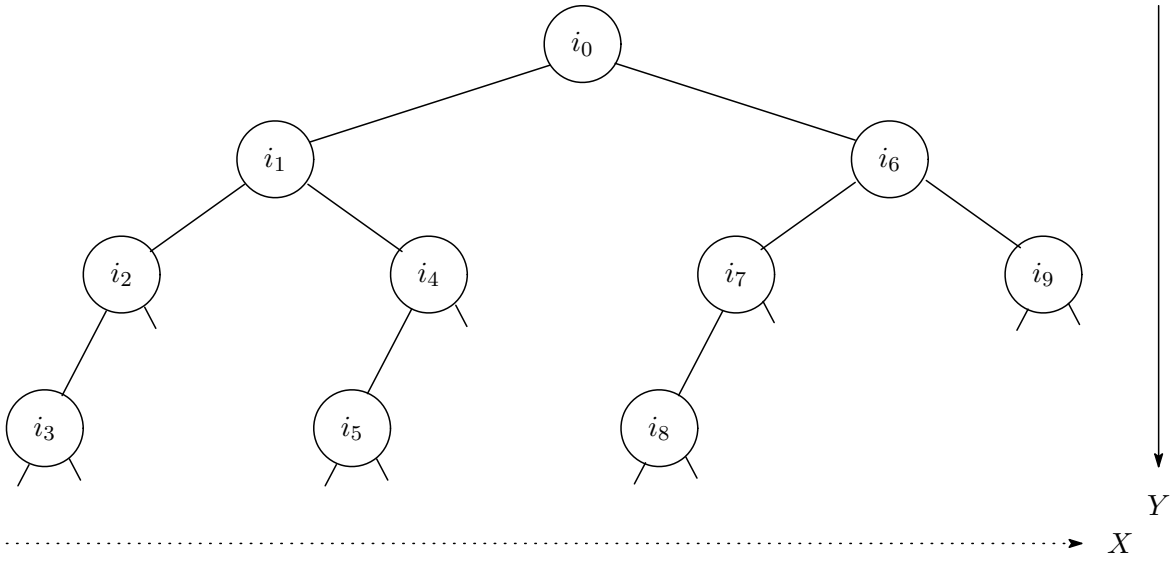
なお，自機だけではなくインベーダーも玉を打つ玉打ち攻撃モデルでは，障害物を回避しながら最長パス探索を行う問題となり，更に複雑なアルゴリズムが要求される．

参考文献

[1] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive Algorithms for On-Line Problems. In Proc. of the



(a) $\mathcal{T}_Y$



(b) $\mathcal{T}_X$

図 4. ヒープ探索木

20th Ann. ACM Symp. on Theory of Computing, pp. 322–333, May 1988.

[2] E. Koutsoupias and C.H. Papadimitriou. On the kserver conjecture. In Proceedings of 26th Annual ACM Symposium on Theory of Computing, pp. 507– 511, 1994.

[3] M. Chrobak, H.J. Karloff, and T Payne. New results on server problems. In Proc. 1st ACM-SIAM Symp. on Discrete Algorithms, pp. 291–300, 1990.

[4] 浅野哲夫, 計算幾何学, 朝倉書店, 1990.