Vol.2019-ARC-240 No.4
Vol.2019-SLDM-191 No.4
Vol.2019-EMB-53 No.4
2019/2/27

スパイキングニューラルネットワークを用いた時間依存ネットワークに対する経路探索手法の検討

穴澤 徳明[1,a)]　上野 洋典[2]　近藤 正章[2]

概要：従来のノイマン型と異なるニューロモーフィックコンピューティングが注目を集めている．ニューロモーフィックコンピューティングは並列性が高く，全体を逐次的に処理する必要が無いため，グラフ問題との相性が良い．本稿では，特にグラフの経路探索問題について考え，ニューロモーフィックコンピューティングの特性を用いたダイナミックな経路探索アルゴリズムを提案する．この手法はアルゴリズムの進行が時間依存なため，グラフの各頂点の通過可能性が時間変化する場合にも適用できる．また，グラフ上のスタートノードからゴールノードまでなるべく早くエージェントを動かす問題を考え，頂点通過可能性の確率過程が与えられていた場合に効率的にゴールまで到達するアルゴリズムを提案する．具体的な実験として，グリッド上に配置された障害物が確率で移動する場合について確率過程を学習したエージェントの性能評価を行う．

1. はじめに

近年，自動運転の実現に向けた取り組みが世界各国で活発に進められている．その構成要素の中でも経路探索アルゴリズムは最も重要なもののひとつである．また，経路探索は輸送計画の最適化や自律移動ロボットの行動決定，地図アプリケーションのルート案内などにも用いられており，応用先は多岐にわたる．また，経路探索アルゴリズムは通常時間的に変化しないグラフに対して用いられるが，自動運転や自律移動ロボットなどの応用では経路上を障害物が移動することも十分に考えられる．上記のようなリアルタイム性が要求される応用先において，状況が変化するたびに経路を決定しなおすアプローチは現実的ではない．状況の変化にも頑健な経路探索手法が必要である．さらに，IoT 化がすすみスマートフォンなどのエッジデバイスの数が飛躍的に増加していくスマート社会においては，エッジデバイスでルート案内などの経路探索を行う需要も増大すると考えられる．そのためエッジデバイスのような計算資源の乏しい環境においても，低消費電力で経路探索を実行できる必要がある．

本稿では，高効率性および時間変化する環境への適応性を両立する経路探索アルゴリズムを実現するために，ニューロモーフィックコンピューティング（脳型計算）を利用する方法を提案する．

ニューロモーフィックコンピューティングは脳の神経細胞を模した計算原理である．ニューロモーフィックコンピューティングでは，従来のノイマン型の中央集約的な処理とは異なり，ニューロンを模したロジック一つ一つがコアとして働き，コア間のコミュニケーションを神経活動のスパイクを模した信号で行う並列分散処理が基本となる．そのため，並列性を向上させることで特定の計算処理を高速に行える可能性がある．また，ニューロン素子 1 つ 1 つで行われる処理およびニューロン間のコミュニケーションは非常にシンプルであり，消費電力を抑えられるという利点もある．IBM などの研究グループは 100 万個のニューロンと 2 億 5600 万個のシナプスからなる ASIC を製造しており，このチップはリアルタイム，非同期で駆動し，消費電力は 63mW である．[1]

ニューロモーフィックコンピューティングは，その成り立ちからニューラルネットワークを基本とする機械学習技術に用いられることが多い．しかしながら，ニューロモーフィックコンピューティングの本質的な利点はニューロンを模した複数のコアと，コア間のシンプルなコミュニケーションに基づいた高い並列計算性であるため，応用先はニューラルネットワークに限られず幅広い．しかしながら，ニューラルネットワーク以外のアプリケーションをニューロモーフィックコンピューティングにマッピングすることは単純ではない．

[1] 東京大学 工学部 計数工学科
[2] 東京大学 大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[a)] anazawa@hal.ipc.i.u-tokyo.ac.jp

Vol.2019-ARC-240 No.4
Vol.2019-SLDM-191 No.4
Vol.2019-EMB-53 No.4
2019/2/27

本稿では，ニューロモーフィックコンピューティングを用いてグラフ処理の一つである最短経路問題を解くアルゴリズムを提案する．スタートとゴールの間に障害物がある場合でも適切に最短経路を決定できることを実験により示す．さらに，障害物の挙動を学習することにより，障害物の位置が動く，すなわちグラフが時間に依存して変化する場合にも提案手法が有効であることを示す．

本稿で “時間的に変化するグラフ” あるいは “時間依存グラフ” と述べた場合には，頂点と辺の数や接続関係等は変わらず，各頂点の到達可能性が時間毎に変化するものを指す．

2. 関連研究

2.1 グラフ上で最短経路を探索するアルゴリズム

与えられたグラフ上の特定の 2 頂点間の最短経路を求めるには，A*アルゴリズム [2] およびその特殊な場合である Dijkstra 法 [3] が広く用いられている．Dijkstra 法はスタート地点からの経路コストの和が小さい点を優先的に展開していくことで，ゴール地点への最短経路を求めるアルゴリズムである．しかし，個々の経路コストの差が大きくない場合は幅優先探索と同様の探索挙動となり，遠方に行くほど探索範囲が拡大するような環境では効率的に最短経路を求めることが出来ない．そこで，この Dijkstra 法に，ある地点 n からゴール地点までの距離コストの推定値を表すヒューリスティック関数 $h(n)$ を導入したものが A*アルゴリズムである．ヒューリスティックな経路コストの推定値を利用し，探索を展開するノードの優先順位をつけることで，探索効率を向上させている．ヒューリスティック関数には，その地点からゴール地点までのユークリッド距離やマンハッタン距離が用いられることが多い．Dijkstra 法は A*アルゴリズムにおいて，ヒューリスティック関数 h が恒等的に 0 である場合に相当する．また，A*の探索効率はヒューリスティック関数 $h(n)$ の設計に左右されるが，どのような $h(n)$ を用いても，いつかは必ず最適解を発見できることが保証されているという，完全性という性質を持っている．

グラフ上で最短経路を探索するその他のアルゴリズムとして，Wavefront アルゴリズムもよく用いられている [4]．Wavefront アルゴリズムは，スタート地点の近傍から幅優先で各ノードにスタート地点からの距離コストを設定していき，ゴール地点の距離コストが設定された時点でコストの高い順にノードをたどることで，最短経路を見つけるアルゴリズムである．スタート地点を中心に距離コストが設定されていく様子を，波面が広がっていくさまに喩えてその名前が付けられている．任意の環境形状に対応できることから，モバイルロボットの経路決定によく用いられている [5], [6]．

2.2 脳型計算を用いてグラフアルゴリズムを実行する方法についての研究

脳型計算はその成り立ちから機械学習および人工知能（AI）分野への応用が盛んに研究されている．IBM による True North[1] や Intel の Loihi[7] などがその例である．Schuman らの研究グループは，ニューロモーフィックなアーキテクチャにグラフを上手くマッピングすることで，グラフの最短経路および近傍探索を行う手法を提案した [8]．Wavefront アルゴリズムにおける距離コストの設定をニューロンのスパイクの伝播に対応させることで，脳型計算の並列性を上手く利用して効率的にアルゴリズムを実行する．同様の手法を用いて，特定の頂点の近傍のグラフを抽出する方法も提案されている．さらに，同研究グループはより大規模なグラフネットワークにおいて，コミュニティを検出する方法も提案している [9]．ソーシャルグラフをニューロモーフィックなアーキテクチャにマッピングし一定の間隔でニューロンを駆動させることで，同じコミュニティ内のニューロン間に類似性のある発火パターンを示した．

また，Blin らの研究グループは，ニューロモーフィックなアーキテクチャを用いてグラフの PageRank を効率的に計算する方法を提案した [10]．通常のノイマン型のアーキテクチャと比較してスケーラビリティが高いことを示した．

3. スパイキングニューロンによるグラフ上の最短経路導出

3.1 スパイキングニューラルネットワークの理論

近年注目されている深層学習の要素技術である人工ニューラルネットワークは，神経回路を模して簡略化した数理モデルである．その一方，1952 年の Hodgkin-Huxley モデル [11] に始まり，脳神経系の電気生理学的な活動を忠実に再現する数理モデルも多数提案されている．これらの数理モデルはニューロン間の情報伝達にスパイクを用いるため，スパイキングニューラルネットワーク (Spiking Neural Network: SNN) と呼ばれる．ニューロモーフィックアーキテクチャの構成要素には，SNN が用いられる場合が多い．本稿では，数あるスパイキングニューロンモデルの中から最も単純でなモデルである Leaky integrate-and-fire(LIF) モデル [12] を扱う．

ニューロン N_i の膜電位を $V_i(t)$, 静止膜電位を V_{rest}, 入力電流を $I_i(t)$, 膜抵抗を R_i, 時定数を τ_i として，LIF モデルの膜電位は次の微分方程式を満たす．

$$\tau \frac{dV_i(t)}{dt} = -(V_i(t) - V_{\mathrm{rest}}) + R_i I_i(t) \qquad (1)$$

V_i が閾値 V_{th} を超えると，V_i は瞬間的にピーク電位 V_{peak} まで上昇し発火が起こる．その後，V_i はリセット電位 V_{reset} まで下がり，不応期 $t_{\mathrm{ref}}^{(i)}$ の間は膜電位が変化しなくなる．時間軸を離散化したシミュレーションを行う場合は，時間の刻み幅を Δt として，以下のように変形する．

$$dV_i(t) = \frac{(-(V_i(t) - V_{\text{rest}}) + R_i I_i(t))\Delta t}{\tau} \quad (2)$$

シミュレーションでの $V_i(t)$ の更新式は以下のようになる．

$$V_i(t + \Delta t) = V_i(t) + dV_i(t) \quad (3)$$

V_i が閾値 V_{th} を超えた場合に生じる，V_i のピーク電位までの上昇，リセット電位への下降は，以下の更新式で与える．

$$V_i(t) = V_i(t)(1 - p(i,t)) + V_{\text{peak}} p(i,t) \quad (4)$$

$$V_i(t + \Delta t) = V_i(t)(1 - p(i,t)) + V_{\text{reset}} p(i,t) \quad (5)$$

$$p(i,t) = \begin{cases} 1\ (V_i(t) \geq V_{\text{th}}) \\ 0\ (V_i(t) < V_{\text{th}}) \end{cases} \quad (6)$$

次に，ニューロンに生じた発火を次のニューロンに伝播させるシナプスについて考える．SNN におけるシナプス $s_{j\to i}$ はニューロン間を結合し，結合元ニューロン N_j の発火に応じて結合先ニューロン N_i の入力電流 $I_i(t)$ を変化させる役割を持つ．本稿では，$I_i(t)$ の変化が以下の式で定義される最も単純なシナプスを利用する．

$$I_i(t) = \sum_{j\to i} w_{j\to i} p(j, t - L_{j\to i}) \quad (7)$$

N_j から N_i へのシナプスの効率 $w_{j\to i}$, 遅延 $L_{j\to i}$ によって，結合先 N_i の $I_i(t)$ の変化量，変化するタイミングを設定できる．

3.2 グラフの SNN へのマッピング

グラフ上の最短経路問題を扱うために，グラフを SNN にマッピングする方法について説明する．

頂点集合 V，正の重み付き辺集合 E で構成される一般グラフ $G(V,E)$ 上で，ある 2 頂点間の最短経路を求める問題を考える．すなわち，$G(V,E)$ 上の 2 頂点，スタート v_{start} とゴール v_{goal} を結ぶ経路のうち，辺の重みの和が最小となるものを求める問題である．

Schuman らの先行研究と同様に，グラフ $G(V,E)$ を SNN へマッピングする [8]．まず，頂点 v_i をニューロン N_i に，辺 $e_{i,j}$ を両方向のシナプス $s_{i\to j}, s_{j\to i}$ に対応させる．また，全てのニューロンについて自己ループシナプス $s_{i\to i}$ を追加する．全てのニューロンは同一のパラメータ設定であり，全ての N_i について，初期膜電位 $V_i(0) = 0$，静止膜電位 $V_{\text{rest}} = 0$，膜抵抗 $R_i = 1$，時定数 $\tau_i = 1$, 閾値 $V_{\text{th}} = 1$，ピーク電位 $V_{\text{peak}} = 1$，リセット電位 $V_{\text{reset}} = 0$ とする．不応期 $t_{\text{ref}}^{(i)}$ に関しては，ニューロモーフィックアーキテクチャ上では非常に短い時間を想定しているが，CPU 上の時間軸を離散化したシミュレーションの場合は，$t_{\text{ref}}^{(i)} = \Delta t$ としてよい．シナプス効率は全て $w_{i\to j} = 1$ に統一し，自己ループ以外のシナプスの遅延 $L_{i\to j}$ は対応する元の辺 $e_{i,j}$ の重みと等しい値にする．自己ループシナプスの遅延は $L_{i\to i} = \Delta t$ に統一する．以上のように SNN を構成することで最短経路問題を解くことができる．

3.3 SNN を用いた最短経路の導出

SNN において，あるニューロンが発火すると，シナプスを通じて隣接するニューロンにスパイクが伝播する．前節で構成した SNN のパラメータ設定は，あるニューロン N_i に対して，隣接するニューロンからのスパイクの伝播が 1 回でも発生したら，即座に N_i が発火するようになっている．このような SNN の特徴を用いて，最短経路を決定する手法を提案する．

提案手法は，順探索と逆探索と呼ぶ二つの操作で構成される．まず順探索では，スタートノードに対応するニューロン N_{start} を刺激してからゴールノードに対応するニューロン N_{goal} が発火するまでの最短時間を求める．その後，逆探索で各時刻のニューロンの発火情報から最短経路を決定する．

3.3.1 順探索による最短時間の導出

順探索による最短時間の決定は以下のように行う．時刻 $t = 0$ で N_{start} に入力電流を与え，発火させる．発火はシナプスを通して隣接するニューロン間を伝播し続ける．その結果，N_{start} から N_{goal} に到達可能であれば，いずれは N_{goal} が発火する．順探索において時刻 t で N_i が発火しているとき，その時刻で v_{start} から v_i まで到達可能であることを意味する．よって，N_{goal} が初めて発火した時刻を $t = T$ とすると，T がゴールノードに対応するニューロンが発火するまでの最短時間となる．また，N_{goal} がいつまでも発火しない場合は，v_{start} から v_{goal} に到達不可能であることが分かり，到達可能性の判定も行うことができる．

静止した障害物のあるグラフの場合

ここで，最短経路を決定したいグラフ上に障害物が存在するような場合を考える．時刻 t において頂点 v_i を通過可能かどうかを以下のように表す．

$$f(v_i, t) = \begin{cases} 1\ (\text{時刻 } t \text{ で頂点 } v_i \text{ を通過できる}) \\ 0\ (\text{上記以外}) \end{cases} \quad (8)$$

例えば，ある頂点 v_a に静止した障害物があり，常に通過できないという条件を問題に加えると，$f(v_a, t) = 0\ (^\forall t)$ となる．グラフ上に障害物が存在しない場合は任意の時刻で全ての頂点を通過可能であるため，$f(v_i, t) = 1\ (^\forall i, ^\forall t)$ となる．

この条件で最短時間を求めたい場合，$f(v, t) = 0\ (^\forall t)$ となる v に対応するニューロンが発火しないように完全に停止させた状態で，順探索を行えばよい．通過できない頂点が複数存在した場合も，対応する複数のニューロンを停止させることで最短時間を求められる．

障害物が時間と共に移動する場合

これまでは，全ての頂点について，通過可能性が一定な場合の最短経路問題を扱ってきた．頂点毎の通過可能性が時間変化する場合，すなわちグラフ上の障害物が時間と共に動くような場合にも同様の順探索が適用できる．式 (8)

で与えられる $f(v_i,t)$ が t に依存するため，N_i の膜電位と不応期を以下のように変化させる．

$$V_i(t) = V_i(t)f(v_i,t) \tag{9}$$

$$t_{\mathrm{ref}}^{(i)} = \begin{cases} \Delta t\ (f(v_i,t)=1) \\ \infty\ (f(v_i,t)=0) \end{cases} \tag{10}$$

v_i を通過できないとき，$V_i(t)=0$, $t_{\mathrm{ref}}^{(i)}=\infty$ になり，N_i は発火しなくなる．各イテレーションでこの更新を加えた上で，同様に N_{start} から N_{goal} までの発火が伝播する時間を計測することで最短時間が得られる．

3.3.2 逆探索による最短経路の導出

順探索によりゴールノードが発火するまでの最短時間 T が求まると，以下に示す逆探索の操作で最短経路を導出することができる．

まず，順探索で利用した $f(v_i,t)$ を $f_0(v_i,t)$ とする．また，式 (6) にならって，時刻 t におけるニューロン N_i の発火の有無を $p_0(i,t)$ で表す．

$$p_0(i,t) = \begin{cases} 1\ (V_i(t) \geq V_{\mathrm{th}}) \\ 0\ (V_i(t) < V_{\mathrm{th}}) \end{cases} \tag{11}$$

逆探索では，頂点 v_i の通過可能性を表す関数として以下の $f_1(v_i,t)$ を用いる．

$$f_1(v_i,t) = p_0(i,T-t)f_0(v_i,T-t) \tag{12}$$

ニューロン，シナプスは順探索で扱ったものと同様のものを用いるが，逆探索においては v_{start} と v_{goal} の役割を入れ替える．つまり，時刻 $t=0$ で N_{goal} を発火させ，N_{start} が初めて発火した段階で計測終了となる．計測終了の時刻は順探索で得られた T と一致する．また，逆探索における時刻 t でのニューロン N_i の発火の有無を $p_1(i,t)$ で表すとする．ニューロンを指定するインデックス i と，時刻 t の 2 変数関数 $x(i,t)$ を以下のように定義する．

$$x(i,t) = p_1(i,T-t) \tag{13}$$

この $x(i,t)$ は，求めたい最短経路の情報を与えている．通常，最短経路問題の解は辺の集合で与えられるため，最短経路が複数存在する場合は辺の集合も複数与える必要がある．一方，$x(i,t)$ は「時刻 t で $x(i,t)=1$ ならば，v_i は最短時間で v_{goal} に到達する経路上の頂点である」ことを意味しているため，この表現で複数の解を含んでいる．また，最短で v_{goal} に到達するために静止が必要であったり，寄り道が許容されていた場合でも，対応する経路が $x(i,t)$ に網羅されている．

4. エージェントによる経路探索手法

本章では，前章と同様なグラフ $G(V,E)$ に対して，エージェントを v_{start} から動かし，なるべく早く v_{goal} に到達させる問題を考える．全ての t において各頂点の通過可能性 $f(v_i,t)$ が既知な場合，前章の順探索・逆探索で最短経路を求め，その経路に沿ってエージェントを動かすことで最短で v_{goal} に到達する．$f(v_i,t)$ が完全に既知でなく未来の通過可能性が分からない場合，各頂点の通過可能性がいつ変化するか分からないため，事前に最短経路を知ることができない．そのため，各時刻毎に現在地 v_{now} にいるエージェントの移動先 v_{next} を決めるアルゴリズムが別途必要になる．v_{next} は v_{now} に隣接している頂点か v_{now} そのものの中から選ぶ．

本章では貪欲エージェント，最短経路学習エージェント，確率過程学習エージェントの 3 種類を考える．1 つ目の貪欲エージェントは，$f(v_i,t)$ が完全に未知な場合でも利用できる．$f(v_i,t)$ の確率過程が与えられている場合は，さらに 2 つ目の最短経路学習エージェントや 3 つ目の確率過程学習エージェントが利用可能である．貪欲エージェントと確率過程学習エージェントはステップ毎に順探索・逆探索を行うことで v_{next} を決定する．

4.1 貪欲エージェント

$f(v_i,t)$ が未知であっても，t 以前のある時刻 t' において，$f(v_i,t')$ は既知であるとする．t' 以降の任意の時刻 t での頂点通過可能性を $f(v_i,t)=f(v_i,t')$ で固定し，以降は変化しないと仮定する．この条件でエージェントの現在地 v_{now} をスタート，v_{goal} をゴールとした順探索・逆探索による最短経路を求めることができる．貪欲エージェントは，まず各時刻毎に現在のグラフでの最短経路 $x(i,t)$ を求める．そして $t=0$ を除いて $x(i,t)=1$ となる (i,t) の組のうち，t が最小な組の i に対して，移動先を $v_{\mathrm{next}}=v_i$ と決める．各ステップごとに繰り返し最短経路を求める必要があるので，計算時間は長くなる．

4.2 最短経路学習エージェント

各頂点の到達可能性 $f(v_i,t)$ の確率過程をもとに，事前に順探索・逆探索で $x(i,t)$ を求める操作を M 回行う．n 回目の試行における $x(i,t)$ を $x^{(n)}(i,t)$ $(n=1,...,M)$ とおくと，$f(v_i,t)$ は毎回等しくなるとは限らないので，$x^{(n)}(i,t)$ も n によって異なる．これらの平均 $\bar{x}$ を求める．

$$\overline{x}(i,t) = \frac{1}{M}\sum_{n=1}^{M} x^{(n)}(i,t) \tag{14}$$

最短経路学習エージェントは，この $\overline{x}(i,t)$ を利用して移動先 v_{next} を決定する．v_i は $f(i,t)=1$ を満たす頂点であり，v_{now} に隣接している頂点か v_{now} 自身のうちのどれかである．これらを満たす v_i の中で，$\overline{x}(i,t+L_{\mathrm{now}\to i})$ が最大になる i をとり，$v_{\mathrm{next}}=v_i$ と決める．この方法は貪欲エージェントのような逐次的な最短経路の導出は無く，各ステップごとに複数の値の中から最大値をとるだけなの

Vol.2019-ARC-240 No.4
Vol.2019-SLDM-191 No.4
Vol.2019-EMB-53 No.4
2019/2/27

で，計算時間が非常に短くなる．

4.3 確率過程学習エージェント

各頂点の到達可能性 $f(v_i,t)$ の確率過程が与えられている場合，未来における各頂点の通過できる確率を計算できる．現在時刻 t' での $f(v_i,t')$ から確率過程を元に得られる，τ 秒後に v_i が通過不可能になる確率を $P(v_i,\tau)$ とする．また，確率閾値 $P_{\rm th}$ を設定する．確率過程学習エージェントは，まずの 15 式のような確率閾値で二値化した $f'(v_i,t)$ を通過可能性に設定し，$t=t'$ での現在地 $v_{\rm now}$ をスタート，$v_{\rm goal}$ をゴールとした順探索・逆探索を行う．

$$f'(v_i,t)=\begin{cases}0\ (P(v_i,t)\geq P_{\rm th})\\ 1\ (P(v_i,t)<P_{\rm th})\end{cases} \tag{15}$$

以降は貪欲エージェントと同様に，得られた $x(i,t)$ で $t=0$ を除いて初めて $x(i,t)=1$ となる組 (i,t) を求め，移動先を $v_{\rm next}=v_i$ と決める．

確率閾値 P_{th} が高い場合，障害物が存在する確率が高い頂点のみを通過不可能な頂点として扱う．よって，最短経路上に障害物があると想定すると，エージェントはその障害物の近傍を通ることをゴールに到達しようとすると考えられる．確率閾値 P_{th} が低い場合には，少しでも通過できない可能性のある頂点を通過不可能とみなして行動する．すなわち，エージェントはその障害物を大きく避けて遠回りでゴールに到達しようとする．

5. 確率過程を用いたエージェントの性能評価

5.1 各エージェントの性能評価

15×30 のグリッド上に 11×5 の長方形状の障害物があり，2s 毎に上下左右 1 マス移動，或いはその場での静止のいずれかがそれぞれ 1/5 の確率で行われる．$f(v_i,t)$ が確率分布に従って変化する場合を考える．最も左上のグリッドを座標 $(1,1)$，最も右下のグリッドを座標 $(15,30)$ とする．最短経路のスタート地点は $(8,30)$ でゴール地点は $(8,1)$ とし，障害物の中心の初期位置は $(8,15)$ であるとする．最短経路学習エージェントの学習回数 M は 100 とする．まず，この問題の SNN へのマッピングを行う．各グリッド内にニューロンを配置し上下左右又は斜めに隣接するニューロン同士を双方向のシナプスで結合し，各ニューロンに自己ループシナプスを追加する．各ニューロンやシナプスは 3 章のパラメータ設定に従う．また障害物に覆われているグリッドに対応するニューロンを通過不可能にする．

貪欲エージェント，最短経路学習エージェント，$P_{\rm th}=0.5,0.1,0.01$ の確率過程学習エージェントの計 5 種類の各エージェントが経路探索をおこない，$v_{\rm goal}$ に到達するまでのステップ数を比較する．ただし，最短経路学習エージェントの学習にかかる時間および各ステップで経路を決定するのにかかる時間は考慮していない．

表 1 に各エージェントのゴールへの到達ステップ数の平均と分散を示す．

表 1 各エージェントのゴールへの到達ステップ数の平均と分散 (4708 回)

エージェントの種類	到達ステップ数の平均	到達ステップ数の分散
貪欲	74.088	29.189
最短経路学習	77.339	35.577
確率過程学習 ($P_{\rm th}=0.5$)	85.776	77.968
確率過程学習 ($P_{\rm th}=0.1$)	75.205	3.841
確率過程学習 ($P_{\rm th}=0.01$)	73.489	3.406

表 1 より，到達時間の平均だけを見ると最も性能が良いのは $P_{\rm th}=0.01$ の確率過程学習エージェントであり，2 番目は貪欲エージェントである．

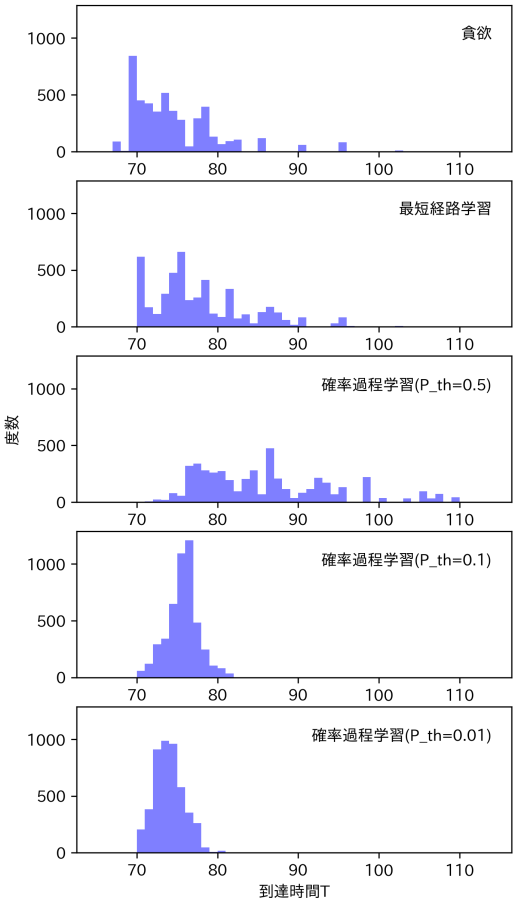

図 1 到達時間のヒストグラム (4708 回)

エージェント別の到達時間のヒストグラムを図 1 に示す．貪欲エージェントに関して，到達時間 $T=70$ 付近の度数が他のエージェントに比べて最も高いが，T の分散が大きく $T>80$ および $T<70$ の試行も存在する．各エージェントのすべての試行の中で，最も到達時間が短かったのは貪欲エージェントの 68 ステップである．

一方，$P_{\rm th}=0.1$ および $P_{\rm th}=0.01$ の確率過程学習エージェントはグラフの散らばりが比較的小さく，ほとんどの

試行が $70 < T < 80$ の範囲に収まっている．$P_{\mathrm{th}} = 0.5$ の確率過程学習エージェントは T の分散が非常に大きく，平均的にも到達時間が遅くなっている．確率過程学習エージェントのみで比較すると，確率閾値 P_{th} が小さくなればなるほど到達時間の平均および分散が小さくなり，ヒストグラムが左寄りになっている．

また，最短経路学習エージェントは平均分散ともに $P_{\mathrm{th}} = 0.5$ の確率過程学習エージェントと貪欲エージェントの間の結果となっている．

5.2 各エージェントの考察

各エージェントのすべての試行の中で，最も到達時間が短かったのは貪欲エージェントであった．これは，エージェントが各ステップで計算した最短経路を障害物が阻害しないように動いた場合の試行であると考えられる．同様に T が 90 以上となるのは，障害物が最短経路を阻害するように動いた場合である．貪欲エージェントのゴールまでの到達時間の分散が大きいのは，以上のように障害物の動き方に大きく依存するためであると考えられる．

最短経路学習エージェントは試行毎に等しい $\bar{x}(i,t)$ を用いているのでエージェントの移動先決定が障害物の現在地に依らない．そのため貪欲エージェントと同様に障害物の移動方法が到達ステップ数に強く影響し分散が大きくなったと考えられる．

また，確率過程学習エージェントについて 4.3 節で述べたように，確率閾値 P_{th} が小さいエージェントほど障害物を大きく避けてゴールへ到達する傾向がある．$P_{\mathrm{th}} = 0.1$ および 0.01 のエージェントにおいては，他のエージェントに比べて到達時間が平均的に短く，その分散も小さい傾向がある．これは，エージェントが障害物を大きく避けるような経路を選択しているため，経路探索が障害物の動きに依存せず，到達時間のばらつきが小さくなるためであると考えられる．$P_{\mathrm{th}} = 0.1$ および 0.01 の 2 つのエージェントで比べると，後者の方がその傾向は顕著である．

確率閾値の高いエージェントは，高い確率で通過可能でない点を通過可能であると判断してしまうので障害物とぶつかりやすくなる，つまり障害物の影響を大きく受けるため，到達時間が平均的に長く，分散も大きくなっていると考えられる．

6. まとめと今後の展望

本稿では，重み付きグラフを上手くニューロンにマッピングすることで，ニューロモーフィックコンピューティングを用いて最短経路問題を解くアルゴリズムを提案した．また，提案したアルゴリズムが，グラフの各頂点の到達可能性が時間変化する場合にも有効であることを示した．さらに，提案したアルゴリズムを用いて最短経路を決定しながら行動するエージェントを想定し，確率分布に従ってグラフを障害物が動く，すなわちグラフの頂点到達可能性が確率分布に従って変化する場合について性能評価を行った．その結果，障害物の確率分布を元に学習を行って経路探索するエージェントが最も良い性能であることがわかった．

グラフ上の障害物が時間と共に動く場合でも，ステップごとに経路探索を行う必要がない点が，他の経路決定アルゴリズムにはない提案手法の利点である．

動的な障害物がある状況で経路探索を行う現実の応用については様々なものが考えられる．提案手法において，障害物を避けてなるべくぶつかりにくい経路を選択することと，障害物にぶつかる危険を冒してでも早くゴールに到達する経路を選択することはトレードオフの関係にあるが，どちらが好ましいかは応用によって異なる．状況に応じて P_{th} を設定することで，上記のバランスを調整できる点も提案手法が優れている点の 1 つである．

本稿では簡単のためにグリッド状のグラフにおいてのみ評価実験を行った．一般のグラフでも適切に動作することを確認するのが今後の課題の 1 つとして挙げられる．また，SNN のハードウェアによる実装については盛んに研究されており，本稿で用いた Leaky integrate and Fire モデルはその単純さから，特にハードウェア実装と相性が良い．よって，それらを参考に FPGA などでニューロモーフィックなハードウェアを実装し，提案したアルゴリズムの実行効率を測定することも今後の課題として挙げられる．

謝辞 本研究の一部は，JST CREST 課題番号 JPMJCR18K1（研究課題名「エッジでの高効率なデータ解析を実現するグラフ計算基盤」）によるものである．

参考文献

[1] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[4] A. Zelinsky, R.A. Jarvis, J. C. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *In Proceedings of International Conference on Advanced Robotics*, pages 533–538, 1993.

[5] Bhavya Ghai and A. Shukla. *Wave Front Method Based Path Planning Algorithm for Mobile Robots*, pages 279–286. 01 2016.

[6] Anshika Pal, Ritu Tiwari, and Anupam Shukla. A fo-

cused wave front algorithm for mobile robot path planning. In Emilio Corchado, Marek Kurzyński, and Michał Woźniak, editors, *Hybrid Artificial Intelligent Systems*, pages 190–197, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[7] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, January 2018.

[8] Catherine Schuman, Kathleen Hamilton, Tiffany Mintz, Md Musabbir Adnan, Bon Ku, Sung-Kyu Lim, and Garrett Rose. Shortest path and neighborhood subgraph extraction on a spiking memristive neuromorphic implementation. pages 1–6, 03 2019.

[9] Kathleen E. Hamilton, Neena Imam, and Travis S. Humble. Community detection with spiking neural networks for neuromorphic hardware. In *Proceedings of the Neuromorphic Computing Symposium*, NCS ' 17, New York, NY, USA, 2017. Association for Computing Machinery.

[10] L. Blin, A. J. Awan, and T. Heinis. Using neuromorphic hardware for the scalable execution of massively parallel, communication-intensive algorithms. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 89–94, Dec 2018.

[11] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[12] L. LAPICQUE. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej*, 9:620–635, 1907.