The 32nd Annual Conference of the Japanese Society for Artificial Intelligence, 2018

半教師有りグラフニューラルネットワークを用いたCRUD関係に基づくシステム移行単位の最適化

Optimization for the Migration Units in Information Systems based on CRUD Relations with Semi-Supervised Graph Neural Networks

貞光 九月 *1
Kugatsu Sadamitsu

*1フューチャー株式会社 Strategic AI Group
SAIG, Future Corporation

Migrating large-scale information system as a whole is impossible due to its size, thus it is necessary to divide the system into manageable migration units. Typically, optimizing the migration units can only be done by experienced engineers, and takes large amount of time. In this paper, we propose a method finding optimal migration units. To find optimal migration units, we interpret CRUD relations between functions and tables as a graph structure, and adapt semi-supervised graph neural network, Diffusion Convolutional Neural Networks, with a graph embedding method. Furthermore, we unify CRUD relation with file arrangement information and attributes of each node. In experiments, the proposed methods find practical migration units that can be applied to real examples.

1. はじめに

産業の基幹となる大規模な情報システムにおいて，外部環境変化への対応や業務の最適化等の目的から，システム刷新の要求は必然的に生じる [木原他 2015]．大規模な情報システムの刷新を行う際，一度にすべての機能を新システムに切り替えることはリスクが高く非現実的であり，実際の開発では，一部の機能やテーブルの集合を部分的に移行単位として切り出し，段階的に移行するという安全策が採用されることが多い．この時の移行単位の設計は，業務とプログラムに精通する技術者が自身の経験則に基づいて決定することが多く，設計自体に多大な工数を要するという課題や，対象システムに対する知識と経験を持つ一部の技術者にスキルが偏在してしまうという属人性の課題があった．我々はこれらの課題を解決するために，大規模システムの段階移行開発における，移行単位の自動最適化手法を提案する．

本稿では，移行単位最適化の手がかりとして，システム設計時に用いられる，機能とテーブルの操作関係を記述したCRUD表に着目する．CRUDとは，テーブルに対する代表的な操作である，Create, Read, Update, Deleteの頭文字をとったものであり，機能からテーブルへの操作がどのように行われるか記述したものである．CRUD表は人手による移行単位設計の際にも参照されることが多く，自動化においても有益なリソースとして活用できることが見込まれる．また，2. 節で述べる通り，CRUD表は一種のグラフ構造とみなすことができ，そのためシステム移行単位の最適化問題は，CRUD関係を表すグラフにおける各機能ノード及び各テーブルノードの分類問題と定義できる．さらに本稿ではCRUD関係に加え，機能のソースコードやテーブルデータのコンピュータ内のディレクトリ配置情報や，機能やテーブル自体が持つ属性情報を含めた統一的な1つのグラフとして取り扱うことで，より高精度な解析を可能とする．

グラフ中のノード分類手法はこれまでに多くの手法が提案されており，本稿では半教師有りグラフノード分類手法の一種で，近年高い精度が報告されているグラフニューラルネットワークの一種，Diffusion Convolutional Neural Network [Atwood and Towsley 2016] を用い，さらにグラフ埋め込み (graph embedding) 法の一種であるnode2vec [Grover and Leskovec 2016] と Poincaré embedding [Nickel and Kiela 2017] をそれぞれ併用することで，比較的少量の教師データのみを用いた場合でも，高い精度で移行単位の推定が可能となることを示す．

連絡先: 貞光九月，フューチャー株式会社，東京都品川区大崎 1 － 2 － 2 アートヴィレッジ大崎セントラルタワー，k.sadamitsu.ic@future.co.jp

2. 移行単位最適化のためのCRUD関係の利用

2.1 CRUD関係とCRUD表

本節では，移行単位の最適化に必要な手がかりとして，CRUD関係およびCRUD表を用いることを考える．システム開発において，機能とテーブルの関係は極めて重要である．システムの設計や開発の各フェーズにおいて，機能とテーブルの操作関係を漏れなく設計・開発に落とし込むために広く用いられるものがCRUD表である．図1 (a) にCRUD表の例を，図1 (b) にCRUD表に対応するシステム中のCRUD関係を表す．例えば，図1(a)において，機能ノード “Func2” は，テーブルノード “Table2” に対し “Create” の関係を持ち，“Table3” に対し “Read” の関係を持つことを表す．CRUD関係を持つノード同士は比較的強い結びつきを持つと考えられることから，本稿ではこのCRUD表に着目した移行単位の最適化を行う．

本稿で扱うタスク概要についても同図を用いて表す．まず図1 (a) 中，“Unit class1”，“Unit class2” と記した囲いが，人手で解析した移行単位の理想的な分類結果であるとする．本稿で扱う移行単位最適化は，一部の機能またはテーブルに対応する移行単位クラスを教師信号として与え (図 (b) 中 “seed” と記した “Func2” と “Func5”)，その他のノードの移行単位クラスを正しく分類することが目的となる．

さらにCRUD表は，CRUD関係をグラフとみなした場合の隣接行列とみなすことが可能であり，実際に，図1 (b) のCRUD関係に対応するグラフ構造を，図1(a)のCRUD表という隣接行列に等価に変換することができる．システム操作関係をグラフとして解釈することにより，3. 節で述べるように，グラフ解析のための機械学習を直接的に適用できるという利点が生じる．

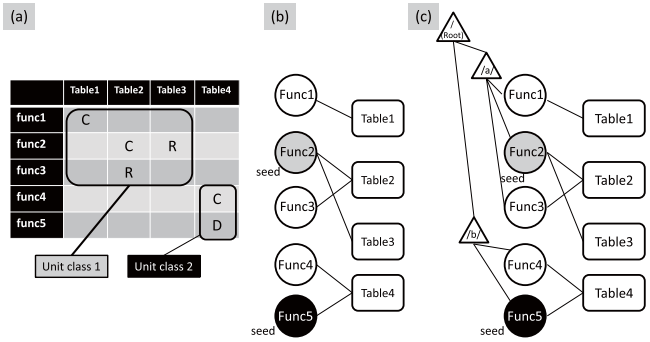

図 1: (a)CRUD 表と人手付与した正解移行単位 (Unit class 1, 2)，(b) CRUD 表に対応する機能とテーブル間の関係図．“seed” と記す “Func2” と “Func5” が教師データで，残る全ノードがクラス推定対象ノード．(c) 機能とテーブル間の関係図に，ファイル配置情報を重畳したもの．Func1 と Func2 が同じクラスに属す可能性を示している．

2.2 ファイル配置情報とノード属性の利用

CRUD 関係は機能とテーブルの関係を表現する上で重要な手がかりとなりえるが，本節では CRUD 関係以外で追加できる手がかり情報を検討する．

1 つ目の手がかりは，各機能のソースコードやテーブルデータのコンピュータ上でのファイル配置情報である．システム開発を行う上で，同じディレクトリに配置される機能やテーブルは，1 つの単位としてまとめて移行する方が適切であるという仮説に基づいて導入する．ファイル配置情報の利用方法にはいくつか方策が考えられるが，本稿では直接 CRUD 関係のグラフ構造に重畳する方法を提案する．具体的には，元となる CRUD 関係を表すグラフ構造に対し，OS のディレクトリに相当する仮想ノードを設け，そのディレクトリ内に配置される機能のソースコード・テーブルデータ，あるいは親・子ディレクトリに対し，CRUD 関係と同様にエッジを張ることとする．実際に重畳した具体例を図 1 の (c) に示す．このような重畳を行うことで，図 1 の (b) における “Func1” と “Table1” のように他ノードから孤立したサブグラフが，(c) ではファイル配置情報を考慮することで “Func2” 等と同じディレクトリ “/a/” 以下の配置であることが分かり，“Func2” と同じ移行単位とすべきと判断する根拠とすることができる．なおグラフ中のエッジを定義する際に，CRUD 関係の種別や，ファイル配置情報か否かといったエッジの種類を複数分けて想定したり，エッジの重みとして考慮することも考えられるが，本稿では全てのエッジを同一種，同一重みのエッジとして扱うこととした．

2 つ目の手がかりは，ノード自体に付与される属性情報である．CRUD 関係やファイル配置情報は機能・テーブルノード間のエッジに関する情報であるが，ノード自体にも情報が付属する場合がある．例えば，テーブルの場合「マスタ」「トランザクション」といったテーブルの種類を表す属性が付与されている場合が多くある．これらの属性を考慮することで，重要な属性値を持つ機能ノードとテーブルノード間に CRUD 関係がある場合，他の CRUD 関係よりも優先してそれらノードを同じクラスにまとめるべきである，といった指針を得ることが可能となる．次節で述べるグラフノード分類法の中には，各ノードの属性を直接取り入れることが可能な手法が多く存在し、これら属性情報を導入することは容易である．

3. 半教師有りグラフニューラルネットワークを用いた移行単位最適化

3.1 半教師有りグラフニューラルネットワーク

前節において，CRUD 関係を一種のグラフ構造とみなせることを示し，これにより移行単位の最適化をグラフ構造におけるノードのクラス分類問題と捉えることができた．

グラフ中のノードのクラス分類手法としてはこれまでに数多くの手法が提案されている．まず，教師無しのグラフノードクラスタリング法として，ラプラシアン行列から得られるグラフスペクトルを用いた手法や，スピングラス法 [Reichardt and Bornholdt 2006] 等の手法が知られている．教師無しのアプローチは，教師データを要しないという点でコスト面で優位であるものの，人が意図する移行単位とは異なる分類がなされる可能性が高く，実用に適さない場合もある．

一方，少量の教師データを用意することで，残るノードを人の意図に沿った形で分類できる半教師有り学習法も多く提案されており，ラベル伝搬法やランダムウォークに基づく手法 [Talukdar et al. 2008] が知られている．本稿でもシステム開発者の意図に沿った分類を行うという目的に則り，半教師有りのアプローチを採用することとする．

近年ではさらに，グラフに対して適用可能なニューラルネットワークが提案されており，グラフノードの分類や複数のグラフにおける各グラフの分類が可能となっている [Atwood and Towsley 2016, Kipf and Welling 2017]. Kipf らは，グラフスペクトルに基づく Graph Convolutional Network による半教師有り学習法を提案している [Kipf and Welling 2017]．一方，Atwood らは，diffusion process を用いた Diffusion-Convolutional Neural Network (DCNN) を提案している [Atwood and Towsley 2016]．本稿では高精度なグラフノードの分類結果を得るため，CRUD 表によって定義されるグラフ構造に対し DCNN を適用する．

DCNN は以下の式で定義される．

$$Z = f(W^c \odot P^* X) \tag{1}$$

$$Y = \arg\max(f(W^d \odot Z)) \tag{2}$$

ここで W^c, W^d は $H \times F$ のモデルパラメータ，X は $N \times F$ の全ノードに関する素性行列，Z は diffusion-convolutional activation と呼ばれ，$N \times H \times N$ のテンソルを表す．N は全ノード数，F は全素性数，H はホップ数をそれぞれ表す．P^* は 隣接行列を P としたときの冪級数であり，$P^*_{ijl} = P^j_{il}$ であらわされる．i は元ノード，l は行き先ノード，j はホップ数を表す．DCNN における diffusion とは，あるノードの素性値がホップ数と冪級数に基づいて拡散することを指し，これにより周辺ノードの影響を加味した上でグラフ中の各ノードの特徴を捉えることが可能となる．

3.2 グラフノードの分散表現

前節で述べた半教師有りグラフノード分類法を移行単位の最適化に適用した場合，各ノードの属性情報に基づく元の表現能力が十分でないために，学習が進まないケースが実験的に多く見られた．本節ではグラフ埋め込み法に基づく各ノードの分散表現を用いることでこの課題を解決する．具体的なグラフ埋め込み法としては node2vec [Grover and Leskovec 2016] と Poincaré embedding [Nickel and Kiela 2017] を導入する．本稿では，これら手法で得られた各ノードの分散表現を新たな追加の属性情報として扱う．

node2vec は，自然言語処理の分野で用いられる word2vec [Mikolov et al. 2013] をグラフ構造へ適用するよう拡張した手法で，word2vec の推定に用いられる Negative Sampling を用いた skip-gram 法の枠組みを援用し，任意のノードの近傍を得ることで自ノードに対する分散表現を得る手法である．一列のシーケンス構造で構成されるテキスト中の近傍とは異なり，グラフにおける近傍の定義は様々考えられるが，node2vec ではランダムウォークによって近傍を得る．

一方 Poincaré embedding では双対空間の一種である Poincaré 空間においてグラフ埋め込みを行う手法で，ユークリッド空間に比べ，空間を効率的に利用することができる．特に階層構造における表現能力の高さが知られており，wordnet 等への適用において高い精度が報告されている [Nickel and Kiela 2017] ．本稿で扱うシステム移行単位の解析においても，2.2 節で導入したファイル配置情報がまさに階層構造を取ることから，Poincaré embedding の効果が期待される．

4. 実験

4.1 実験条件

本節では提案手法の効果を確認するため，実際の段階的移行に基づくシステム開発において，人手で設計された 8 クラスの移行単位を教師信号，および正解とみなすことで実験を行った．機能ノード数は 5108，テーブルノード数は 6953，CRUD 関係によるエッジは約 86000，ファイル配置情報によるエッジは約 15000 である．CRUD 関係の種類は，通常の C,R,U,D に加え “Truncate” (T) という操作関係を含め計 5 種を扱う．

各ノードには事前に定義された 32 種類の属性の有無をそれぞれ付与した．人手で設計した移行単位クラスに含まれるノード数は，それぞれ 1106, 1542, 743, 2019, 2806, 800, 481, 2564 で，ノード数最大のクラスを常に推定結果とした場合の正解率，チャンスレートは，21.3% となる．

また，初期に与える教師データ（シード）は，0.1% , 1%, 10%, 30%, 50%, 70%, 90% とデータ量を変化させつつ，全機能・テーブルノードからランダムに選択した．さらに DCNN においては，教師データのうち，1 割を validation のために使用した．

本実験で取り扱う比較対象として，以下 3 つの条件の組み合わせを行う．

- グラフ構造: CRUD 関係を単独で用いる場合 (“CRUD”) と，CRUD 表にファイル配置情報を重畳させた場合 (“CRUD+filepath”) の 2 種類を比較する．
- 半教師有りノード分類手法: 半教師有りグラフノード分類法の一種であるラベル伝搬法 (LP) と，DCNN の 2 種類を比較する．DCNN の epoch 数は 100 とし，validation の誤差が最小となった時点のモデルをテストに用いた．hop 数は 3，dcnn layer および dense layer 数はそれぞれ 5 層とした．
- ノードの分散表現: グラフノード分散表現を用いないベースラインに加え，node2vec, Poincaré embedding を 128, 256, 512 次元で試したうち，LP ではテスト時最良となった node2vec の 128 次元をそのまま用いることとし，DCNN では validation のエラー率が最小となった各手法の 512 次元を用いた．

評価は，各手法が推定したクラスが，人手で作成した移行単位と一致するか否かの正解率に基づいて評価した．

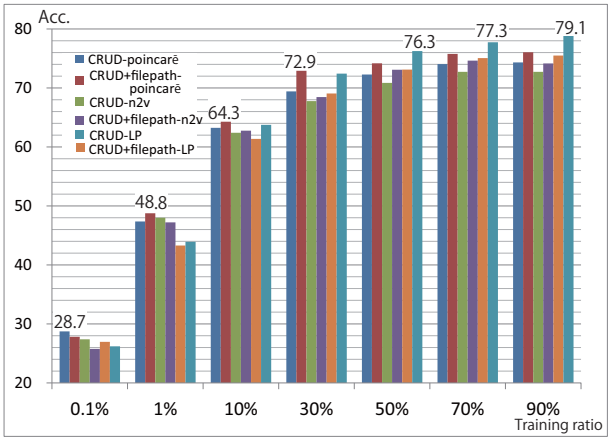

図 2: 推定した移行単位と人手で設計した移行単位との一致率．横軸は学習量割合を示す．グラフ中の数値は各学習量における最高値を表している．

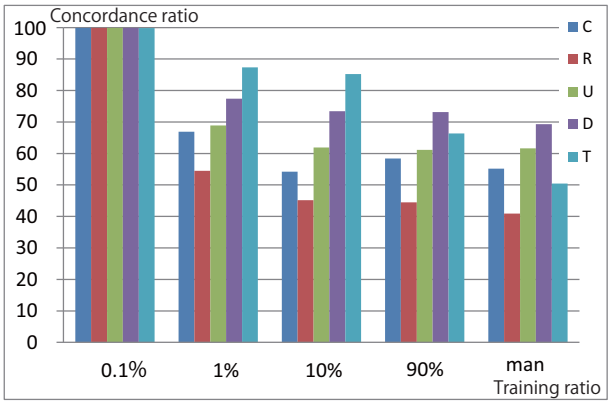

図 3: CRUD 関係を持つ 2 つのノードが同じ移行単位クラスに属する割合を学習量毎に比較した結果．“man” は人手で設計した移行単位の結果を示す．

4.2 実験結果

各比較条件における精度を教師データ量毎に比較した結果を図 2 に示す．ただし，グラフノードの分散表現を追加しない場合については，CRUD 表とファイル配置情報を併用した場合 (“CRUD+filepath”) でも，教師データが 0.1% の時に約 23%，教師データが 90% の時に約 51% と，分散表現を追加する場合と比べ精度が大きく劣ったため，見やすさのため図中から除外している．Poincaré embedding と node2vec の比較では，DCNN の全ての条件下において，Poincaré embedding の方が高い精度を示した．階層構造を持つファイル配置情報を用いる場合だけでなく，通常の CRUD 表に基づく場合でも Poincaré embedding の効果を得られた．

DCNN と LP を比較した場合，教師データ量の少ない条件下 (0.1%～30%) で比較的高い精度が得られた．一方 LP では，教師データ量が多い条件下 (50% 以上) において DCNN を用いた場合よりも高い精度を示した．実利用の面において，本提案法が特に効果を発揮するのは，教師データ量が少ない初期導入での利用と想定され，初期に人手で用意可能な教師データの数を 10～100 個程度と考えると，初期導入においては DCNN

を用いた手法が優位と言える．その後，人手のクリーニング等により移行単位の整理を進めた後，再度 LP で分類，チェックを行うことで，より実用的な効果を得られることが示唆される．

次にファイル配置情報の効果について，“CRUD” と “CRUD+filepath” の結果を比較すると，ほとんどのケースにおいてファイル配置情報を併用した場合に精度向上の効果をもたらしている．しかし，教師データ量 0.1% の DCNN においてのみ，ファイル配置情報を用いない方が精度が高い傾向が見られた．0.1% は 10 個のデータしかなく，教師データの選択方法に精度が敏感に反応した可能性が考えられるが，詳細な検証は今後の課題である

最後に提案手法によって得られた移行単位分類結果をもとに，CRUD 関係にのみ着目した最適性を評価する．図 3 は，各 CRUD 関係 (“C”,“R”,“U”,“D”,“T”) で結ばれた機能とテーブルの対が，同じ移行単位クラスに属する割合を示す．まず，人手で作成した正解となる移行単位の場合 (“man”)，機能とテーブルのクラス一致割合は全体平均で 47.8% であった．人は CRUD 関係以外の要因も加味した上で移行単位を決定するため，CRUD 関係における最適性を満たさなくなったものと考えられる．これに比べ提案手法では，教師データ量が少ない場合，比較的高い一致率を示し，特に学習量が 0.1% においてはほぼ 100% 一致している．そして教師データ量が増えるに従い，人手で作成した場合の一致率に近づく結果 (学習量が 90% において，全体平均で 51.0% の一致率) となった．このことから，提案手法は CRUD 関係における最適化を指向しつつ，人が設計した移行単位に近い分類となるように学習が進んでいると言える．

5. 結論

本稿では，大規模なシステム開発において必要となる移行単位の最適化を目的に，システムを構成する機能とテーブルの関係を表す CRUD 表に着目し，ソースコードのディレクトリ構造や機能・テーブルの属性情報を加味した一元的なグラフとみなした上で，半教師有りグラフノード分類法，特に半教師有りグラフニューラルネットワークの一種である DCNN を用いることで，比較的少量の教師データ (0.1%) でも高精度な移行単位の推定を可能とした．

今後の課題としては以下の三点が挙げられる．第一に，実際の移行単位設計時には人が考慮しているが，現在まだ扱えていない情報を，推定器に組み入れることが挙げられる．例えば，対象システムを用いた実際の業務がどのような手順で行われているかを知ることで，よりリスクの低い，人の分類により近い移行単位の設計が可能となると考えられる．第二に，本稿では既知として与えた移行単位の数を推定することで，これを解決するためには教師無しグラフクラスタリングとの併用アプローチが考えられる．第三に，本稿で各ノードの情報を分散表現から得ている箇所について，本来的にはグラフニューラルネットワークの中で一括して得られることが望ましく，グラフニューラルネットワーク自体及び適用法の改良が必要となる．これらの課題を解決することで，より高精度かつ利便性の高い移行単位の最適化が可能となり，ひいては情報システムの頑健性を一層高められると考える．

参考文献

[木原他 2015] 木原 純平, 鈴木 邦彦, 小倉 孝昭, 浜崎 俊行, レガシーマイグレーションプロジェクトにおける提供品質確保の取り組み, 2015, プロジェクトマネジメント学会誌 17(5), pages 9-14.

[Atwood and Towsley 2016] James Atwood and Don Towsley, Diffusion-Convolutional Neural Networks, 2016, In *Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pages1993–2001.

[Kipf and Welling 2017] Thomas N. Kipf and Max Welling, Semi-Supervised Classification With Graph Convolutional Networks, 2017, In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

[Grover and Leskovec 2016] Aditya Grover and Jure Leskovec, node2vec: Scalable Feature Learning for Networks, 2016, In *Proceedings of the 22nd ACM SIGKDD (KDD 2016)*, pages 855-864.

[Reichardt and Bornholdt 2006] Jörg Reichardt and Stefan Bornholdt, Statistical mechanics of community detection, 2006, *Physical Review*, 74. 1

[Talukdar et al. 2008] Partha P. Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat and Fernando Pereira, Weakly-supervised acquisition of labeled class instances using graph random walks, 2008, In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 582-590.

[Nickel and Kiela 2017] Maximilian Nickel and Douwe Kiela, Poincaré Embeddings for Learning Hierarchical Representations, 2017, In *Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 6338–6347.

[Mikolov et al. 2013] Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean, Efficient estimation of word representations in vector space, 2013, *arXiv preprint, arXiv:1301.3781.*