6K-06

大規模システム分析のためのコールグラフの多段階階層的クラスタリング *

李丞浩（法政大学情報科学部）、伊藤克亘（法政大学情報科学部）

1 Introduction

プログラムの理解はソフトウェアの再利用、デバッグ、テスト、維持及び発展に必須である。その中でもコールグラフはプログラムの理解を手伝い、プログラムの脆弱性検査、修正、手続き間分析、静的分析などで使われている。

大規模システムでは、一枚の図に全てのノードを表現するとシステムの規模が大きくなるほどエッジ数が多くなってしまい、ノードとノード間の関係をグラフを見ても複雑すぎて分かりにくいという問題がある。Codex[1]ではこのような問題を解決するためにコールグラフの多段階抽象化及び実行経路のクラスタリングを提案した。しかし、Codex の抽象化段階はオブジェクト指向言語でないとクラスレベルの抽象化段階を使えなくなる問題と、ノードの数が多くて分かりにくいという問題が残っていた。

本論文では上記の問題を解決するためにコールグラフをパッケージ、ファイル、モジュール段階に抽象化しオブジェクト言語でなくても多段階に抽象化できるようにする。また、ノードとエッジの数が多くなることを防ぐために抽象化の最後の段階を Modular Decomposition されたコールグラフを可視化することでよりグラフを見やすくすることで大規模システムのためのコールグラフの可視化を提案する。

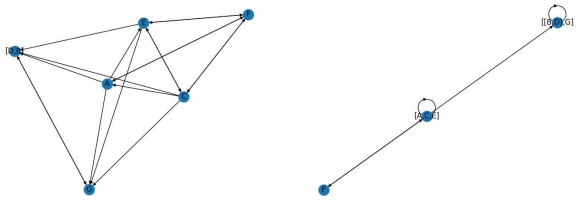
2 関連研究

2.1 コールグラフの多段階抽象化

Codex では、コールグラフを異なるレベルに細分化し多重レベル階層的抽象化を作ることを目標に、コールグラフの多段階抽象化および実行経路の階層的クラスタリングを用いてコールグラフを拡張する。

多段階抽象化ではコールグラフをシステムの最も抽象的な段階であるパッケージ段階、次にクラス段階、最終的に関数段階まで多段階に抽象化させて、階層的にクラスタリングされた実行経路をクラスタリングし、ユーザーが徐々にシステムを探索できる手法を提案した。この論文は、被験者実験の結果、大規模システムを探索するとき役立つ、段階的にシステムを探索することは分かりやすいなどの評価を得た。しかし、クラスレベルから関数レベルに細密化する時はまだノードとエッジの数が多いという問題があるという意見があった。そこで、本論文では二番目の抽象化段階から三番目の抽象化に細密化するときにノードとエッジの数がまだ多い問題を解決する手法を提案する。

2.2 Modular Decomposition



(a) Matching Neighbors　(b) Modular Decomposition

図 1: エッジを圧縮するグラフ

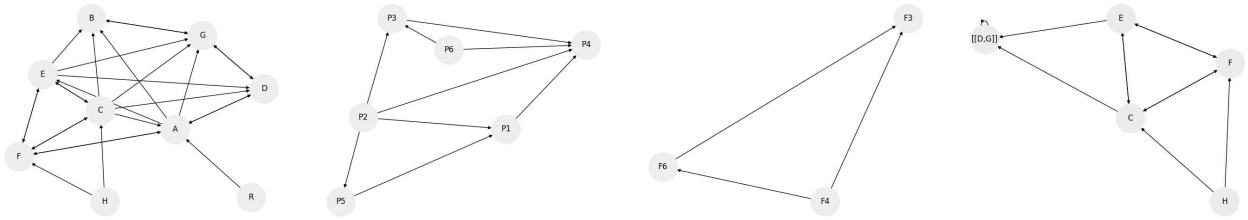
この研究 [2] ではグラフのエッジの数を圧縮するための技術を紹介している。各ノードの neighbors を一つのグループにまとめて可視化する Matching Neighbors、各モジュール内部のノードは同じモジュール内の他ノードと繋がってないか全部繋がっているかでグループ化する Modular Decomposition、モジュールの定義を緩和してエッジがモジュールの境界を越えれらるようにしてよりグラフを圧縮する Power Graph Analysis の三つの技術を比較している。その結果、ユーザーのグラフ探索速度は Matching Neighbors で可視化したグラフが元のグラフより 36%、ModularDecomposition でグラフを可視化した場合、さらに 17% の速度向上を記録した。本研究ではコールグラフ抽象化の最も低レベルである関数コールグラフに図 1 (b) のような Modular Decomposition を用いて可視化することでユーザーのシステム理解を向上させる。

3 提案手法

3.1 コールグラフの多段階抽象化

この段階では抽出された Caller-Callee 関係を Codex と同じ手法を使ってコールグラフを多段階に抽象化する。但し、従来研究で提案したパッケージ、クラス、関数のレベルではなく図 2 の (b),(c) のようにパッケージ、ファイル、モジュールレベルに構成するようにし各段階ごとに実行経路をクラスタリングさせるようにする。ここでパッケージレベルは基のグラフからパッケージだけ

* **Multilevel hierarchical clustering of call graphs for large-scale system analysis by Seungho Lee. (Hosei University) et. al.**



(a) node-link グラフ　(b) パッケージレベル　(c) ファイルレベル　(d) モジュールレベル

図 2: エッジを圧縮するグラフ

の関係を表していてファイルレベル、モジュールレベルではそれぞれ前の段階で選択されたクラスターに属する下位レベルのノードの関係を表している。

3.2 Edge Compression

この段階では図 2 (d) の D,G のようにファイルレベルの抽象化段階で選択されたクラスターに属している関数のコールグラフを可視化する前に Modular Decomposition でノードをグループ化しコールグラフのエッジの数を減らす作業を行う。

3.2.1 Matching Neighbors

エッジ圧縮を提供するモジュールの最も簡単な定義で、モジュールの全てのノードが同一な親ノードと子ノードを持っているモジュールになっている。つまり、u, v が同じグループであれば $N^+(u) = N^+(v), N^-(u) = N^+(v)$ の条件を満たす。この条件に加えて、本研究ではこの条件を緩和し u, v が同じグループであれば $u \in N^+(v), v \in N^+(u), N^+(u) \backslash \{v\} = N^+(v) \backslash \{u\}$ 及び逆の条件を満たすとき同じグループとする。

3.2.2 Modular Decomposition

Matching Neighbors 作業を完了した後、グラフからモジュール化できる部分を見つけてノードをモジュール化する。各モジュール内部のノードは他モジュールのノードに繋がってないかモジュール内の全てのノードに繋がっている。つまり、module M、vertex V で $M \subset V$ が modular decomposition で作成されるモジュールであるとしたら、$u, v \in M, N^+(u) \backslash M = N^+(v) \backslash M$ 及び逆も成り立つ条件を満たす。この条件を満たすモジュールを探してモジュール化した後、そのモジュールを新しいノードとして用いることでグラフのエッジの数を減らす。

3.2.3 コールグラフのプロセス

コールグラフを可視化し、探索するプロセスは図 2 の (a) → (b) → (c) → (d) の順に行われる。動的に動かすためにブラウザーの画面からグラフを可視化し、既存の node-link グラフをパッケージレベルを可視化したあと、パッケージレベルから選んだ経路クラスターに属するファイルの関係をファイルレベルのコールグラフで可視化する。同じくファイルレベルから選択された経路クラスターをモジュールレベルに渡してクラスター内の関数の呼び出し関係を Modular Decomposition で生成されたモジュールにノードを入れ替えて可視化するプロセスを構築する。

4 評価

実験ではコールグラフを可視化するツールを用意し参加者達に課題を提供した後、課題と関連したアンケート [3] を likert scale で答えてもらう。アンケートは likert scale の score を既存研究 [1] の結果と比較し改善されたかを確認する。また、modular decomposition で圧縮されたエッジの数と基のグラフのエッジの数の平均からエッジの減率を示す。

5 conclusion

この研究では三つのレベルに抽象化したコールグラフに Modular Decomposition を用いてエッジの数を減らす手法を提案した。本研究の核としてはまず、大規模システムのコールグラフを可視化する時コールグラフを多段階に抽象化し modular decomposition をすることで全ての実行経路を探すことや全体のコールグラフから Modular Decomposition を計算しないようにするなどの計算量を減らす方法を提案した。また、大規模システムのコールグラフを段階的に細密化する時エッジの数が多くてもエッジの数を減らすことでより分かりやすい可視化の方法を提案した。しかし、本研究にもまだ改善する点が残っている。既存研究 [2] では Power Graph Analysis 手法を使うことでよりエッジの圧縮ができていることや最適化されたアルゴリズムを用いて計算し本研究の計算速度より早いことからまだ本研究にも改善すべき点が残った。

参考文献

[1] Alanazi, et. al. Facilitating program comprehension with call graph multilevel hierarchical abstractions. *Journal Of Systems And Software*. **176** pp. 110945 (2021)

[2] T. Dwyer, et. al. "Edge Compression Techniques for Visualization of Dense Directed Graphs," in IEEE TVCG, vol. 19, no. 12, pp. 2596-2605, Dec. 2013, doi: 10.1109/TVCG.2013.151.

[3] john Brooke. SUS: A 'Quick and Dirty' Usability Scale **6** pp. 9780429157011 (1996)