Vol.2019-ARC-240 No.22
Vol.2019-SLDM-191 No.22
Vol.2019-EMB-53 No.22
2019/2/28

重み付きグラフの最大マッチング問題における脳型計算を用いた近似解法の検討

上野 洋典[1,a)]　近藤 正章[1]

概要：近年，非ノイマン型の計算パラダイムとしてニューロモーフィックコンピューティング（脳型計算）が注目を集めている．本稿では脳型計算を用いて重み付きグラフの最大マッチング問題を近似的に解く手法を提案し，その手法が近似度 1/2 の Greedy アルゴリズムと同一の処理を行う手順であることを示す．特に，規模の異なるランダムグラフに対する提案手法を適用し，近似解の性能および計算量を評価した．その結果，大規模なグラフに対しては，厳密解に近い解が得られ，提案手法の優位性を確認した．また，提案したアルゴリズムを実行するニューロモーフィックなアーキテクチャを FPGA に実装し，実行時間を評価した．

1. はじめに

半導体プロセスの微細化によって計算機の実行効率は指数的に増加してきたが，微細化の物理的限界によりその伸びが鈍化し，いわゆるムーアの法則が終焉を迎えつつある．現在のコンピュータは，命令を記憶装置に格納するノイマン型アーキテクチャのものがほとんどである．ノイマン型コンピュータにおいては，命令の実行のためには必ず記憶装置にアクセスする必要があり，プロセッサとメモリ間のアクセス速度がコンピュータ全体のボトルネックとなる．これはノイマンボトルネックと呼ばれている．プロセッサの処理速度の性能向上に比べ，記憶装置へのアクセス速度の性能向上は小さいため，ノイマンボトルネックがより重要な問題になっている．そのため，ポストムーア時代に向け非ノイマン型コンピュータアーキテクチャに注目が集まっている．その一つとして，ニューロモーフィックコンピューティング（脳型計算）がある．

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．IBM の TrueNorth[1] や Intel の Loihi[2] がその例である．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．

一方，近年ソーシャルネットワーク（SNS）の発展などにより，大規模なグラフ処理を高効率に行う需要が高まっている．グラフは，頂点と頂点間の連結関係を表す辺からなるデータ構造である．単純な構造で多くの情報を表現できるという長所を持つ一方，データへのアクセスがランダムで局所性が少なく，キャッシュを上手く活用できないためメモリアクセスの時間が膨大になり，通常の CPU では効率的に処理を行うことが難しいという問題がある．

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重み付き最大グラフマッチング問題を近似的に解くアルゴリズムを提案し，そのアルゴリズムが近似度 1/2 の Greedy アルゴリズム [3] と同一の処理を行う手順であることを示す．さらに，規模の異な

[1] 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)] ueno@hal.ipc.i.u-tokyo.ac.jp

るランダムグラフに対して提案手法を適用し，近似解の性能を評価する．また，提案したアルゴリズムを実行する回路をハードウェア記述言語で実装し，回路面積および実行時間の評価を行う．

2. 関連研究

本章では，本稿の関連研究について述べる．主に，グラフのマッチング問題を解くアルゴリズムとその応用についての研究，脳型計算を用いてグラフを扱う方法についての研究，およびスパイキングニューラルネットのハードウェアによる実装についての研究について述べる．

2.1 グラフのマッチングとその応用についての研究

グラフ理論においてマッチングとは，グラフの辺集合で互いに端点を共有しないもののことを指す．特にこれ以上辺を追加できないマッチングのことを極大マッチング，マッチングのうち辺数が最大のものを最大マッチングという．また，グラフ上の全ての頂点がマッチング中のいずれかの辺の端点になっている時，そのマッチングを完全マッチングという．極大マッチングおよび最大マッチングは任意のグラフに存在するが，完全マッチングは必ずしも存在するとは限らない．

重み付きグラフにおいてもマッチング問題を考えることができる．最大重みマッチング問題とは，与えられた重み付きグラフに対して，マッチングに含まれる辺の重みの合計を最大化するようなマッチングを求める問題である．この際，マッチングに含まれる辺の数についての制約はない．

最大重み最大マッチング問題とは，マッチングに含まれる辺の数が最大マッチングと同数で，かつ辺の重みの合計を最大化するマッチングを求める問題である．同様に，最大重み完全マッチング問題は，そのマッチングが完全マッチングであるという制約のもと，辺の重みの合計を最大化するマッチングを求める問題である．また，最小重み最大マッチングおよび最小重み完全マッチングについても同様にして考えることができる．

対象とするグラフが 2 部グラフである場合，割当問題に帰着することで Hungarian 法 [4] などのアルゴリズムを用いて最大重み最大マッチングを求めることができる．一般のグラフに対しては，頂点数 $|V|$, 辺数 $|E|$ に対して $O(|E||V|^2)$ で最大マッチングを求めることができる Edmonds の blossom アルゴリズムが存在し，よく用いられている [5]．また，Vazirani らのグループにより，$O(\sqrt{|V|}|E|)$ で実装可能なアルゴリズムも提案されている [6].

マッチング問題は様々な分野に応用でき，近年では量子コンピュータにおけるエラー訂正にも用いられている．量子コンピュータの情報担体である量子ビットは，観測すると状態が確定して量子もつれ状態が壊れてしまうので，エラー検出の際にも直接量子ビットを観測することはできない．そのため，情報を表す量子ビットとかけ合わせる “アンシラビット” を加え，それらを観測することで量子ビットを直接観測せずに状態を調べる必要がある．さらに，1 つの論理量子ビットを表すために複数の物理量子ビットおよびアンシラビットを用いて冗長化することで，エラー耐性のある量子ビットを実現している．符号化方式として，最も有力視されているのが Surface code[7] である．Surface code は二次元正方格子状に並べられた物理量子ビット上で実行できる．隣接したデータビットで特定のエラーが起きると符号が反転するアンシラビットを構成し，これを二次元格子状に展開することでエラーの場所と種類を特定することを可能にしている．エラーが鎖のように空間的に連続して発生する，およびその鎖が短くなるようなエラーのほうが発生しやすいという仮定を置くと，アンシラビットの符号からエラー位置を特定するのは最小重み完全マッチング問題に帰着できる．そのため，Surface code を用いた量子エラー訂正では主に blossom アルゴリズム [5] を実行することでエラー訂正が行われている．

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

グラフ上で最短経路を探索するアルゴリズムとして，Wavefront アルゴリズムが知られている [8]．Wavefront アルゴリズムは，全ノードに対してでスタート地点からの距離コストを幅優先探索で設定していき，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．Schuman らは 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，脳型計算のアーキテクチャにグラフをうまくマッピングす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方法を提案した [9]．同様のグラフのマッピングを用いて，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 [10].

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 [11]．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が報告されている．

2.3 スパイキングニューラルネットのハードウェアによる実装

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 [12] に始まり, 脳神経系の電気生理学的な活動を忠実に

Vol.2019-ARC-240 No.22
Vol.2019-SLDM-191 No.22
Vol.2019-EMB-53 No.22
2019/2/28

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．脳型計算のアーキテクチャは，この SNN をアナログ回路あるいはデジタル回路で実装することで実現されている場合が多い．

Pani らのグループは，SNN モデルの 1 つである Izhikevich モデル [13] を FPGA で実装した [14]．一つのニューロンは微分方程式の計算を行う Nueron section と，隣接しているニューロンから入力されるスパイクの重み付き和を計算する Synapse section から成る．各セクションごとに SNN モデルのパラメータや結合重みを格納する RAM を持っており，分散共有メモリの構成を取っている．64 個のニューロンを 1 ユニットとして，4 ユニットと全体のスパイク情報を保持する Spike Register から 1 つのネットワークを構成する．Xilinx 社の Virtex-6 上に実装し，ニューロン 1440 個の全結合ネットワークを 10kHz で動作させている．

本稿では Pani らの FPGA 実装を参考にして，提案したアルゴリズムを実行する回路を実装する．詳細は 4 章で述べる．

3. 提案手法

本章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重み付き最大グラフマッチング問題を解く近似アルゴリズムを提案し，その詳細について述べる．3.1 節ではアーキテクチャを構成するニューロン素子の特徴について述べ，3.2 節では重み付きグラフをそれらにマッピングする方法について述べる．また，3.3 および 3.4 節では，Greedy と提案手法が本質的に同一の処理を行う手順であることについて述べる．

詳細は 3.3 節で述べるが，最小重み最大マッチング問題は簡単な操作で最大重み最大マッチング問題へと変更できる．以下，本稿では最大重み最大マッチング問題のみを扱うとする．

3.1 ニューロン素子

本手法で用いるニューロン素子は以下のような特徴を持つ．すなわち，

- 活動状態を表す内部変数 F を持つ
- 発火までのディレイを表す内部変数 D を持つ
- F=1 のとき，1 ステップごとに D の値を 1 減らす
- 接続した他のニューロン素子が発火した場合は内部変数 F を 0 にする
- D が 0 になると発火を表す内部変数 S の値を 1 にし，F を 0 にする

以上の 5 つである．

それぞれのニューロン素子は初期状態として $F = 1, S = 0$ であり，D はニューロン毎に特定の値を持っている．ニューロン素子同士を必要に応じて接続することでネットワークを構築する．全てのニューロンが活動を抑制されるまたは発火した場合，すなわち全てのニューロンについて $F = 0$ となった場合，ネットワークは終状態に到達したと表現する．図 1 に 1 つのニューロンの処理をフローチャートで示す．

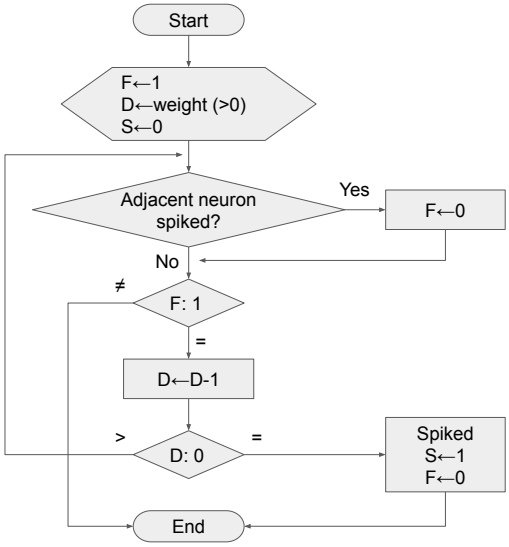

図 1 ニューロン 1 つの処理のフローチャート

3.2 重み付きグラフのニューロンへのマッピング

対象とするグラフは辺に重みが付いた単純グラフ（同一頂点間に複数の辺およびループを持たないグラフ）であるとする．重み付き単純グラフ G の線グラフ $L(G)$ を考える．線グラフとは，G の辺に一対一対応する頂点を持ち，G で端点を共有する 2 本の辺に対応する $L(G)$ の 2 個の頂点を辺で結んで得られるグラフである．G の辺の重みを，$L(G)$ の対応する頂点の重みとすることで，頂点に重みの付いたグラフ $L(G)$ が得られる．線グラフ導出の概要を図 2 にまとめる．

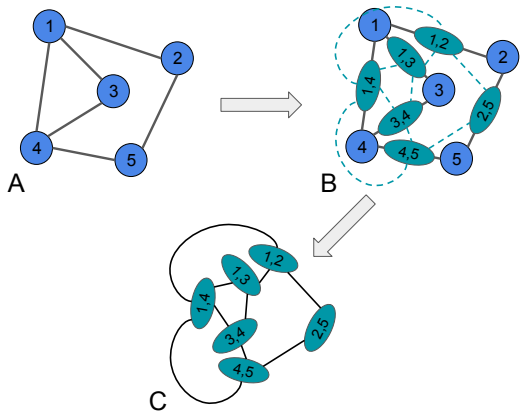

図 2 線グラフ導出の概要．A: 元のグラフ G．B: 辺を頂点とし，端点を共有する辺に対応する頂点同士を辺で繋ぐ．C: 得られた線グラフ $L(G)$.

この頂点重み付きグラフ $L(G)$ の各頂点の重みと，前節で述べたニューロン素子の発火ディレイ D の初期値を対応させることで，グラフの各頂点をニューロン素子にマッピングする．$L(G)$ において隣接している点に対応しているニューロン素子同士を接続することで，グラフ全体をニューロン素子のネットワークに対応させる．

3.3 アルゴリズム実行手順

前節のように解きたいグラフを SNN モデルへマッピングし，ニューロン素子のネットワークを構築する．その後，全ニューロン素子の変数 F を 1 とし，ステップを進めていく．これにより，$L(G)$ の小さい重みの頂点に対応するニューロンが発火し，隣接しているニューロンの活動を抑制する．ニューロンの活動の抑制は変数 F の値を 0 とすることで表す．このように，全てのニューロンが発火または活動を抑制されるまでステップを進める．$L(G)$ の重みの最大値と同じだけステップを進めると必ず計算が終了する．発火したニューロンに対応する $L(G)$ の頂点，およびそれに対応する G の辺を取ることで，マッチングが得られる．この手順を擬似コードの形で Algorithm1 に示す．

また，このアルゴリズムにより得られるのは最小重み最大マッチングの近似解であるが，各重みを（$L(G)$ の重みの最大値 - 元の重み）と設定し直すことで最大重み最大マッチング問題を求めることができる．

Algorithm 1 Neuro Graph Matching Algorithm

```
Input: Weighted graph G(V, E)
Output: M_neuro
  Initialize :
  M_neuro := ∅
  Get line graph L(V', E') from G(V, E);
  for v'_i ∈ V' do
    neuron[i].D = v'_i.weight;
  end for
  main LOOP :
  while Any of neuron[i].F == 1 do
    for each i do in parallel
      if neuron[i].F == 1 then
        decrement neuron[i].D;
      end if
      if neuron[i].D == 0 then
        add e_i ∈ E to M_neuro; (e_i is corresponding to v'_i)
        neuron[i].F = 0;
        for each j in adjacent of neuron[i] do in parallel
          neuron[j].F = 0;
        end for
      end if
    end for
  end while
```

隣接するニューロン同士の重みの初期値が同じ値だった場合，隣接するニューロン同士が同時に発火してしまい，アルゴリズムにより得られる解がマッチングとならない可能性がある．これを回避するためには，各ニューロンにインデックスをつけ，同時に発火した場合はインデックスの若い方を優先するなど，適当な方法で優先順位を導入すれば良い．または，重みの大小関係を保ったまま各重みに適当な値を足し引きして，各重みの値が重複しないようにする方法も考えられる．本稿では前者の方法を取り，隣接するニューロンが同時に発火するのを防ぐ．

この操作により得られた解は，近似度が 1/2 の Greedy アルゴリズム [3] を適用して得られる解と同一である．詳細は 3.4 節で述べる．

3.4 提案手法と Greedy アルゴリズムが同一の処理を行うことの説明

重み付きグラフの最大重み最大マッチングを求める近似アルゴリズムである Greedy アルゴリズム [3] を以下の Algorithm2 に示す．

Algorithm 2 GREEDY-Algorithm

```
Input: weighted graph G(V, E)
Output: M_greedy
  M_greedy := ∅
  while E ≠ ∅ do
    take an edge {a, b} ∈ E with highest weight;
    add {a, b} to M_GREEDY;
    remove all edges incident to a or b from E;
  end while
```

Algorithm 1 において，ニューロンのディレイの値を順にディクリメントしていき，0 になったものが発火するという点が，Algorithm 2 において残っている辺の中で重みが最小のものを選ぶ操作に対応する．また，Algorithm 1 の発火したニューロンに隣接するニューロンの活動が抑制される点が，Algorithm 2 のマッチングに加えた辺と端点を共有する辺を E から除外する操作に対応する．

以上のことから，提案手法は Greedy アルゴリズムと同一の処理を行う手順であることが示された．

4. ハードウェアによる実装

本章では提案したアルゴリズムを実行する脳型計算のアーキテクチャの実装について説明する．まず，4.1 節でアーキテクチャ全体の構成について述べる．そして，4.2 節で計算のコアとなるニューロン素子の詳細について述べる．また，性能評価については次章の 5.2 節で述べる．

4.1 全体アーキテクチャ

図 3 に提案するアーキテクチャの概要を示す．

マッチング問題を解きたいグラフ $G(V, E)$ の辺の数 $|E|$ だけニューロン素子を用意して，それらをネットワークで接続する．グラフ G の隣接行列および重みは DRAM を介してホストコンピュータから与えられる．重みは各ニューロンに対して 8bit で与え，隣接行列は全体で $|E| \times |E|-$bit である．隣接行列のうち，各ニューロンにはそのニューロ

Vol.2019-ARC-240 No.22
Vol.2019-SLDM-191 No.22
Vol.2019-EMB-53 No.22
2019/2/28

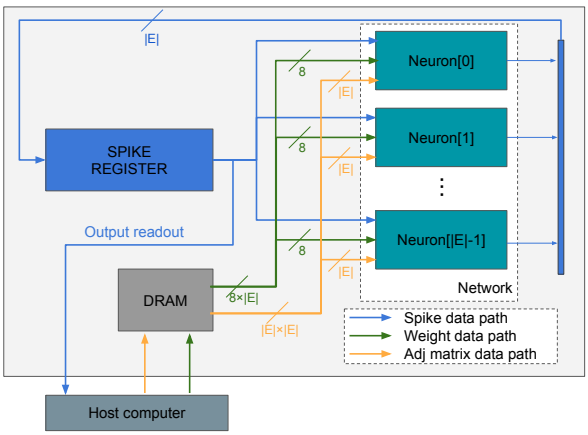

図 3　提案するアーキテクチャの概要図

ンの接続情報を表す行のみを入力する．

また，Spike レジスタは $|E|$bit のレジスタで，各ニューロンの発火情報を保持している．この全てのニューロン素子が発火もしくは活動を抑制された後にこのレジスタの情報を読み取ることでマッチング問題の解を得ることができる．図では省略しているが，各ニューロンおよび Spike レジスタは同一のクロック信号で駆動されている．

重みおよび隣接行列の情報を格納するのに必要なメモリは $|E|^2+8\times|E|$ となる．しかし，これは隣接行列を密行列の形で保持した場合に必要なメモリサイズである．隣接行列は疎行列となることがあるので，その場合には疎行列表現を用いることで情報を圧縮することが出来る．さらに，対象とするグラフが完全グラフのみという制約があった場合，隣接情報を表現するのに必要なメモリ量は $O(|E|)$ まで落とすことができる．

4.2　ニューロン素子

提案アーキテクチャにおいて演算コアとなるニューロン素子の構成を図 4 に示す．

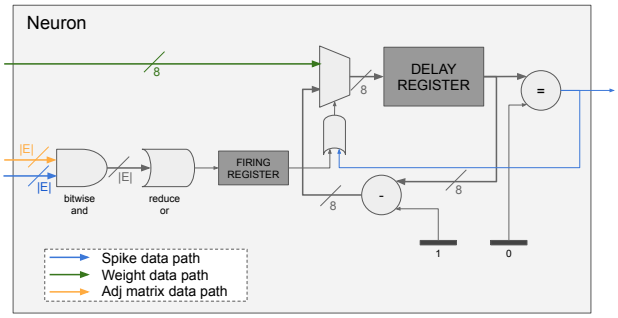

図 4　ニューロン素子

各ニューロンは $|E|$−bit の隣接情報およびスパイク情報と 8bit の重み情報を入力として受け取る．図中の Firing レジスタは変数 F を表しており，初期状態は 1 である． それにより，マルチプレクサでは重みの入力が選択され Delay レジスタが重みの値で初期化される．スパイク情報と隣接情報の 2 つは，ビットごとに AND を取った後に reduce OR を計算している．reduce OR は，$|E|$−bit の入力に対して，全ての bit が 0 であれば 0，少なくとも 1 つ以上の bit が 1 であれば 1 を返す，$|E|$−bit 入力 1bit 出力の演算である．これにより，reduce OR の出力は，“このニューロンに隣接しているかつ発火しているニューロンが少なくとも 1 つある” ときに 1，それ以外に 0 を取る．これを自身の spike 情報と OR を取ることで，活動が抑制されていないかつ自身が発火していないときには，Firing register に 0 が入力される．

Delay レジスタは重みの入力で初期化された後，Firing レジスタの値が 0 である限り，1 ステップごとに値がディクリメントされていく．Firing レジスタが 1 になる前に Delay レジスタの値が 0 になると，スパイク情報が 1 となりニューロンの発火を表す．

5.　評価

本章では，提案手法の性能と実行時間を評価する．5.1 節では，提案したアルゴリズムの近似解法としての性能を評価し，5.2 節では 4 章で実装した，提案アルゴリズムを実行するアーキテクチャを FPGA にマッピングした際の実行時間を評価する．

5.1　近似解の性能評価

本節では頂点数，および辺数を変えながらランダムに生成した重み付きグラフに対して提案手法を実行し，既存手法との乖離度を評価する．また，重みの値の種類が少なく，同一重みのノードが隣接することが多くなる場合，適当に与えた優先順位の寄与により提案手法の近似性能は小さくなると考えられる．よって，重みの値が全ての辺で異なる場合および，各辺の重みが 1〜10 の 10 種類の値をランダムに取る場合で性能を評価する．図 5 に頂点数が 10，20，30 の場合の結果を載せる．

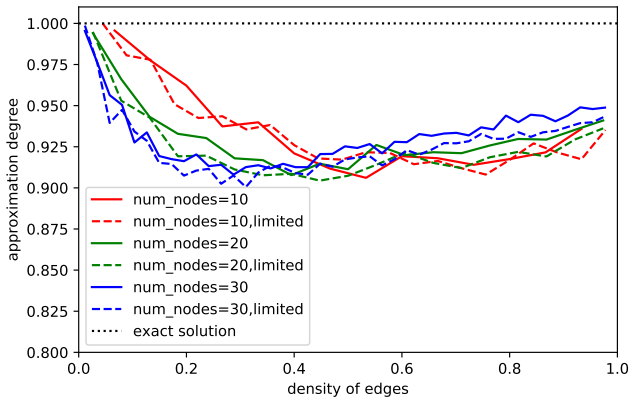

図 5　グラフの辺の密度と近似度の関係

横軸は (辺の数/完全グラフの辺の数) で表される “辺の密度” を表しており，縦軸は (近似手法による重みの和/厳

Vol.2019-ARC-240 No.22
Vol.2019-SLDM-191 No.22
Vol.2019-EMB-53 No.22
2019/2/28

表 1 提案手法により最大マッチングおよび厳密解が得られた回数.
100 回のうちの回数を (最大マッチング/厳密解) で示している.

	density of $\|E\|$									
$\|V\|$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
10	92/89	65/55	66/48	54/32	51/30	49/20	64/24	74/26	93/21	100/32
20	57/40	29/13	17/4	23/2	31/2	42/2	55/1	73/1	86/5	100/2
30	29/9	12/2	7/0	14/1	26/0	37/0	57/0	77/0	90/0	100/0

密解の重みの和) で表される近似度を表している．1 に近いほど厳密解に近い解が得られていることを示している．また，近似度は各条件ごとに 100 回の試行の平均をプロットしている．

各色ごとに，実線が重みの値が全て異なる場合，破線が重みが 1〜10 の値を取る場合を示している．また，黒の点線は近似度 1.0 すなわち厳密解を示している．

どの条件においても近似度の平均は 0.9 を超えている．また，辺の重みの値が全て異なる場合に比べて，重みの値が 10 種類に制限されている場合のほうが僅かに近似度が低い結果となっている．辺の密度が小さい場合には近似度が高く，厳密解およびそれに非常に近い解が得られている．辺の密度が大きくなるにつれて近似度は減少していくが，さらに密度が大きくなりグラフが完全グラフに近づいていくと近似度が少し上昇する傾向が見られる．

また，提案手法により得られたマッチングは極大マッチングであることが保証されているが，それが最大マッチングあるいは完全マッチングであることは保証されていない．表 1 はいくつかの条件でランダムグラフに 100 回適用し，提案手法により最大マッチングおよび最大重み最大マッチングすなわち厳密解が得られた回数を示している．なお，重みの値は 1〜10 の 10 種類を取る場合を考える．

頂点数が少ない方が比較的最大マッチングおよび厳密解が得られる回数が多い．どの頂点数の場合でも，辺の密度が 0.1〜0.2 と小さいときには比較的最大マッチングおよび厳密解を得やすい．これは，辺数が少ないために取りうるマッチングの候補が少なく，最大マッチングを得やすいからであると考えられる．また，辺の密度が 1.0 すなわち完全グラフの場合は必ず最大マッチングを得られている．これは，完全グラフにおいては全ての頂点同士が辺で結ばれているため，極大マッチングと最大マッチングの区別がなくなっているためである．さらに，頂点数が偶数の場合は完全マッチングもそれらと同一になることがわかる．よって，辺の密度が 0.8〜1.0 の場合はグラフが完全グラフに近づいているため，最大マッチングを得やすくなっていると考えられる．これらの中間である，辺の密度が 0.4〜0.6 程度の場合が最大マッチングおよび厳密解を最も得にくくなっている．以上より，図 5 で見られた近似度の傾向と合致する結果が得られた．

5.2 実行時間および消費電力の評価

4 章で提案したニューロモーフィックアーキテクチャを，ハードウェア記述言語で記述し，論理合成を行い評価する．ニューロンの数が 128 個の場合について論理合成を行う．ターゲットデバイスは Xilinx 社の Kintex-7 であり，論理合成には Xilinx Vivado 2018.2 を用いた．表 2 に評価結果を示す．

表 2 論理合成による評価結果

FPGA Device Family	Xillinx kintex-7
Parts	xc7k160tffg676-1
Clock Frequency [MHz]	200
Worst Negative Slack [ns]	1.905

提案手法と同一の処理を行う手順である Algorithm 2 を CPU で実行すると，その計算時間はグラフの辺数 $|E|$ に対して $O(|E|)$ のオーダーで大きくなる．一方，上記で実装したアーキテクチャにおいて，各ニューロンに RAM から読み出した重みの初期値を与えてから計算が終了するまでのステップ数は，解きたいグラフの重みの最大値以下であり，ニューロンの個数すなわち解きたいグラフの規模には関係がない．

以上のことから，解きたいグラフの規模が大きければ，CPU に対して提案したアルゴリズムおよび実装したアーキテクチャが優位であると考えられる．

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小重み最大マッチング問題を近似的に解くアルゴリズムを提案し，そのアルゴリズムが近似度 1/2 の Greedy アルゴリズムと同一の処理を行う手順であることを示した．そして，種々のランダムグラフに対して提案手法を適用してその近似性能を評価した．その結果，辺が疎なグラフ，あるいは辺が非常に密で完全グラフに近いグラフに対しては，完全マッチングおよび厳密解を得やすいことがわかった．

また，提案アルゴリズムを効率的に実行するニューロモーフィックなアーキテクチャを FPGA 上に実装し，実行時間と消費電力を評価した．評価の結果，解きたい問題の規模が大きくなると CPU に比べて非常に高効率に実行可能であることがわかった．

現時点の FPGA での実装では，与えられた問題を解くのに必要なニューロンを全て FPGA 上に展開し，空間的な並列性を利用して効率的に提案アルゴリズムを実行してい

る．しかしながら，問題のサイズが非常に大きくなった場合には，必要なニューロンを全て空間的に展開可能なハードウェアリソースが必要となり，これは現実的ではない．よって，計算を時間方向に展開するような実装を検討している．すなわち，与えられたグラフの一部をニューロンにマッピングして処理を行い，次のステップではグラフの別の部分をニューロンにマッピングして処理を行う，という方針で大規模なグラフを扱う方法である．

本研究の応用先として，量子コンピュータのエラー訂正を検討している．2.1 節で述べたように，surface code という符号化手法を用いてエラー耐性のある量子コンピュータを構築する場合には，重み付きグラフの完全マッチング問題を解く必要がある．現在は blossom アルゴリズムが用いられるのが一般的である．例えば提案手法により，マッチング問題を解くのにかかる時間が 10 分の 1 になった場合，1 回のエラー訂正で訂正しなければならない量子ビットのエラーも 10 分の 1 になると考えられ，解くべき問題も易しくなるため近似解法でも十分な性能が期待できる．また，surface code において想定するグラフは完全グラフであるため，5.1 節で見たように提案手法でも完全マッチングを得られることが保証されている．以上のことから，提案手法の量子エラー訂正への応用可能性は十分高いと考えられる．

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」），および JST 未来社会創造事業 課題番号 JPMJMI18E1（研究課題名「低炭素ＡＩ処理基盤のための革新的超伝導コンピューティング」）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] 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.

[3] David Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13(4):475–493, 1983.

[4] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1 - 2):83–97, 1955.

[5] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[6] Silvio Micali and Vijay Vazirani. An $o(\mathrm{sqrt}(|v||e|)$ algorithm for finding maximum matching in general graphs. pages 17–27, 10 1980.

[7] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] E. M. Izhikevich. Simple model of spiking neurons. *Trans. Neur. Netw.*, 14(6):1569–1572, November 2003.

[14] Danilo Pani, Paolo Meloni, Giuseppe Tuveri, Francesca Palumbo, Paolo Massobrio, and Luigi Raffo. An fpga platform for real-time simulation of spiking neuronal networks. *Frontiers in Neuroscience*, 11:90, 2017.