モンテカルロ木探索を用いたグラフニューラル構造探索

佐々木勇和†

† 大阪大学大学院情報科学研究科
E-mail: †sasaki@ist.osaka-u.ac.jp

あらまし グラフ深層学習（GNN）は，多様な領域のデータサイエンスタスクにおける強力なツールであり，幅広い応用シナリオで GNN は活用される．しかし，多様なグラフに対して最適な GNN 構造を設計および選択するのは困難な作業である．そのため，人間の労力を削減するために，グラフニューラル構造探索（Graph NAS）が既存のコンポーネントを組み合わせた準最適な GNN 構造を探索するために使用されている．しかし，効率性と様々なグラフへの適応性を満たすグラフニューラル構造探索手法は現状存在しない．そこで，本論文では効率的かつ多様なグラフに適応可能なグラフニューラル構造探索手法を提案する．提案手法は (i) 様々なグラフに適応可能なシンプルな探索空間と (ii) モンテカルロ木探索を用いた探索アルゴリズムで構成されている．探索アルゴリズムでは，ニューラルモデルを使用しないモンテカルロ木探索により，GNN 構造を効率的に探索することが可能である．評価実験により，提案手法は最先端のグラフニューラル構造探索手法と比較してノード分類の正解率を最大 12.3 増加させ，実行時間を最大 88 %削減することを示す．

キーワード グラフ深層学習，ニューラル構造探索，モンテカルロ木探索

1 はじめに

グラフ深層学習 (GNN) はデータサイエンスタスクにおいて強力なツールであり，様々な応用にて用いられている．例えば，化学 [1,2] や物理 [3]，ソーシャル科学 [4]，神経科学 [5] などが代表的な例である．多様なグラフに対して GNN 構造が開発されている一方で，全てのグラフに最適な GNN 構造は存在しない．そのため，グラフの特性に合わせて新しい GNN 構造の開発や既存の GNN 構造を選択することが必要であるが，多くの労力を要する作業である．

GNN の設計や選択における人間の労力を削減するために，グラフニューラル構造探索が準最適な GNN 構造の探索に用いられている [6–8]．GNN 構造の増加とグラフ分析の需要の増大から，グラフニューラル構造探索の重要性は増している．

グラフ構造探索の課題. グラフ構造探索には二つの技術的な課題，(1) 探索空間と (2) 探索アルゴリズムがある．まず，探索空間はグラフ構造のパターンを決定し，探索空間内のコンポーネントの組み合わせにより GNN 構造は生成される．そのため，探索空間が適切に設計されていない場合，与えられたグラフに対して適切な GNN 構造が存在しない可能性がある．次に，探索アルゴリズムは GNN 構造の探索順を決定する．膨大なパターンの GNN 構造を全て探索することは困難であるため，いかに優先的に高精度な GNN 構造を発見するかが重要である．探索アルゴリズムが洗練されていない場合，探索に時間がかかることや適切な GNN 構造が発見できない可能性がある．

既存のグラフニューラル構造探索技術は精度と効率を改善するために様々なアプローチが用いられている．複雑なコンポーネントを含む探索空間からニューラルモデルに基づく探索アルゴリズムにて GNN 構造が探索されているため，探索時間が長いことおよび発見された GNN 構造の分析が困難であることがあげられる．加えて，探索空間は主としてホモフィリックグラフを対象として設計されており，ヘテロフィリックグラフへの適用性は低いという課題がある．

研究動機. グラフニューラル構造探索を研究者と利用者の二つの観点から再考する．まず，研究者においては，グラフニューラル構造探索は，多様なグラフに対する高精度なベースラインとして利用できること，および GNN 構造の設計における重要なコンポーネントを把握できる必要である．利用者においては，GNN の知識や，グラフの特性，重いハイパーパラメータチューニングを必要とせずに，効率的かつ簡単に準最適な GNN 構造を多様な構造に対して発見できる必要がある．これらの必要性から，シンプル，高速，かつ多様なグラフに適応可能なグラフニューラル構造探索技術が必要である．

筆者らの知る限り，これらの必要性を満たすグラフニューラル構造探索は存在しない．例えば，GraphGym [9] は GNN 構造の重要なコンポーネントを理解することを目的としており，シンプルな探索空間を設計している．しかし，探索アルゴリズムはランダム探索である点およびヘテロフィリックグラフは対象としていない．Auto-HeG [10] はヘテロフィリックグラフを対象としたグラフニューラル構造探索技術である．しかし，探索が非効率である点と探索空間に複雑なコンポーネントが含まれているため重要なコンポーネントの理解が難しい．また，ホモフィリックグラフとヘテロフィリックグラフにおける重要なコンポーネントを実践的に比較した研究は存在しない．そのため，ホモフィリックグラフとヘテロフィリックグラフの両方に適用可能かつ高速なグラフニューラル構造探索技術の開発が必要である．

貢献. 本研究はシンプル，高速，かつ多様なグラフに適用可能なグラフニューラル構造探索技術を提案する．提案するグラフ

ニューラル構造探索技術はシンプルな探索空間とニューラルモデルを伴わないモンテカルロ木探索を活用した探索アルゴリズムから構成される．まず，探索空間はホモフィリックグラフとヘテロフィリックグラフの両方で高精度となるように設計されている．MLP，活性化関数，ジャンピングナレッジなどの基礎的なコンポーネントのみが含まれており，最新の GNN 構成は含まれていない．そのため，シンプルな GNN 構造が構築され，容易に構造が理解可能である．次に，モンテカルロ木探索を活用した探索アルゴリズムはニューラルモデルを用いないため，学習の必要がなく高速である．探索済み GNN 構造の平均精度を用いて，次に検証する GNN 構造の優先度を決定する．

評価実験では，既存のグラフニューラル構造探索と比較して，最大で正解率が 12.3 向上し，88%の探索時間が減少することを示す．さらに，提案手法で発見した構造を分析することで，ホモフィリックグラフとヘテロフィリックグラフにおける GNN 構造の違いを分析可能であることを示す．

再現性. コードを公開する．[1]

2 事前準備

2.1 グラフ

本研究は節点属性およびラベル付きの無向グラフ $\mathcal{G} = (\mathbf{S}, \mathbf{X}, \mathbf{Y})$ を対象とする．$\mathbf{S} \in \{0,1\}^{n \times n}$ は隣接行列，$\mathbf{X} \in R^{n \times d}$ は節点属性の行列，および $\mathbf{Y} \in \{0,1\}^{n \times y}$ は節点のラベルを表す行列であり，n, d, および y はそれぞれ節点数，各節点の属性数，およびラベルの種類数を表す．節点 v と u 間に枝が存在する場合，$\mathbf{S}_{(v,u)}$ と $\mathbf{S}_{(u,v)}$ は 1 となる．$\mathbf{Y}_v$ はワンホットベクトルであり，節点 v のラベルを表す．

2 つのグラフ種類，ホモフィリックグラフとヘテロフィリックグラフを定義する．枝ホモフィリー率 [11] は，$H(\mathcal{G}) = \frac{\sum_{0 \leqq v,u<n} \mathbf{S}_{(u,v)} \delta(\mathbf{Y}_v, \mathbf{Y}_u)}{\sum_{0 \leqq v,u<n} \mathbf{S}_{(u,v)}}$ で計算される．$\delta(\mathbf{Y}_v, \mathbf{Y}_u)$ は $\mathbf{Y}_v = \mathbf{Y}_u$ の場合，1 を返し，それ以外は 0 を返す関数である．直感的には，枝でつながってる節点が同じラベルをもつ場合に枝ホモフィリー率は増加する $H(\mathcal{G})$ が大きいグラフはホモフィリックグラフ，小さいグラフはヘテロフィリックグラフとなる．ホモフィリックグラフを想定した GNN はヘテロフィリックグラフにおいて有効ではないことが知られている．

2.2 ニューラル構造探索

ニューラル構造探索 (Neural Architecture Search, NAS) は与えられたデータセットに対して最も高精度を達成する深層学習モデルを見つけることを目的とする．モデルは，モデル構造 α とパラメータ $\mathbf{W}$ のペア，$(\alpha, \mathbf{W})$ で表現される．探索空間 $\mathcal{A}$ はモデル構造のパターンを定義する．ベストモデル α^* は下記で定義される．

$$\alpha^* = \arg\max_{\alpha \in \mathcal{A}} \mathrm{E}_{\mathrm{Val}}(\alpha, \mathbf{W}^*_\alpha). \quad (1)$$

$$\mathbf{W}^*_\alpha = \arg\min_{\mathbf{W}} \mathrm{L}_{\mathrm{Train}}(\alpha, \mathbf{W}). \quad (2)$$

1：https://github.com/OnizukaLab/AutoGNN_mcts.

$\mathrm{E}_{\mathrm{Val}}$ および $\mathrm{L}_{\mathrm{Train}}$ は開発データにおける評価指標の値と訓練データにおけるロス関数をそれぞれ表す．

本研究では，既存研究にならいノード分類タスクに取り組む (例えば，[7,8,12–17]). ノード分類タスクは，一部の節点にラベル付けされたグラフが与えられて，グラフ内の他のラベルを予測するタスクである．

3 関連研究

グラフニューラル構造探索. グラフニューラル構造探索手法は，与えられたグラフに対して最適な構造を見つけるために，探索空間と探索アルゴリズムの両面から研究されている [6–10,12–27]. 既存のグラフニューラル構造探索手法の探索空間と探索アルゴリズムを紹介する．

探索空間. 探索空間は二つのパート，構造空間とハイパーパラメータ空間に大別される．構造空間は GNN 構造のパターンを表し，例えば，統合関数や，活性化関数，層数などが含まれ，ハイパーパラメータ空間は，学習率やウェイトディケイなどのハイパーパラメータの値が含まれている．グラフニューラル構造探索手法では，固定のハイパーパラメータで構造空間のみが含まれる探索空間 [7,8,10,12–17] と構造とハイパーパラメータ空間の両方が含まれている探索空間 [6,9,12,18–20,22] が提案されている．どちらの探索空間がよいかは自明ではなく，例えば，ハイパーパラメータを探索した方がよいモデルが見つかる可能性もあるが，ハイパーパラメータを固定した方がより多くの GNN 構造を探索できるためよいモデルが見つかる可能性もある．本研究では，固定のハイパーパラメータで構造空間のみが含まれる探索空間に着目する．これは，グラフの種類に合わせてどのように GNN 構造が異なるかの分析することも目的の一つであるためである．一方で，提案手法はハイパーパラメータ空間も探索アルゴリズムの変更なし組み込むことができる．

構造空間は既存研究により異なるため，この構造空間の設計も技術的な貢献の一つである．探索アルゴリズムを提案していない論文もあることを明記しておく [9,27]. 多くの研究では，基礎的なコンポーネントを組合せて GNN 層を生成し，GNN 層をスキップコネクションなどを用いてつなげることで GNN 構造を構築する [9,13,14,18]. GraphNAS [6] は活性化関数や，アテンション関数，ヘッドの数などの関数を探索するが，層数は固定であるため，柔軟性が低い．GraphGym [9] は GNN 層の前後に MLP を含み，層数やスキップコネクションも構造空間に含まれている．AutoGraph [14] は，スキップコネクションに特化した探索空間であり，層数も自動で決定される．GAUSS [13] は，大規模グラフに対応するためにサンプリングに着目している．AutoGT [27] は，知識グラフにおけるグラフ Transformer に着目した構造空間である．GAP [12], DFG-NAS [15], PDNAS [17], NAS-Bench-Graph [21], および Auto-HeG [10] は，より最新の GNN 層をどのような任意の組合せで GNN 構造を構築するかに着目している．上記において，ヘテロフィリックグラフに着目したグラフニューラル構造探索手法は Auto-HeG [10] のみであり，様々なグラフへの対応性は未だ深く調査されていない．

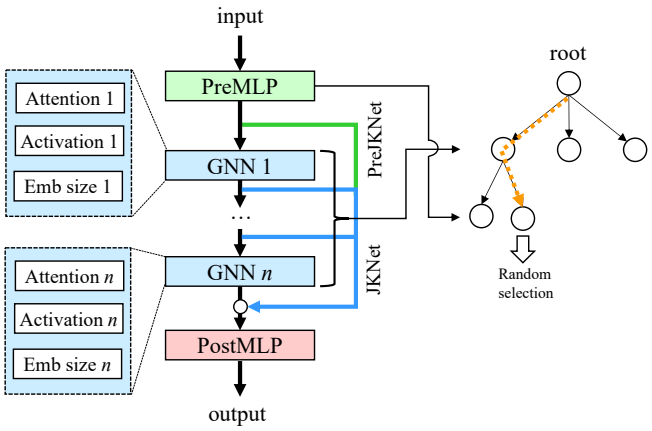


図 1: 探索空間とモンテカルロ木探索

探索アルゴリズム. 探索アルゴリズムは探索空間から効率的に最適な GNN モデルを見つけることを目的としている．大きく分けて，強化学習 [6–8, 28], 進化アルゴリズム [8, 14, 22, 27], および微分可能探索 [10, 12, 17–20, 25] の 3 つに大別できる．モンテカルロ木探索は強化学習に分類することができる．既存の強化学習ベースの探索アルゴリズムは GNN モデルの予想される性能を最大化するようにニューラルモデルが活用されている．進化アルゴリズムベースの探索アルゴリズムは，探索済みのモデルの多くを引き継ぎ，一部を探索したモデルの探索を繰り返す．微分可能探索ベースの探索アルゴリズムは，探索空間を学習パラメータとして，徐々に最適なモデルに近づくようにモデル構造が選ばれている．EGNAS [28] はモンテカルロ木探索と深層強化学習を用いた探索アルゴリズムを採用している．この手法はモンテカルロ木探索により探索空間を分割し，その後深層強化学習により構造を探索している．

4 提案手法

本章では，グラフニューラル構造探索手法を説明する．提案手法は，ホモフィリックグラフとヘテロフィリックグラフの両方に対応可能かつ高速な探索が可能である．以下では,, シンプルかつ効果的な探索空間と探索アルゴリズムについて紹介する．

4.1 探索空間

探索空間は GNN 構造のパターンを定義する．もし探索空間の設計が稚拙な場合，多様なグラフに対する効果的な GNN 構造は含まれないため，探索空間は極めて重要である．我々の探索空間は Micro 構造と Macro 構造の二つの観点が含まれている．Micro 構造は独立した GNN 層の内部を決定し，Macro 構造は GNN 層間の関係性を決定する．図 1 の左図が GNN 構造の探索空間を表す．直感的には， GNN 1 – GNN n が Micro 構造を示し，全体の構造が Macro 構造を示す．

設計方針. 最新の複雑な技術ではなく，基礎的なコンポーネントのみが含まれる探索空間を設計する．これは，複雑ではない GNN 構造のみが候補となるため，与えられたグラフに対してどのようなコンポーネントが重要なのかを理解することに役立つ．ヘテロフィリックグラフに対応するために，有効と知られている (i) 自身の属性の強調および (ii) Jumping knowledge が含まれている [11, 29–32].

探索空間は構造パラメータのセットとして表現される．スーパーネットが NAS では頻繁に用いられるが，グラフニューラル構造探索では一般的ではない．構造パラメータのセットは，重要なパラメータの分析に効果的である．これは，生成された GNN 構造間の比較が容易であり，一方でスーパーネットは複雑な構造が生成されるため比較が容易ではないためである．

Micro 構造. 多層のメッセージパッシングモデルは標準的な GNN 構造である．このモデルはノード特徴量を層毎に畳み込みをしながら学習する．正式には，l 番目の GNN 層は下記で定義される．

$$\mathbf{z}_u^{(l)} = \sigma\left(\sum_{v\in\mathcal{N}_u} e_{(u,v)}\mathbf{W}^{(l)}\mathbf{z}_v^{(l-1)}\right). \tag{3}$$

$\mathbf{z}_u^{(l)}$ は l 層目の節点 u のエンベッディング，$\mathbf{W}^{(l)} \in R^{|\mathbf{z}^{(l)}|\times|\mathbf{z}^{(l-1)}|}$ は学習パラメータ，および $\mathcal{N}_u$ は節点 u の隣接節点を表す．ここでは，$\mathbf{z}_u^{(0)}$ は節点 u の属性である．$e_{(u,v)}$ と σ は節点 u と v 間のアテンションと活性化関数をそれぞれ表す．

Micro 構造はアテンション，活性化関数，エンベッディングサイズの 3 つの設計次元がある，これらは GNN 層における標準的なコンポーネントである．モデル特有のコンポーネント，例えば GAT におけるヘッド数など，は GNN 構造のパターン数削減のために含まれていない．

Macro 構造. Macro 構造は GNN 層のつながりを全体のグラフ深層学習としてどのように構成するかを決定する．最も簡単な方法は GNN 層を単に重ねる場合となる．提案手法はそれぞれの GNN 層の出力をつなげる jumping knowledge network (*JKNet*) を用いる．また，GNN 層の前後に MLP 層を用いる．我々はそれぞれ *preMLP* と *postMLP* と呼ぶ．加えて，preMLP の出力を最終層の出力の入力とする．*PreJKNet* と呼ぶ．

この Macro 構造の利点はヘテロフィリックグラフに対して高精度なモデルの構築が期待できる点である．ヘテロフィリックグラフにおいて，自身の節点属性は隣接節点からの特徴量の統合に比べて効果的であることが多い [9, 11, 33–36]．PreMLP と PreJKNet の組合せはヘテロフィリックグラフにおいて効果的であるが，既存のグラフニューラル構造探索ではまだ用いられていない．

構造パラメータ. 探索空間の構造パラメータについて述べる．表 1 に構造パラメータをまとめる．提案手法はそれぞれの GNN 層に異なる構造パラメータを選択できる．例えば，層数が 2 の場合，第一層のアテンションが GCN，第二層のアテンションが GAT のような構造が可能である．この構造パラメータでは，探索空間には 2 億以上の構造パターンが含まれている．ここで，構造パラメータには依存関係がある．例えば，JKNet が Max の場合，全てのエンベッディングサイズが同じ必要がある．エンベッディングサイズはグラフサイズと GPU メモリに合わせて設定することができる．

表 1: 構造パラメータ．d_u と || はそれぞれ節点 u の次数と連結関数を示す．

Function	Parameters
The number of GNN layers	1, 2, 3
Attention	Constant: $e_{(u,v)} = 1$ GCN: $e_{(u,v)} = 1/\sqrt{d_u d_v}$ GAT: $e_{(u,v)} = leakyReLU(\mathbf{W}_l \mathbf{z}_u \| \mathbf{W}_r \mathbf{z}_v)$
Activation	None, Relu, Sigmoid, Tanh
The embedding size of of GNN layer	16, 32, 64, 128, 256, y
JKNet	None, concat, max
PreJKNet	None, use
PreMLP	None, use
The embedding size of preMLP	16, 32, 64, 128, 256
The number of layers in postMLP	0, 1, 2
The hidden size of postMLP	64, 128, 256

4.2 探索アルゴリズム

提案手法における探索アルゴリズムの特徴はモンテカルロ木探索を利用し，ニューラルモデルを利用しないことである．これにより，高速かつスケーラブルな探索可能となる．このアプローチはシンプルである一方で，GNN 構造の決定に有効である．

図 1 の右図はモンテカルロ木探索による構造パラメータの決定方法を表している．モンテカルロ木を辿ることで，構造パラメータを決定する．その際，それぞれの木の深さにコンポーネントを割り当て，モンテカルロ木の節点に構造パラメータを割り当てる．例えば，この図では層数が深さ 1 に割り当てられている．

設計方針．探索アルゴリズムでは，微分可能アルゴリズム (例，[37]) や深層強化学習 (例，[6]) などのニューラルモデルが頻繁に用いられる．このようなニューラルモデルを用いるアプローチは高精度な GNN 構造を選択するのに有効であることが多い，一方で学習コストが高く，非効率であることが多い．そこで，ニューラルモデルを用いないモンテカルロ木探索を用いることで，高速かつスケーラブルな探索を可能とする．加えて，構造パラメータ選択やそのパラメータを用いた場合の平均精度がわかるため，モデルの分析にも有効である．

構造選択. 下記の手順により GNN 構造を選択する．(1) 葉ノードを選択する，(2) ルートから葉ノードの経路上にある節点に割り当てられている構造パラメータを決定する，そして (3) それ以外の構造パラメータをランダムに決定する．

葉ノードの決定において，UCB (Upper Confidence Bound) [38] を拡張したスコアを節点毎に与える．節点 i の ucb スコアは下記で計算される．

$$ucb(i) = \frac{\sum_{(\alpha,\mathbf{W})\in\mathcal{M}_i} \mathrm{E}_{\mathrm{Val}}(\alpha,\mathbf{W})}{m_i} + c\sqrt{\frac{\ln M}{m_i}} \qquad (4)$$

$\mathcal{M}_i$ は評価済みのモデル，m_i は i が利用された回数，および M は探索数である．c は探索と活用のバランスを調整するハイパーパラメータであり，c が大きいほどより未知のものを探索する．m_i が増加すると，$usb(i)$ が低下するため，モンテカルロ木探索は局所探索に陥ることなく探索を行える．

木構造の更新. モデル構造を選択した後は，選択されたモデル構造の性能を確認するために学習と検証が行われる．その後，ルートから葉ノードの経路上にある節点の ucb を更新する．節点 i が θ 以上選ばれたら，新しい子節点を作成する．これらの子ノードは $m = 0$ (つまり, $ucb = \inf$) であるため，次の探索時には優先的に選択される

この木構造には平均性能や，モデルが選択された回数，平均学習時間も格納するため，その構造パラメータが精度と効率性に与える影響も調査できる．

ハイパーパラメータ. この探索アルゴリズムには c と θ という二つのハイパーパラメータがある．既存研究にならい，$c = \sqrt{2}$ とする [38]．加えて，これらのハイパーパラメータの影響が大きくないことを本実験にて示す．そのため，提案手法はハイパーパラメータチューニングの必要はない．

コンポーネント順. 高精度な GNN 構造を効果的に発見するためにモンテカルロ木のコンポーネント順を決定する．小さい木の深さに割り当てられたコンポーネントは早めに決定されるため，その構造パラメータの精度への影響が早めに精査される．そのため，重要なコンポーネントを小さい木の深さに割り当てる必要がある．

提案手法のモンテカルロ木では，GNN の層数, preMLP の有無, preJKNet の有無, および JKNet の種類の順に割り当てる．これらは，性能に大きく影響することが知られている [11,32–36]. その後，最初の GNN 層の活性関数とアテンション関数，PreMLP のエンベッディングサイズ，PostMLP の中間層の数，最初の GNN 層のエンベッディングサイズを決定する．最後に，性能の影響を予測するのが難しいその他のコンポーネント，つまり 2 層目以降の活性化関数，アテンション関数，エンベッディングサイズを決定する．

時間計算量. GNN 構造の選択では，モンテカルロ木を辿り，その後ランダムに構造のパラメータを決定する．この処理は，$O(|\mathbb{F}||\mathbb{A}|)$ となる．ここで，$|\mathbb{F}|$ と $|\mathbb{A}|$ はそれぞれコンポーネントと構造パラメータの数であるが，これらは定数と考えることができるため $O(1)$ である．木構造の更新においても，モンテカルロ木を辿るだけであるため $O(1)$ である．そのため，GNN 構造を決定するための探索アルゴリズムは $O(1)$ となる．

疑似コード. アルゴリズム 1 に提案手法の疑似コードを示す．このアルゴリズムは GNN 構造 α の選択と訓練を繰り返す（3–6 行目）．モデルの訓練が完了後に，モンテカルロ木を更新する（7–9 行目）．L 回のモデル探索を繰り返す（2 行目）．

5 評 価 実 験

本章では，ノード分類タスクにおける提案手法の有効性を検証する．本実験では，提案手法の精度と効率性を評価し，発見した GNN 構造の分析が設計に役立つかを示す．

提案手法は Python3 で実装し，AWS の p3.x2large インスタンス (NVIDIA V100 Tensor Core GPU および 16 GB の GPU メモリを搭載) を用いる．

Algorithm 1: Ours

input : $\mathcal{G}$, L
output: α^*, $\mathbf{W}^*$, MCT

Initialize MCT, $best$;
for $1, \ldots, L$ **do**
  $i \leftarrow$ leaf node with the maximum ucs;
  $\alpha \leftarrow$ functions of n_i with random parameters;
  Initialize $\mathbf{W}$ of α;
  Train model $(\alpha, \mathbf{W})$ on $\mathcal{G}$;
  Update MCT;
  if $m_i \geqq \theta$ **then**
    Generate child nodes of i;
  if $best < \mathrm{E}_{\mathrm{VAL}}(\alpha, \mathbf{W})$ **then**
    $\alpha^*, \mathbf{W}^* \leftarrow \alpha, \mathbf{W}$;
    $best \leftarrow \mathrm{E}_{\mathrm{VAL}}(\alpha, \mathbf{W})$;
return α^*, $\mathbf{W}^*$, MCT;
end procedure

5.1 実験環境

データセット. GNN のタスクでよく用いられている 12 個のグラフを用いる．これらのグラフは，様々な応用から作成され，異なる枝ホモフィリー率を有する．表 2 にグラフの統計を示す．

表 2: データセットの概要

Dataset	# nodes	# edges	# features	# labels	$H(\mathcal{G})$
Cora	2,708	5,429	1,433	7	0.81
CiteSeer	3,327	4,732	3,703	6	0.73
Amz-P	7,650	238,162	745	8	0.82
Amz-C	13,752	491,722	767	10	0.77
Co-CS	18,333	163,788	6,805	15	0.79
PubMed	19,717	44,338	500	3	0.81
Cornell	195	304	1,703	5	0.13
Wisconsin	265	530	1,703	5	0.20
Chameleon	2,277	36,101	2,325	5	0.23
Squirrel	5,201	217,073	2,089	5	0.23
Actor	7,600	30,019	932	5	0.23
Penn94	38,815	2,498,498	4,772	2	0.53

比較手法. コードが公開されている 4 つのグラフニューラル構造探索 GraphNAS [6], GraphGym [9], DFG-NAS [15], および Auto-HeG [10] を比較手法として用いる. GraphNAS, DFG-NAS, および Auto-HeG はそれぞれ深層強化学習，進化アルゴリズム，および微分可能探索を探索アルゴリズムに用いている．GraphGym は探索アルゴリズムが提案されていないため，既存手法に合わせて一様サンプリングを探索アルゴリズムとして用いる [15].

ハイパーパラメータ. ニューラル構造探索のモデル探索数は 1000 とする [39]. 提案手法における θ と c は，それぞれ 10 および $\sqrt{2}$ とする．GraphNAS, GraphGym, DFG-NAS, および Auto-HeG においては，GitHub に記載されているデフォルトパラメータを用いる．また，公平な評価のためにハイパーパラメータチューニングは行わないが，GraphGym と Auto-HeG は探索空間にハイパーパラメータが含まれている．訓練，開発，および検証データの割合は，それぞれ 0.6/0.2/0.2 とする．

5.2 性能評価

精度比較. 表 3 および表 4 にそれぞれ AUC と正解率の実験結果を示す．各手法において正解率と AUC の最大化のために異なる GNN モデルを構築している．

提案手法はヘテロフィリックグラフとホモフィリックグラフの両方で高い平均正解率と AUC を達成している．提案手法は各グラフで頻繁に最高性能ではないが，最低性能の場合であっても，他の手法との差は非常に小さい．ヘテロフィリックグラフにおける GraphNAS とホモフィリックグラフにおける Auto-HeG を除く既存のグラフニューラル構造探索手法は平均的に高い性能を達成している．ホモフィリックグラフとヘテロフィリックグラフ間の GraphNAS の性能差は，ホモフィリックグラフ用に設計された探索空間がヘテロフィリックグラフには適していない可能性を示している．GraphGym と DFG-NAS は一部のグラフにおいて最高性能を達成している一方で，ヘテロフィリックグラフのいくつかに対しては低い性能となっている，例えば，GraphGym の Chameleon と Squirrel や DFG-NAS の Penn94 である．Auto-Heg は主にヘテロフィリックグラフに焦点を当てているため，ホモフィリックグラフに対するは性能が低い．また，Auto-Heg は多数の枝を持つグラフにおいて複雑な GNN 構造を構築するためにスケーラブルではないことがわかる．

結果として，提案手法は 12 のグラフの中で最も高い平均 AUC と正解率を達成しているため，提案手法がホモフィリックグラフとヘテロフィリックグラフの両方に高い適応性を持っていることが確認できる．

探索時間の比較. 図 2 にグラフ構造探索の実行時間を示す．この探索時間にはモデルの訓練時間も含まれている．実験結果より Ours, GraphGym, および DFG-NAS が効率的であることがわかる．これらの手法はニューラルモデルを構造選択に利用していない．提案手法はヘテロフィリックグラフで高速，一方 GraphGym と DFG-NAS はホモフィリックグラフで高速であることが多い．GraphNAS と Auto-HeG は探索アルゴリズムに訓練時間が含まれるため実行時間が低速である．この結果より，提案手法は高速であることを確認できる．

モデルサイズ比較. 表 5 に出力したモデルのパラメータ数を示す．提案手法が出力したモデルはモデルサイズが小さいことがわかる．特に，ヘテロフィリックグラフにおいて全てのグラフで最小のモデルを出力している．この結果より，提案手法のモデルは高精度かつ小モデルであることがわかる．

5.3 ハイパーパラメータセンシティビティ

提案手法は c と θ の二つのハイパーパラメータがある．図 3 はそれぞれのハイパーパラメータの AUC への影響を示している．これらの実験結果より，c と θ は AUC に大きく影響しないことがわかる．そのため，提案手法はハイパーパラメータチューニングを実施することなくデフォルトパラメータで十分であることを示しており，計算コストの低下と使いやすさが向上しているといえる．

表 3: AUC の比較．青と赤のハイライトは各列におけるそれぞれ最大と最小を表す．DNF は 48 時間以内で実行できなかったもの，OOM は ‘Out of Memory.’ を表す．

	Heterophilic							Homophilic						
	Cornell	**Wisconsin**	**Chameleon**	**Squirrel**	**Actor**	**Penn94**	Average	**Cora**	**CiteSeer**	**Amz-P**	**Amz-C**	**Co-CS**	**PubMed**	Average
GraphNAS	69.2±0.3	74.7±0.2	57.5±0.1	36.7±0.7	33.2±0.0	DNF	54.3	87.1±0.0	76.0±0.0	95.8±0.0	**91.9**±0.0	94.2±0.0	87.8±0.0	88.8
GraphGym	74.9±5.9	87.2±6.3	54.0±2.0	37.2±1.3	**38.3**±1.0	**85.6**±0.4	62.9	82.4±1.6	74.2±1.5	**96.2**±0.3	**91.9**±0.3	95.3±0.4	**89.7**±0.7	88.3
DFG-NAS	**83.1**±0.4	**91.7**±0.1	67.0±0.1	47.2±0.0	38.1±0.0	76.1±0.0	67.2	**88.5**±0.0	**77.1**±0.0	95.2±0.0	88.0±0.0	**95.8**±0.0	87.6±0.0	88.7
Auto-HeG	74.9±14.2	86.8±39.9	OOM	OOM	38.0±1.0	OOM	66.6	75.5±19.9	70.5±3.5	OOM	OOM	OOM	88.3±0.1	78.1
Ours	72.3±0.3	84.2±0.1	**69.9**±0.0	**59.5**±0.0	37.9±0.0	82.5±0.0	**67.7**	**88.5**±0.0	74.7±0.0	95.6±0.0	91.2±0.0	95.6±0.0	89.3±0.0	**89.2**

表 4: 正解率の比較．青と赤のハイライトは各列におけるそれぞれ最大と最小を表す．DNF は 48 時間以内で実行できなかったもの，OOM は ‘Out of Memory.’ を表す．

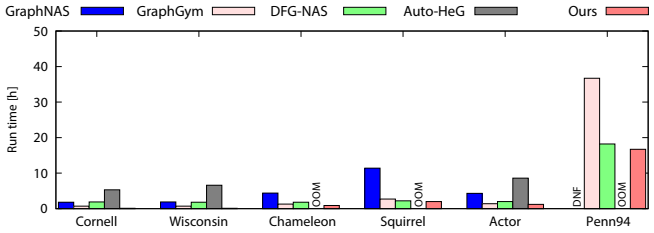
	Heterophilic							Homophilic						
	Cornell	**Wisconsin**	**Chameleon**	**Squirrel**	**Actor**	**Penn94**	Average	**Cora**	**CiteSeer**	**Amz-P**	**Amz-C**	**Co-CS**	**PubMed**	Average
GraphNAS	86.1±0.2	89.8±0.1	83.9±0.0	73.8±0.0	65.1±0.0	DNF	79.7	97.2±0.0	92.1±0.0	99.5±0.0	**99.2**±0.0	99.7±0.0	97.0±0.0	97.5
GraphGym	93.0±2.1	95.4±3.8	83.4±1.5	71.4±1.0	**72.4**±0.6	**93.4**±0.2	84.8	96.7±0.5	92.6±0.6	**99.6**±0.2	**99.2**±0.1	99.8±0.0	**97.5**±0.3	97.6
DFG-NAS	**94.5**±0.0	**95.8**±0.0	87.3±0.0	77.2±0.0	70.0±0.0	84.8±0.0	84.9	**98.2**±0.0	**93.6**±0.0	99.4±0.0	98.9±0.0	**99.9**±0.0	96.8±0.0	**97.8**
Auto-HeG	89.3±0.3	**95.8**±0.1	OOM	OOM	70.0±0.0	OOM	85.0	92.9±0.0	91.0±0.0	OOM	OOM	OOM	96.8±0.0	93.6
Ours	91.0±0.1	93.6±0.2	**87.6**±0.0	**81.4**±0.0	70.1±0.0	91.9±0.0	**85.9**	98.0±0.0	93.4±0.0	99.1±0.0	98.8±0.0	99.7±0.0	97.3±0.0	97.7

表 5: モデルサイズ．青のハイライトは各列におけるそれぞれ最小値を表す．

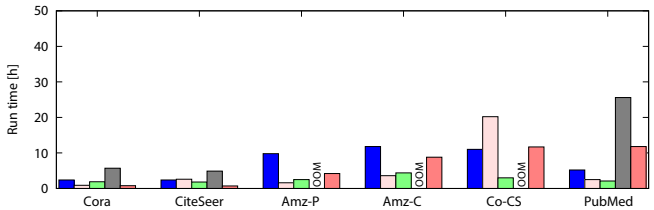
	Heterophilic						Homophilic					
	Cornell	**Wisconsin**	**Chameleon**	**Squirrel**	**Actor**	**Penn94**	**Cora**	**CiteSeer**	**Amz-P**	**Amz-C**	**Co-CS**	**PubMed**
GraphNAS	1288.4K	2214.7K	1806.0K	1584.9K	1470.4K	OOM	1590.7K	1213.1K	224.6K	239.7K	1346.3K	320.1K
GraphGym	491.0K	491.0K	572.4K	541.5K	390.0K	891.9K	456.1K	753.2K	366.3K	369.7K	1161.6K	332.9K
DFG-NAS	430.1K	459.9K	314.9K	284.7K	272.0K	611.2K	191.1K	1699.8K	96.5K	99.6K	873.1K	94.2K
Auto-HeG	357.7K	442.5K	OOM	OOM	529.3K	OOM	667.3K	930.2K	OOM	OOM	OOM	69.7K
Ours	136.5K	377.0K	100.5K	241.6K	53.3K	102.2K	317.1K	545.6K	143.5K	111.0K	1812.8K	69.6K

表 6: アブレーションスタディ．

	GraphGym	**Uniform**	**Max**	**Ours**
Homophilic	97.8	97.3	97.3	97.7
Heterophilic	84.9	85.5	85.4	85.9



(a) ヘテロフィリックグラフ



(b) ホモフィリックグラフ

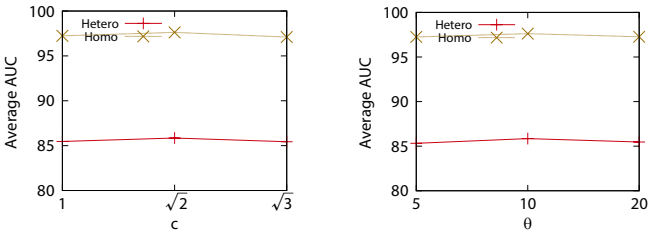
図 2: 実行時間 [h].

5.4 アブレーションスタディ

表 6 は GraphGym と提案手法に加えて, 提案手法の探索アルゴリズムを一様サンプルに変更した手法 (Uniform) と最大性能であるモデル構造を優先的に探索した手法 (Max) の実験結果を示す. GraphGym と Uniform は同じ探索アルゴリズムを用いているため探索空間を比較することができ，Uniform，Max，および提案手法を比べることで探索アルゴリズム間の比較ができる．まず，Uniform はヘテロフィリックグラフにおいて GraphGym より性能が高く，ホモフィリックグラフにおいて GraphGym より性能が低い．これは提案手法の探索空間がヘテロフィリックグラフにより適していることを示している．次に，提案手法は Uniform と Max よりも性能が良いため，モンテカルロ木探索が優先的に良いモデルを探索できていることがわかる．

5.5 GNN 構造分析

提案手法が GNN 構造の分析に効果的であることを示す．ここでは，特にホモフィリックグラフとヘテロフィリックグラフの違いについて着目する．既存のグラフニューラル構造探索では，コンポーネントが複雑であることや両方のグラフタイプにて高精度な構造を発見できないため，構造の分析では効果的ではない．



(a) c (b) θ

図 3: ハイパーパラメータ c と θ の影響

選択された GNN 構造の例. 図 4 に Citeseer, Amz-c, Cornell, および Squirrel で選択されて GNN 構造を示す．これらの例から選択された GNN 構造が大きく異なることがわかる．まず，ホモフィリックグラフにおける GNN 構造（Citeseer と Amz-c）は単純であり，ヘテロフィリックグラフにおける GNN 構造（Cornell と Squirrel）は複雑であることがわかる．特に，Citeseer における GNN 構造は極めて単純であり，ノードラベルが隣接節点情報から推測できることがわかる．一方で，Cornell と Squirrel では，より広範囲な節点情報と自身の属性情報が重要であることがわかる．これらの例から，ヘテロフィリックグラフとホモフィリックグラフでは最適なモデル構造が大きく異なることがわかる．

選択された構造パラメータ分析. 表 7 に本実験にて選択された構造パラメータの割合を示す．まず，ヘテロフィリックグラフではより多くの層を用いることがわかる．GNN 構造の層数は一般的に 2 層であるが，ヘテロフィリックグラフでは最適ではないことが多いことがわかる．次に，ハイパボリックタンジェントが活性化関数で最も多く選ばれている．一方で，GNN では ReLU を活性化関数として用いることが一般的であるが，最適ではないことが多いことを示している．三番目に，JKNet はヘテロフィリックグラフでは効果的であることがわかる．最後に，アテンションはホモフィリックおよびヘテロフィリックグラフともに大きな差がないことがわかる．そのため，constant, GCN，および GAT をグラフタイプに合わせて事前に選択することが難しいことがわかる．これらの洞察は GNN 構造の設計に有益である．

上記の分析にあわせて，GCN の性能を Tanh と ReLU を用いた場合で比較する．標準的な GCN は ReLU を活性化関数として用いており，多くの既存研究でも活性化関数を変えた評価はしていない．表 8 に Tanh と ReLU を用いた場合の GCN の AUC を示す．Cornell, Wisconsin, および Actor では，Tanh を用いた方が AUC が増加し，一方他のグラフでは低下した．Wisconsin では AUC が 6.0 増加している．この実験により，活性化関数はグラフに合わせて選択する必要があることを表している．提案手法は，重要なコンポーネントの分析と既存の GNN 構造の再設計に有効であることがわかる．

6 ま と め

本論文はグラフニューラル構造探索を提案した．評価実験において，提案手法はホモフィリックグラフおよびヘテロフィリックグラフの両方において，効果的かつ効率的に GNN 構造を探索できることを示した．今後は，構造パラメータを自動的に生成する手法およびより効果的な探索空間と探索アルゴリズムを開発する．

謝　　辞

本研究は JST さきがけ JPMJPR21C5 および JSPS 科学研究費 JP20H00583 の支援よって行われた．

文　　献

[1] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015.

[2] Victor Fung, Jiaxin Zhang, Eric Juarez, and Bobby G Sumpter. Benchmarking graph neural networks for materials chemistry. *npj Computational Materials*, Vol. 7, No. 1, pp. 1–8, 2021.

[3] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *ICML*, pp. 8459—-8468, 2020.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[5] Peixiang Zhong, Di Wang, and Chunyan Miao. Eeg-based emotion recognition using regularized graph neural networks. *IEEE Transactions on Affective Computing*, 2020.

[6] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *IJCAI*, pp. 1403–1409, 2020.

[7] Huan Zhao, Lanning Wei, and Quanming Yao. Simplifying architecture search for graph neural network. *arXiv preprint arXiv:2008.11652*, 2020.

[8] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184*, 2019.

[9] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *NeurIPS*, 2020.

[10] Xin Zheng, Miao Zhang, Chunyang Chen, Qin Zhang, Chuan Zhou, and Shirui Pan. Auto-heg: Automated graph neural network on heterophilic graphs. *WWW*, pp. 611—-620, 2023.

[11] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *NeurIPS*, 2020.

[12] Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *CVPR*, pp. 6657–6666, 2021.

[13] Chaoyu Guan, Xin Wang, Hong Chen, Ziwei Zhang, and Wenwu Zhu. Large-scale graph neural architecture search. In *ICML*, pp. 7968–7981, 2022.

[14] Yaoman Li and Irwin King. Autograph: Automated graph neural network. In *NeurIPS*, 2020.

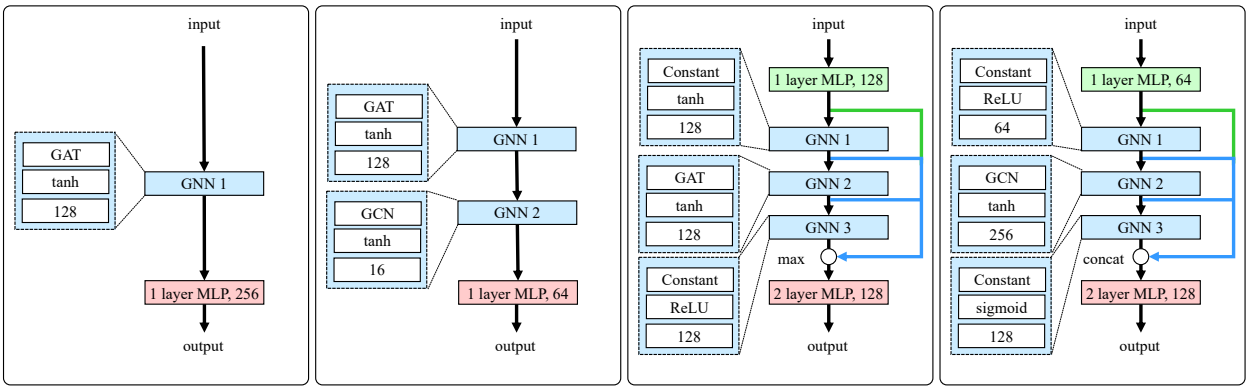
[15] Wentao Zhang, Zheyu Lin, Yu Shen, Yang Li, Zhi Yang, and Bin Cui. Dfg-nas: Ddeep and flexible graph neural architecture search. In *ICML*, pp. 26362–26374, 2022.

[16] Wentao Zhang, Yu Shen, Zheyu Lin, Yang Li, Xiaosen Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. Pasca: A graph neural architecture search system under the scalable paradigm. In *WWW*, pp. 1817–1828, 2022.

[17] Yiren Zhao, Duo Wang, Xitong Gao, Robert Mullins, Pietro Lio, and Mateja Jamnik. Probabilistic dual network architecture search on graphs. *arXiv preprint arXiv:2003.09676*, 2020.

[18] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *KDD*, pp. 279–288, 2021.

[19] ZHAO Huan, YAO Quanming, and TU Weiwei. Search to aggre-



(a) Citeseer (b) Amz-c (c) Cornell (d) Squirrel

図 4: 選択された GNN 構造の例

表 7: 選択された構造パラメータの割合

	層数			活性化関数				JKNet			アテンション		
	1	2	3	none	ReLU	Sigmoid	Tanh	none	concat	max	constant	GCN	GAT
Hom	0.14	0.63	0.23	0.32	0.10	0.04	0.55	0.29	0.54	0.17	0.30	0.48	0.22
Hetero	0	0.47	0.53	0.11	0.21	0.17	0.51	0.03	0.67	0.30	0.25	0.51	0.24

表 8: GCN における活性化関数の影響

	Cornell	Wisconsin	Chameleon	Squirrel	Actor	Penn94
GCN (ReLU)	70.2	67.6	85.7	71.9	56.4	89.3
GCN (Tanh)	71.1	73.5	80.3	69.8	58.1	87.9

gate neighborhood for graph neural network. In *ICDE*, pp. 552–563, 2021.

[20] Yanxi Li, Zean Wen, Yunhe Wang, and Chang Xu. One-shot graph neural architecture search with dynamic search space. In *AAAI*, pp. 8510–8517, 2021.

[21] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. Nas-bench-graph: Benchmarking graph neural architecture search. 2022.

[22] Min Shi, David A Wilson, Xingquan Zhu, Yu Huang, Yuan Zhuang, Jianxun Liu, and Yufei Tang. Evolutionary architecture search for graph neural networks. *arXiv preprint arXiv:2009.10199*, 2020.

[23] Zhen Wang, Zhewei Wei, Yaliang Li, Weirui Kuang, and Bolin Ding. Graph neural networks with node-wise architecture. In *SIGKDD*, pp. 1949–1958, 2022.

[24] Ziwei Zhang, Xin Wang, and Wenwu Zhu. Automated machine learning on graphs: A survey. *arXiv preprint arXiv:2103.00742*, 2021.

[25] Peng Xu, Lin Zhang, Xuanzhou Liu, Jiaqi Sun, Yue Zhao, Haiqin Yang, and Bei Yu. Do not train it: A linear neural architecture search of graph neural networks. In *ICML*, 2023.

[26] Guanghui Zhu, Wenjie Wang, Zhuoer Xu, Feng Cheng, Mengchuan Qiu, Chunfeng Yuan, and Yihua Huang. Psp: Progressive space pruning for efficient graph neural architecture search. In *ICDE*, pp. 2168–2181, 2022.

[27] Zizhao Zhang, Xin Wang, Chaoyu Guan, Ziwei Zhang, Haoyang Li, and Wenwu Zhu. Autogt: Automated graph transformer architecture search. In *ICLR*, 2023.

[28] TianJin Deng and Jia Wu. Efficient graph neural architecture search using monte carlo tree search and prediction network. *Expert Systems with Applications*, Vol. 213, p. 118916, 2023.

[29] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, pp. 21–29, 2019.

[30] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, pp. 1725–1735, 2020.

[31] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *ICLR*, 2021.

[32] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

[33] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? 2021.

[34] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Is heterophily a real nightmare for graph neural networks to do node classification? 2022.

[35] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *ICML*, pp. 13242–13256, 2022.

[36] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *NeurIPS*, 2021.

[37] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. 2019.

[38] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML*, pp. 282–293, 2006.

[39] Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca. Neural architecture search using deep neural networks and monte carlo tree search. In *AAAI*, pp. 9983–9991, 2020.