

Redes de Software Sintéticas Aplicadas à Avaliação de Algoritmos de Recuperação de Modularização

Rodrigo Rocha Gomes e Souza¹

¹Departamento de Sistemas e Computação
Universidade Federal de Campina Grande (UFCG)
Caixa Postal 10.106 – CEP 58.109-970
Campina Grande – PB – Brazil

rodrigo@dsc.ufcg.edu.br

Abstract. *Software modularization recovery algorithms automatically recognize a system's modular structure by analyzing its implementation. Because the empirical evaluation of these algorithms depends on the availability of test sets composed of systems with documented modular structure, most empirical studies are, in fact, small case studies that cannot be generalized. We propose an evaluation method that rely on software models to produce arbitrarily large test sets. We hope this approach will increase the available knowledge about the algorithms and will provide insights on how they can be improved.*

Resumo. *Algoritmos de recuperação modularização de software procuram identificar automaticamente a estrutura de módulos de um sistema de software a partir de sua implementação. Como a avaliação empírica desses algoritmos depende da disponibilidade de um conjunto de testes formado por sistemas cuja organização modular é bem documentada, a maioria dos estudos empíricos são pequenos estudos de caso que não podem ser generalizados. Propomos uma forma de avaliação que usa modelos de software para produzir conjuntos de teste de tamanho arbitrário. Esperamos com isso aumentar o conhecimento disponível sobre os algoritmos e sobre como eles podem ser melhorados.*

1. Introdução

A divisão conceitual de um sistema de *software* em módulos é uma informação valiosa durante o seu desenvolvimento. Uma boa organização modular revela subconjuntos de um sistema que podem ser desenvolvidos por equipes trabalhando de forma mais ou menos independente, o que contribui para reduzir o tempo de implementação. Apesar disso, o conhecimento sobre a organização de um sistema muitas vezes é mal documentado e acaba se perdendo à medida que os desenvolvedores são substituídos [Clements et al. 2002].

Algoritmos de recuperação de modularização de *software* procuram extrair da implementação de um sistema uma organização modular e, por isso, auxiliam a divisão de tarefas entre desenvolvedores. Para realizar a decomposição de um sistema em módulos, muitos desses algoritmos usam heurísticas baseadas nas dependências existentes entre os componentes básicos da implementação do sistema.

Em sistemas implementados em linguagens orientadas a objetos, os componentes básicos são as classes e as interfaces, as quais se relacionam através de mecanismos como herança e agregação. Essas relações estabelecem dependências entre os componentes

que, unidas, formam uma rede de dependências, também chamada de rede de *software*. Formalmente, a entrada de um algoritmo de recuperação de modularização é uma rede de *software*, modelada como grafo orientado, e a saída é uma rede modular resultante do agrupamento dos componentes da rede original, como mostra a Figura 1. Os vértices do grafo representam componentes e as arestas, relações de dependência entre componentes.

Recentemente alguns pesquisadores analisaram redes de *software* sob a luz da teoria das redes complexas e encontraram propriedades topológicas marcantes que também estão presentes em redes biológicas, redes sociais e muitas outras [Myers 2003, Valverde and Solé 2003]. Descobriu-se que nessas redes a distribuição dos graus dos vértices é bem aproximada por uma lei de potência, $N(k) \sim k^{-\gamma}$, ou seja, o número de vértices ligados a exatamente k arestas é proporcional a uma potência de k com expoente negativo e constante, como mostra a Figura 2. Redes caracterizadas por essa distribuição de graus específica são chamadas de redes livres de escala. [Barabasi and Albert 1999].

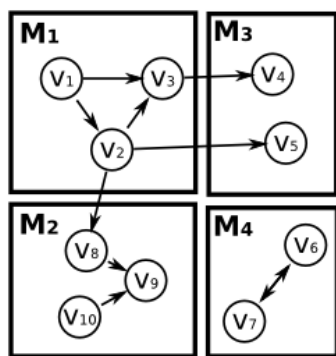


Figura 1. Uma rede modular com 10 vértices organizados em 4 módulos.

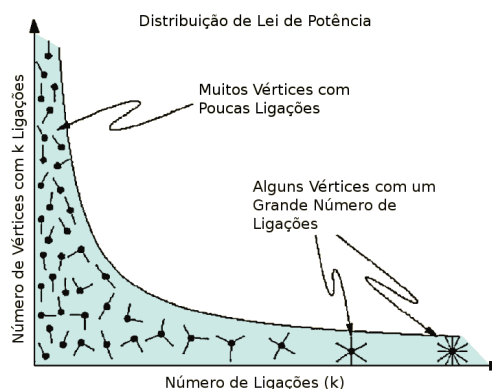


Figura 2. Distribuição de graus como lei de potência. Adaptado de [Barabasi 2007].

Para tentar explicar os mecanismos responsáveis pela formação de redes livres de escala em diversos domínios, pesquisadores criaram vários modelos. Esses modelos são algoritmos que geram vértices e arestas de maneira probabilística porém de acordo com certas regras que garantem que, quando o número de vértices tende a infinito, a distribuição dos graus dos vértices tende a uma lei de potência. Por essa razão tais modelos produzem redes similares a redes de *software*, ao menos quanto à distribuição de graus.

2. Identificação do Problema

Idealmente os algoritmos de recuperação de modularização de *software* devem encontrar organizações modulares semelhantes àquelas que seriam encontradas por especialistas nos sistemas analisados. Uma forma de testar os algoritmos consiste, pois, em aplicá-los à rede de um sistema para o qual exista uma organização modular de referência e então comparar essa organização à organização encontrada pelo algoritmo. A diferença entre as organizações modulares pode ser quantificada por uma métrica de distância entre particionamentos [Tzerpos and Holt 1999].

Infelizmente é difícil encontrar sistemas cuja organização modular é documentada, e por isso é mais comum que organizações modulares de referência sejam obtidas através de um experimento controlado no qual um grupo de programadores experientes analisa o código-fonte de um sistema e propõe uma organização modular para ele [Koschke and Eisenbarth 2000]. Trata-se, no entanto, de um experimento caro, e por isso os trabalhos presentes na literatura se limitam a analisar um pequeno número de sistemas [Wu et al. 2005].

Para que os resultados sejam estatisticamente significativos e, portanto, generalizáveis, seria necessário realizar estudos em larga escala, com uma grande amostra de sistemas com organização modular conhecida. Como não há um *benchmark* abrangente para esses algoritmos, qualquer reivindicação do tipo “o algoritmo A é melhor do algoritmo B” deve ser olhada com desconfiança. Essa dificuldade de avaliação é uma barreira para a adoção dos algoritmos na indústria e um obstáculo para o avanço das pesquisas.

3. Objetivos

Para suprir a escassez de sistemas que possuem organização modular de referência, propomos o uso de modelos capazes de sintetizar redes de *software* com organização modular embutida para servirem como conjunto de teste. Para que essa abordagem seja bem-sucedida, no entanto, é preciso que as redes sintéticas sejam realistas, isto é, semelhantes a redes extraídas de sistemas de *software* reais.

Assim, o objetivo desta pesquisa é analisar modelos que possam ser usados para avaliar algoritmos de recuperação de modularização de *software*. Esse objetivo pode ser decomposto nos objetivos específicos apresentados a seguir:

- Descobrir modelos de redes livres de escala organizadas em módulos. Embora a maioria dos modelos disponíveis na literatura ignorem a organização modular, encontramos dois modelos que produzem redes organizadas em módulos [Chen et al. 2008, Lancichinetti and Fortunato 2009]. Um terceiro modelo de redes modulares foi desenvolvido a partir da adaptação de um modelo que produz redes sem módulos [Bollobás et al. 2003].
- Desenvolver um método para avaliar o realismo de uma rede. Esse método deve encontrar um grau de realismo alto para redes de *software* reais e um grau de realismo baixo para redes de outros domínios.
- Avaliar o realismo dos modelos. Um modelo é realista se ele é capaz de produzir redes realistas.
- Avaliar algoritmos de recuperação de modularização através de sua aplicação a um grande número de redes modulares sintéticas. Os resultados poderão ser comparados com resultados experimentais encontrados na literatura.

4. Atividades Planejadas e Realizadas

Para apoiar a avaliação de realismo dos modelos, coletamos um corpo de 65 sistemas de *software* escritos na linguagem Java¹. A rede de dependências entre componentes de cada sistema foi extraída através da ferramenta Dependency Finder². Coletamos ainda cerca de 100 redes de diversos domínios, sobretudo redes biológicas.

¹Disponíveis em <http://sourceforge.net/>.

²Disponível em <http://depfind.sourceforge.net/>.

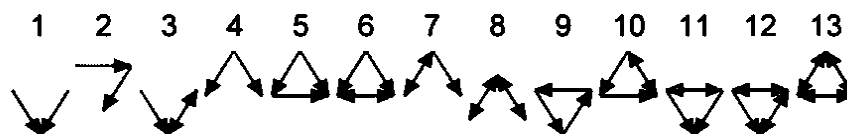


Figura 3. As 13 triades e seus números de identificação. Adaptado de [Milo et al. 2002].

Encontramos na literatura uma métrica de distância entre redes [Andrade et al. 2008]. Utilizamos a métrica em um experimento, descrito em detalhes em um artigo não publicado³, que procurou avaliar o realismo dos modelos através da comparação de distâncias entre redes sintéticas e redes de *software* reais com distâncias típicas entre redes de *software* reais. Essa métrica de distância foi, no entanto, descartada, uma vez que indicou que redes metabólicas [Jeong et al. 2000] estão mais próximas de redes de *software* do que as redes de *software* estão próximas entre si.

No momento estamos procurando outro método para avaliar o realismo de redes e, conseqüentemente, de modelos. Estamos trabalhando em um método baseado na frequência de triades, descrito na próxima seção, que pretendemos concluir até o final de julho. A análise da frequência de triades já nos permitiu detectar diferenças fundamentais entre redes de *software* e redes metabólicas.

Poderemos então usar o método para avaliar os três modelos. Esperamos aproveitar boa parte da base de dados e do conhecimento que acumulamos durante o experimento realizado anteriormente, e por isso estimamos concluir essa avaliação no final de agosto. Em setembro ou outubro submeteremos para publicação um artigo com os resultados desta etapa.

Em paralelo aplicaremos algoritmos de recuperação de modularização a redes sintéticas realistas a fim de comparar os algoritmos. Esperamos concluir essa etapa no final de setembro. A conclusão da dissertação de mestrado está prevista para meados de novembro.

5. Resultados Obtidos

Um dos resultados da pesquisa foi a concepção e a implementação de um modelo de redes modulares. Não foi possível comparar quantitativamente esse modelo aos dois modelos encontrados na literatura, uma vez que ainda não temos uma métrica para quantificar o realismo de uma rede de *software*. Já descobrimos, no entanto, que a métrica de distância entre redes descrita em [Andrade et al. 2008] não é adequada.

Estamos trabalhando em uma nova métrica de realismo baseada na frequência de triades e já temos alguns resultados. As triades, ou grafos com três vértices, podem ser resumidas em 16 possíveis configurações, das quais 13 possuem todos os vértices conectados. A Figura 3 mostra essas 13 triades e seus números de identificação.

Analizamos as redes dos 65 sistemas em Java já mencionados e constatamos que em todos eles a triade de número 1 é a que aparece com maior frequência, respondendo

³Disponível em <http://code.google.com/p/swasr/wiki/Home>.

por 30% a 90% das tríades de cada rede. A identificação da tríade mais frequente é suficiente para diferenciar redes de software de redes metabólicas, pois nestas últimas predomina a tríade de número 2.

Referências Bibliográficas

- Andrade, R. F. S., Miranda, J. G. V., Pinho, S. T. R., and Lobão, T. P. (2008). Measuring distances between complex networks. *Physics Letters A*, 372(32):5265–5269.
- Barabasi, A.-L. (2007). The architecture of complexity: From network structure to human dynamics. *Control Systems Magazine, IEEE*, 27:33–42.
- Barabasi, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509.
- Bollobás, B., Borgs, C., Chayes, J., and Riordan, O. (2003). Directed scale-free graphs. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–139, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.
- Chen, T., Gu, Q., Wang, S., Chen, X., and Chen, D. (2008). Module-based large-scale software evolution based on complex networks. *8th IEEE International Conference on Computer and Information Technology*, pages 798—803.
- Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., and Little, R. (2002). *Documenting Software Architectures: Views and Beyond*. Pearson Education.
- Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A.-L. (2000). The large-scale organization of metabolic networks.
- Koschke, R. and Eisenbarth, T. (2000). A framework for experimental evaluation of clustering techniques. In *Proc. 8th International Workshop on Program Comprehension IWPC 2000*, pages 201–210. Kosche-Eisenbarth (KE) measure.
- Lancichinetti, A. and Fortunato, S. (2009). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities.
- Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827.
- Myers, C. R. (2003). Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. *Phys Rev E Stat Nonlin Soft Matter Phys*, 68(4 Pt 2):046116.
- Tzerpos, V. and Holt, R. C. (1999). Mojo: a distance metric for software clusterings. In *Proc. Sixth Working Conference on Reverse Engineering*, pages 187–193. MoJo.
- Valverde, S. and Solé, R. V. (2003). Hierarchical small worlds in software architecture. (Directed Scale-Free Graphs).
- Wu, J., Hassan, A. E., and Holt, R. C. (2005). Comparison of clustering algorithms in the context of software evolution. In *Proc. 21st IEEE International Conference on ICSM'05 Software Maintenance*, pages 525–535.