# Busca Local Iterada para Detecção de Comunidades

**Rafael Marini, Rafael de Santiago, Alex Luciano Roesler Rese**

Laboratório de Inteligência Aplicada – Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360– 88.302-202 – Itajaí – SC – Brasil

{rafaelmarini, rsantiago}@univali.br, alexrese@outlook.com

***Abstract.*** *This paper aims to implement and evaluate the metaheuristic Iterated Local Search for the partitioning of a network into communities by maximizing the modularity, and to get this objective a study of three themes was conducted: (i) the community detection; (ii) modularity measure and (iii) the Iterated Local Search, building the necessary theoretical foundation for the development step. By applying experiments into the developed heuristic, was possible evaluate it under quality of the results gathered, and was found that the metaheuristic Iterated Local Search is plausible to resolve this problem.*

***Resumo.*** *O presente trabalho tem como objetivo aplicar e avaliar a metaheurística Busca Local Iterada para o particionamento de uma rede em comunidades através da maximização da modularidade, e para atingi-lo foi realizado o levantamento e estudo sobre três temas: (i) a detecção de comunidade; (ii) medida de modularidade e (iii) a Busca Local Iterada, construindo então o embasamento teórico necessário para etapa de desenvolvimento. Através de experimentos realizado sobre a heurística desenvolvida, foi possível avalia-la perante a qualidade das soluções retornadas e constatar que a metaheurística Busca Local Iterada é viável para a resolução deste problema.*

## 1. Introdução

A detecção das estruturas de comunidade auxilia na compreensão e na eficiência da exploração de redes aplicadas a diversas áreas de estudo, principalmente na Biologia, Sociologia e Computação, as quais são disciplinas em que os sistemas são comumente representados como grafos [Fortunato and Castellano 2012]. Formadas por um conjunto de vértices que possuem uma alta concentração de arcos ou arestas conectando-os, e uma baixa concentração de ligações com vértices de outros grupos [Girvan and Newman 2002], estas estruturas podem reter informações relevantes para a análise de dados, visto que seus elementos (vértices) compartilham funcionalidades e características em comum dentro do sistema que pertencem [Fortunato 2010].

Como exemplos de sistemas em que foi aplicada a detecção de comunidades, pode-se citar: redes sociais, na Sociologia [Liu et al. 2014]; redes de genes co-ocorrentes na Biomedicina [Wilkinson and Huberman 2004]; redes metabólicas na Biologia [Guimera and Amaral 2005].

Neste trabalho é abordado a maximização da função de modularidade, que trata de qualificar cada estrutura encontrada, ou seja, é realizado o particionamento de grafos visando o melhor valor para esta medida. A complexidade da divisão de uma rede em comunidades se dá pelo fato de que a quantidade de partições possíveis cresce

exponencialmente com o tamanho da mesma, logo algoritmos extados são impraticáveis por demandar muito tempo computacional [Fortunato 2010], assim justificando o uso de heurísticas e metaheurísticas. Neste trabalho foi utilizada a metahuerística Busca Local Iterada (*Iterated Local Search*), com o intuito de avalia-la perante ao problema apresentado.

Nas próximas seções serão abordados os temas: (i) função de modularidade; (ii) a metaheurística Busca Local Iterada; (iii) os experimentos aplicados a heurística desenvolvida; (iv) os resultados obtidos com a etapa de experimentação; (v) e as considerações finais do trabalho.

## 2. Modularidade

Com o surgimento de algoritmos para a detectar comunidades em sistemas de diferentes áreas, houve a necessidade de avalia-los frente a qualidade em sua aplicação. Dentre as formulações que qualificam as partições de um grafo retornadas pelos algoritmos de detecção de comunidades, a mais popular é a função denominada modularidade concebida por Girvan e Newman [Newman and Girvan 2004], e pode ser escrita como na equação:

$$Q(C) = \frac{1}{2m} \sum_{c \in C} \sum_{i,j \in c} \left( a_{i,j} - \frac{d_G(i) d_G(j)}{2m} \right)$$

Onde, para cada comunidade sua modularidade é calculada. Para cada par de vértices pertencentes a mesma comunidade, $a_{i,j}$ representa a ligação entre os vértices $i$ e $j$, se estes possuírem ligação o parâmetro receberá o valor 1 caso contrário receberá o valor 0, $d_G$ representa o grau do vértice, e $2m$ é a quantidade de arestas presentes no grafo [Nascimento and Pitsoulis 2013].

O resultado obtido com a aplicação da função de modularidade é dado por um valor no intervalo [-1,1], quanto mais próximo do valor 1 maior a qualidade das partições do grafo analisado. A partir deste princípio surgiram os métodos de maximização da modularidade, os quais visam particionar a rede buscando o maior valor desta medida [Fortunato 2010]. Em 2006 foi provado que o problema de maximização da modularidade, também conhecido como *Modularity Clustering Problem*, possui uma versão de decisão NP-Difícil, ou seja, a resolução para este demanda tempo combinatória, inclusive para instâncias com menos de 1000 vértices [Brandes et al. 2006].

Apesar de a maximização da modularidade proporcionar soluções de qualidade, ela possui um problema denominado limite de resolução, o qual diz respeito ao tamanho da estrutura de comunidade em relação ao tamanho do grafo, o que pode impedir a detecção de agrupamentos pequenos em relação à rede, ou seja, comunidades pequenas em grafos relativamente grades a elas, podem ser agrupadas a outras comunidades, assim tornando-as indetectáveis [Fortunato and Castellano 2012].

## 3. Busca Local Iterada

A Busca Local Iterada é uma metaheurística utilizada em problemas de otimização. Ela trabalha como um complemento de uma Busca Local, aplicando uma perturbação

na solução quando um ótimo local é atingido, assim possibilitando a exploração de soluções [Gendreau and Potvin 2010].

**Quadro 1. Pseudocódigo Busca Local Iterada**

$S_0 = GenerateInicialSolution$

$S^* = LocalSearch(S_0)$

***REPEAT***

$S' = Pertubation(S^*, history)$

$S^{*\prime} = LocalSearch(S')$

$S^* = AcceptanceCriterion(S^*, S^{*'}, history)$

***UNTIL*** *termination condition met*

O Quadro 1 exibe o pseudocódigo da Busca Local Iterada, o qual possui cinco etapas: (i) geração da solução inicial, construída de forma aleatória ou gulosa, a solução inicial é o ponto de partida da exploração do espaço de soluções de um problema; (ii) busca local, técnica de intensificação de solução, varre as soluções vizinhas em busca de uma melhor; (iii) perturbação, modifica uma solução, para atingir outra solução do espaço; (iv)critério de aceitação, decide se a solução encontrada será aceita ou não; (v) e a condição de parada do algoritmo.

### 3.1. Descrição das etapas implementadas

Para a solução inicial, foi desenvolvido um algoritmo construtivo que monta a solução da seguinte forma: (i) selecionar o vértice de maior grau e que ainda não pertence a uma comunidade; (ii) atribuir este juntamente com seus vizinhos a uma nova comunidade; (iii) repetir as etapas anteriores até que cada um dos vértices do grafo pertença a uma comunidade.

Na etapa de busca local, foi utilizada a heurística *best improvement*, um algoritmo de intensificação, onde dado uma estrutura de vizinhança, este a percorre até atingir um ótimo local [Hansen and Mladenović 2006]. As estratégias para construção das estruturas de vizinhança utilizadas neste trabalho foram, Singleton, Fusion, Division, Redistribution [Aloise and Caporossi 2012] e 1-Vizinhança [Nascimento and Pitsoulis 2013].

Para a perturbação foi adotado um nível de força, um valor do intervalo entre 0.1 à 0.9, que delimita a quantidade de vértices que são selecionados e realocados de forma aleatória entre as comunidades da solução.

Como critério de aceitação, somente são aceitas soluções que forem melhores que a solução incumbente. A heurística é encerrada quando for atingido 1000 iterações sem alteração na melhor solução.

## 4. Experimentos

Os experimentos foram realizados através da aplicação de instâncias do problema de maximização da modularidade submetidos à heurística desenvolvida, medindo o tempo de execução e a qualidade dos resultados obtidos para cada uma.

A Tabela 1 exibe as sete instâncias clássicas que foram utilizadas nos experimentos, instâncias utilizada como *benchmark* para algoritmos que resolvem a maximização da modularidade. Esta é composta por quatro colunas, sendo: (1) Instância, o nome da rede; (2) Modularidade, o melhor valor conhecido da medida de modularidade; (3) Vértices, o número de vértices da rede e (4) Arestas, o número de arestas presentes na rede.

**Tabela 1. Instâncias clássicas para maximização da modularidade**

| Instância | Modularidade | Vértices | Arestas |
|---|---|---|---|
| Karate | 0,419790 | 34 | 78 |
| Dolphins | 0,528519 | 62 | 159 |
| Lesmis | 0,566688 | 77 | 254 |
| Polbooks | 0,527237 | 105 | 441 |
| Adjnoun | 0,313367 | 112 | 425 |
| Football | 0,604570 | 115 | 613 |
| Jazz | 0,445144 | 198 | 2742 |

Estas instâncias são compostas por grafos não dirigidos e grafos não dirigidos e ponderados, as quais estão disponíveis no site do evento DIMACS [DIMACS 2012].

A Tabela 2 exibe as sete instâncias aleatórias que foram utilizadas nos experimentos. Esta é composta por quatro colunas, sendo: (1) Instância, o nome da rede; (2) Modularidade, o melhor valor da medida de modularidade alcançado por Nascimento e Pitsoulis (2013); (3) Vértices, o número de vértices da rede e (4) Arestas, o número de arestas presentes na rede.

**Tabela 2. Instâncias aleatórias para maximização da modularidade**

| Instância | Modularidade | Vértices | Arestas |
|---|---|---|---|
| 100_1 | 0,42399 | 100 | 95 |
| 100_2 | 0,52270 | 100 | 98 |
| 100_3 | 0,45740 | 100 | 99 |
| 100_4 | 0,40169 | 100 | 97 |
| 100_5 | 0,45844 | 100 | 97 |
| 100_6 | 0,48794 | 100 | 97 |
| 100_7 | 0,39175 | 100 | 95 |

As instâncias citadas na Tabela 2 são compostas por grafos não-dirigidos e não ponderados, gerados e publicados por Nascimento e Pitsoulis (2013).

Cada experimento foi realizado sobre cada uma das configurações abaixo, submetendo todas as combinações possíveis sobre elas:

- Cinco vizinhanças: parâmetro $v \in N = \{singleton, division, fusion, redistribution, 1 - vizinhança\}$;
- Nível de perturbação: parâmetro $\rho$ que identifica quão forte é a perturbação, onde $\rho \in$ P $= \{0.1,0.2,0.3,0.4,0.5,0.7,0.8,0.9\}$;
- Quatorze instâncias: Karate, Dolphins, Lesmis, Polbooks, Adjnoun, Football, Jazz, 100_1, 100_2, 100_3, 100_4, 100_5, 100_6, e 100_7.

Para cada combinação de parâmetros foram executadas trinta replicações, devido ao fator aleatório presente no algoritmo, totalizando 18900 experimentos. Estes foram executados em um microcomputador com processador Intel Core™ i5-2400 CPU @ 3.10GHz, 4GB de memória RAM e sistema operacional Xubuntu 14.04. Levando em consideração seus quatro núcleos, os experimentos foram executados em paralelo, sendo escalonados a quatro threads disponíveis no processador.

## 5. Resultados

Com a execução dos experimentos foram geradas 30 amostras para cada uma das 45 combinações, as quais foram submetidas ao teste de Wilcoxon, um teste de hipótese pareado e não paramétrico [WILCOXON 1946]. Para determinar a combinação de parâmetros que conduz a resultados de maior qualidade, duas hipóteses foram aplicadas a $C_i$ e $C_j$:

- $H_0$: a combinação $C_i$ gera soluções com o mesmo valor de modularidade que $C_j$;
- $H_1$: a combinação $C_i$ gera soluções com valor de modularidade maior que $C_j$.

E para determinar a combinação de parâmetros que conduz a resultados de menor tempo de execução, duas hipóteses foram aplicadas a $C_i$ e $C_j$:

- $H_0$: a combinação $C_i$ gera soluções em mesmo tempo de execução que $C_j$;
- $H_1$: a combinação $C_i$ gera soluções em menor tempo de execução que $C_j$.

Com o resultado do teste de hipóteses, foi possível identificar o número de vezes que determinada combinação de parâmetros rejeitou a hipótese nula, tanto para qualidade de soluções como para o tempo de execução, assim permitindo a criação de dois *rankings*.

### 5.1. Qualidade

A Tabela 3 exibe o *ranking* com as dez combinações de parâmetros que obtiveram melhor classificação em relação as demais, sob o critério de qualidade de soluções, ou seja, rejeitaram a maior quantidade de vezes a hipótese nula. Esta é composta por quatro colunas, sendo: (1) Posição, posição no *ranking*; (2) Vizinhança, o nome da estratégia para construção de vizinhança; (3) Perturbação, o nível de força da perturbação e (4) Rejeições, o número de vezes que $H_0$ foi rejeitada.

**Tabela 3. *Ranking* de qualidade de soluções**

| Posição | Vizinhança | Perturbação | Rejeições |
|---|---|---|---|
| 1º | 1-Vizinhança | 0.5 | 38 |
| 1º | 1-Vizinhança | 0.6 | 38 |
| 1º | 1-Vizinhança | 0.7 | 38 |
| 1º | 1-Vizinhança | 0.8 | 38 |
| 2º | 1-Vizinhança | 0.3 | 37 |
| 2º | 1-Vizinhança | 0.4 | 37 |
| 2º | 1-Vizinhança | 0.9 | 37 |
| 3º | 1-Vizinhança | 0.1 | 36 |
| 3º | 1-Vizinhança | 0.2 | 36 |
| 4º | *Fusion* | 0.1 | 34 |

Como pode-se observar, de todas as dez combinações que compõe o *ranking* de qualidade, as nove primeiras possuem a estratégia para construção de vizinhança 1-Vizinhança, ou seja, esta estratégia trouxe soluções de melhor qualidade comparando-a com as demais utilizadas (*Singleton*, *Division*, *Fusion* e *Redistribution*).

Na Figura 1 é exibido o gráfico de qualidade média, o qual relaciona a proximidade do valor médio com o melhor valor de modularidade conhecido na literatura. Por motivos de visualização, apresenta-se apenas as melhores combinações de parâmetros (sob o critério de qualidade). As combinações apresentadas, referem-se as dez listadas na Tabela 3, ranking de qualidade.

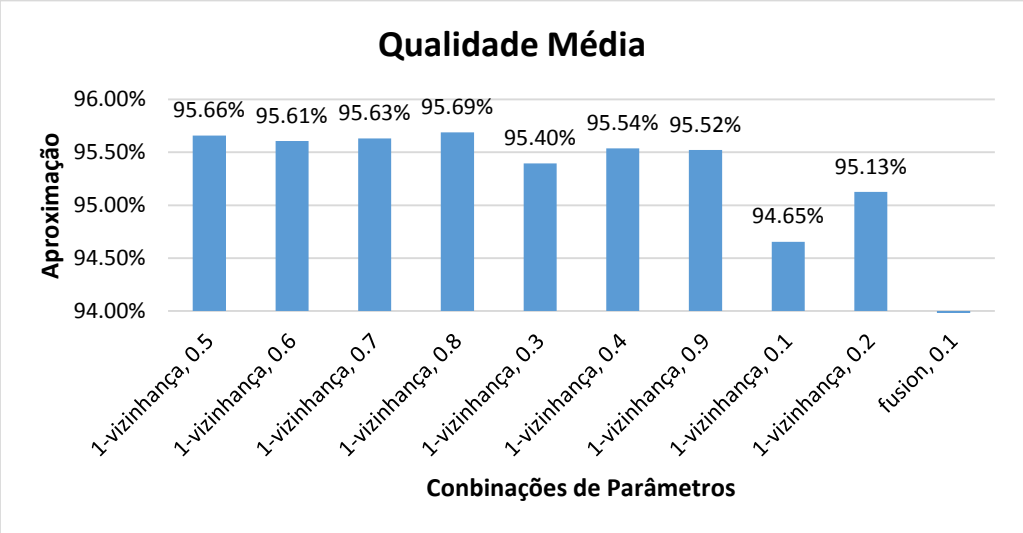


**Figura 1. Gráfico da qualidade média em relação as combinações de parâmetros**

Pode-se observar que a maioria das combinações obteve um valor médio de modularidade próximo ao melhor valor conhecido. A combinação que sobressaiu-se perante as demais, com aproximação de 95,69%, possui a estratégia para construção de vizinhança 1-Vizinhança, e nível de força da perturbação de 0.8.

## 5.2. Tempo

Um ranking contendo as dez combinações de parâmetros que rejeitaram o maior número de vezes a hipótese nula, em relação ao tempo de execução, é exibido no Tabela 4. Esta é composta por cinco colunas, sendo: (1) Posição, posição no *ranking*; (2) Vizinhança, o nome da estratégia para construção de vizinhança; (3) Perturbação, o nível de força da perturbação; (4) Rejeições, o número de vezes que $H_0$ foi rejeitada e (5) Complexidade, a complexidade amortizada, calculada com 99% de confiança, a qual diz respeito ao número de operações necessárias para se executar uma instância em uma determinada combinação de parâmetros, dado o tamanho do problema ($n$).

**Tabela 4. *Ranking* de tempo de execução**

| Posição | Vizinhança | Perturbação | Rejeições | Complexidade |
|---|---|---|---|---|
| 1º | 1-Vizinhança | 0.9 | 42 | $n^{1.69\pm0.17}$ |
| 2º | 1-Vizinhança | 0.7 | 40 | $n^{1.84\pm0.2}$ |
| 2º | 1-Vizinhança | 0.8 | 40 | $n^{1.71\pm0.17}$ |
| 3º | 1-Vizinhança | 0.5 | 39 | $n^{1.74\pm0.2}$ |
| 3º | 1-Vizinhança | 0.6 | 39 | $n^{1.8\pm0.18}$ |
| 4º | 1-Vizinhança | 0.3 | 37 | $n^{1.87\pm0.22}$ |
| 4º | 1-Vizinhança | 0.4 | 37 | $n^{1.79\pm0.21}$ |
| 5º | 1-Vizinhança | 0.2 | 36 | $n^{1.92\pm0.21}$ |
| 6º | *Fusion* | 0.1 | 35 | $n^{2.24\pm0.1}$ |
| 7º | *Redistribution* | 0.1 | 34 | $n^{2.21\pm0.06}$ |

Pode-se observar que as combinações de parâmetros que possuem a estratégia para construção de vizinhanças 1-Vizinhança aparecerem em maior número, o que implica, em relação ao tempo de execução, sua predominância perante as demais estratégias comparadas. Com relação a complexidade amortizada, vê-se que esta não segue a mesma colocação do ranking de tempo, pois não está diretamente relacionada ao tempo, e sim ao número de operações necessário para executar uma instância, isto significa, que para instâncias maiores, o esforço exigido será maior, e possivelmente, a combinação de parâmetros que ficou na sétima colocação será mais rápida que a sexta colocada, pois possui complexidade amortizada menor.

O gráfico da Figura 2 exibe o tempo médio de execução em segundos para cada instância a qual a heurística, com determinada configuração de parâmetros foi executada. Foram relacionadas as dez configurações presentes no ranking de tempo na Tabela 4. No gráfico, ao lado do nome da instância está descrito o número de vértices que ela possui.

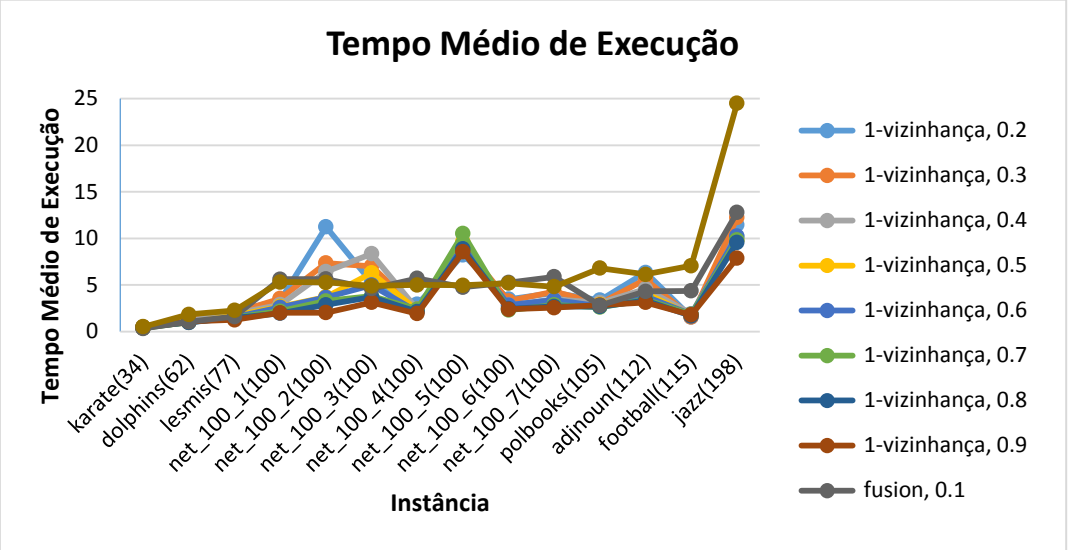


**Figura 2. Gráfico do tempo médio de execução**

No gráfico de tempo médio de execução, nota-se o fato de que o número de vértices está diretamente relacionado com o tempo de execução, ou seja quanto maior a quantidade de vértices que a rede possui, mais serão as possíveis combinação para se formar comunidades, logo será despendido uma maior quantidade de tempo para encontrar as combinações que favorecem um maior valor de modularidade. Comparando-se com as demais, a combinação que utiliza os parâmetros 1-Vizinhança e 0.9, foi a que conseguiu o menor tempo médio de execução.

### 5.3. Comparativo

Através dos experimentos realizados foi possível determinar qual combinação de parâmetros aplicados à heurística implementada que trazem os resultados de maior qualidade. Desta maneira pode-se então comparar os melhores resultados atingidos, com os melhores resultados relatados na literatura para as 7 instâncias clássicas. A Tabela 5 mostra este comparativo, esta é composta pelas colunas: (i) Instância, nome da instância; (ii) $Q^*$, o melhor valor de modularidade descrito na literatura, retirados de Nascimento e Pitsoulis (2013); (iii) $Q'$, o melhor valor de modularidade atingido pela heurística implementada neste trabalho.

**Tabela 5. *Comparativo de resultados***

| **Instância** | $Q^*$ | $Q'$ |
|---|---|---|
| Karate | 0,4197 | 0.4197 |
| Dolphins | 0,5285 | 0.5285 |
| Lesmis | 0,5666 | 0.3343 |
| Polbooks | 0,5272 | 0.5272 |
| Adjnoun | 0,3133 | 0.3086 |
| Football | 0,6045 | 0.6045 |
| Jazz | 0,4451 | 0.4451 |

Como pode-se observar, foi possível chegar ao melhor valor para quase todas as instâncias clássicas aplicadas, atingindo resultados de qualidade, ou seja, próximos aos melhores conhecidos na literatura.

## 6. Conclusão

Neste trabalho foi realizada a implementação e análise da metaheurística Busca Local Iterada (*Iterated Local Search*), gerando então uma heurística capaz de trazer resultados de qualidade para o problema de detecção de comunidades através da maximização da modularidade. Com a aplicação dos experimentos sobre esta, constatou-se que a metaheurística é viável para a resolução deste problema, visto que, foi possível atingir as melhores soluções conhecidas pela literatura para a maioria das instâncias aplicadas. Também pode-se observar que, das combinações de parâmetros para a heurística desenvolvida neste trabalho, a que possui estratégia para construção de vizinhança 1-Vizinhança e nível de força para perturbação de 0.8 sobressaiu-se perante as demais, obtendo soluções de maior qualidade.

## 7.Referências


Aloise, D. and Caporossi, G. (2012). Modularity maximization in networks by variable neighborhood search. *10th DIMACS implementation challenge – graph partitioning and graph clustering.*

Brandes, U. , Delling, D. , Gaertler, M. and Görke, R. (2006). On modularity-np-completeness and beyond. n. 001907.

DIMACS (2012), 10th DIMACS Implementation Challenge - Graph Partitioning and Graph Clustering, http://www.cc.gatech.edu/dimacs10/archive/clustering.shtml, October.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*.

Fortunato, S. and Castellano, C. (2012). Community structure in graphs. *Computational Complexity*, pages 490–512.

Gendreau, M. and Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. second ed.Heidelberg: Springer. v. 146p. 648

Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*.

Guimera, R. and Amaral, L. (2005). Functional cartography of complex metabolic networks. *Nature*, v. 433, n. February, pages 895–900.

Hansen, P. and Mladenović, N. (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, v. 154, pages 802–817.

Liu, X. , E, H. , Tong, J. and Song, M. (2014). Collaborative recommendation based on social community detection. *The Journal of China Universities of Posts and Telecommunications*, v. 21, n. July, pages. 20–45.

Nascimento, M. C. V. and Pitsoulis, L. (2013). Community detection by modularity maximization using GRASP with path relinking. *Computers & Operations Research*, v. 40, n. 12, pages. 3121–3131.

Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*.

WILCOXON, F. (1946). Individual comparisons of grouped data by ranking methods. *Journal of economic entomology*, v. 39, page 269.

Wilkinson, D. M. and Huberman, B. a (2004). A method for finding communities of related genes. *Proceedings of the National Academy of Sciences of the United States of America*, v. 101 Suppl , pages 5241–8.