2014-07

Multilevel refinement based on neighborhood similarity

International Database Engineering & Applications Symposium, 18th, 2014, Porto.
http://www.producao.usp.br/handle/BDPI/46712

Multilevel refinement based on neighborhood similarity

Alan Valejo, Jorge Valverde-Rebaza, Brett Drury and Alneu de Andrade Lopes
Department of Computer Science
ICMC, University of São Paulo
C. P. 668, CEP 13560-970, São Carlos, SP, Brazil
{alan, jvalverr, bdrury, alneu}@icmc.usp.br

ABSTRACT

The multilevel graph partitioning strategy aims to reduce the computational cost of the partitioning algorithm by applying it on a coarsened version of the original graph. This strategy is very useful when large-scale networks are analyzed. To improve the multilevel solution, refinement algorithms have been used in the uncorsening phase. Typical refinement algorithms exploit network properties, for example minimum cut or modularity, but they do not exploit features from domain specific networks. For instance, in social networks partitions with high clustering coefficient or similarity between vertices indicate a better solution. In this paper, we propose a refinement algorithm (RSim) which is based on neighborhood similarity. We compare RSim with: 1. two algorithms from the literature and 2. one baseline strategy, on twelve real networks. Results indicate that RSim is competitive with methods evaluated for general domains, but for social networks it surpasses the competing refinement algorithms.

Categories and Subject Descriptors

I.5.3 [**Pattern Recognition**]: Clustering—*Algorithms*; H.2.8 [**Database Applications**]: Data mining; E.1 [**Data**]: Data structures—*Graphs and networks*; G.4 [**Mathematics of Computing**]: Mathematical software—*Algorithm design and analysis, Efficiency*

Keywords

Graph Clustering, Multilevel Partitioning, Refinement, Complex Networks, Social Networks

1. INTRODUCTION

The study of partition (community or cluster or module) structures in complex networks[1] has recently become a popular area of research [35, 9]. Graph partitioning is an important problem with many applications such as clustering of web clients who have similar interests to improve the performance of web services [17], customer clustering with similar interests to improve the performance of recommendation systems [36], and others.

Graph partition techniques aim to divide the set of vertices of a graph into k disjoint partitions such that the number of edges connecting vertices among partitions is minimal [35, 15]. Vertices belonging to the same partitions share common properties and have similar roles, consequently graph partitioning is useful to understand the topological structure and dynamic processes of graphs [9].

The graph partitioning problem is $\mathcal{NP}$-complete, consequently the identification of an optimal solution is a computationally expensive task, which may be infeasible for large-scale networks [10]. A possible solution could be to identify an optimal topological partition by optimizing some objective function. A drawback to this approach is that differing graph partitioning algorithms may obtain different solutions when evaluated in the same graph. Each graph partitioning algorithm has a trade-off between accuracy and execution time, and consequently the context of the problem will influence the selection of the algorithm [15, 9].

One of the most popular approach is the multilevel graph partition strategy (MGR) [12, 4], which: reduces the size of the graph by collapsing vertices and edges, partitions the reduced graph, and then uncoarsening it to construct a partition for the original graph. Figure 1 shows in more detail the three phases of MGR, which in this case are: **(i)** the initial graph G_0 being reduced recursively (coarsening process) into a sequence of smaller graphs $G_0, G_1, \ldots, G_N$; **(ii)** partitioning of the reduced graph G_N and **(iii)** the partition of the graph G_N is projected back to the initial graph G_0 (uncoarsening process).

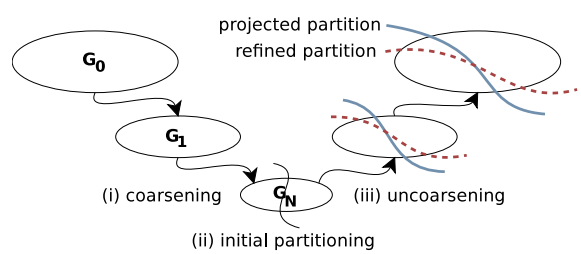


Figure 1: The multilevel graph partitioning strategy scheme. The solid lines represent the projection of the partition. The dashed lines represent the refinement process

[1] We do not distinguish network and graph.

IDEAS'14 July 07 - 09 2014, Porto, Portugal

http://dx.doi.org/10.1145/2628194.2628227

An important criteria of the multilevel strategy is the choice of the initial partition because an non-optimal choice may lead the partition algorithm to converge to a local minima[2]. It is important to note that, even though the graph G_i is at a local minima, the projected partition of the graph G_{i-1} can not be at local minima, therefore the use of a local refinement algorithm it is possible to improve the quality of the partition [15]. The refinement process is typically performed by vertex moving among adjacent partitions. Figure 2 shows local refinement operations. Given an initial partition, the refinement process allows: (i) vertex moving among partitions, (ii) merge two partitions, and (iii) create new partitions.

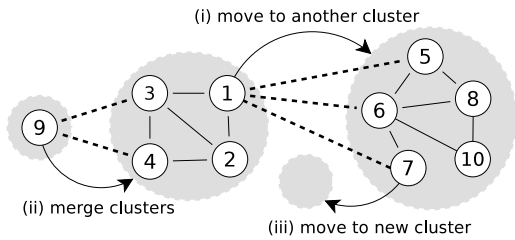


Figure 2: Local refinement operations. Dashed lines represent relationships among vertices belonging to different partitions. Arrows indicate refinement operations

Refinement methods tend to improve the partition quality by an optimization of an objective function, for instance, cut minimization and balancing [16, 8, 28, 29, 12, 15] or maximization of the modularity [23, 30, 35, 26, 27]. These methods use the general structural properties of complex networks, however, some intrinsic properties of specific types of complex networks have not been explored efficiently in the refinement process.

One of the most studied types of complex networks are social networks, which are characterized by: high clustering coefficient, significant assortativity mixing and numerous common relationships among their members [22, 23]. These properties are quantified using neighborhood and similarity measures [20, 36, 32, 33].

In this paper, we propose a new algorithm, called *Refinement algorithm based on similarity* (RSim), which uses intrinsic social network properties. Our proposal uses the similarity between pairs of vertices to perform refinement operations and thus improve the partition quality maximizing the number of cycles with three vertex and merging vertices with similar degrees to the same partitions.

In order to systematically describe and compare our proposal with existing refinement algorithms, we performed experiments on twelve real complex networks. Analysis from a post-hoc test [6] indicated that, in general our proposal has a comparative performance with methods described in the research literature. Nevertheless, when we consider the analysis only on social networks, our proposal outperformed all other methods. In addition, to evaluate the performance of our proposal in a problem of the real world, we present a case study which addresses the problem of ambiguous citations in scientific cooperation networks. Also, independently of the type of complex network analyzed our proposal is faster than compared graph partitioning methods.

The paper is organized as follows: Section 2 describes graph definitions and the multilevel graph partition approach, respectively. Section 3 details our proposal. Section 4 shows the experimental results. Section 5 discusses the real problem of ambiguous citations and shows how our proposal outperforms various refinement algorithms. Finally, conclusions are drawn in Section 6.

[2]The solution is at a local minima if the exchange of vertices between partitions not minimize cutting [15].

2. MULTILEVEL GRAPH PARTITIONING

Consider an undirected graph $G = (V, E, W)$, where V is the set of vertices and E is the set of edges. An edge is defined for the relation $e = \{(v, u) = (u, v) \mid v, u \in V\}$. The weight of the edge formed between the pair of vertices v and u is represented by $w(v, u)$. The total number of vertices is $n = |V|$ and the total number of edges is $m = |E|$, where "|.|" indicates the cardinality of a set.

The basic structural definition for a vertex $v \in V$ is its degree $k(v) = \sum_{u \in V} w(v, u)$, which denotes the total weight of its edges. Similarly, the definition of its neighborhood is $\Gamma(v) = \{u \mid (v, u) \in E \ \vee \ (u, v) \in E\}$, which denotes the set of neighbors of v.

Denote $C = \{C_1, \ldots, C_k\}$ as the partitioning of V into disjoint non-empty partitions such that $C_i \cap C_j = \emptyset, \forall\ i \neq j$. A partition C is considered trivial if $k = 1$ or $k = n$. A partition C_i composed just by one vertex is called as singleton. The internal degree of v is the number of their neighbors that are in the same partition, i.e. $k_{C_i}(v) = |\{(v, u) \mid v, u \in C_i, (v, u) \in E\}|$. Given a partition C, the number of edges whose vertices belong to different partitions is called the edge cut and is defined as $\sum_{u \in C_i, v \neq C_i} w(u, v)$ where $1 \leq i \leq k$.

Most of partitioning algorithms tend to optimize the modularity due to it is one of the most widely used quality measures [25, 27, 26, 35, 30]. The modularity has a formal relation to intra-cluster density and inter-cluster sparsity [25]. Formally, the modularity is defined as stated in Equation 1:

$$Q = \frac{1}{2m} \sum_{u,v} [A_{u,v} - \frac{k(u)k(v)}{2m}] \delta(C_u, C_v), \qquad (1)$$

where $A_{u,v}$ is an entry in the adjacency matrix A and the function $\delta(C_u, C_v)$ is 1 if $C_u = C_v$ and 0 in another case. Modularity values greater than 0.3 indicate a good partition quality [23].

2.1 Multilevel approach

Given an initial graph G_0, with weights in the vertices and edges. Multilevel approach aims to recursively reduce this graph into a graph G_N which is as small as it is desired. After this, G_N is partitioned into k sub-graphs using a partitioning algorithm. Lastly, the graph G_N is projected back to the intermediate graphs ones until the original graph G_0. This approach has three phases: *coarsening*, *initial partitioning* and *uncoarsening*, illustrated in Figure 1.

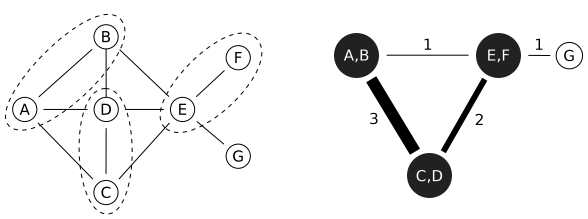
Algorithm 2.1 shows the steps of the multilevel approach. In this algorithm, the graph G_0 indicates the original graph. The *rf* is a variable used to calculate the size of the matching in the coarsening phase. The three phases of the multilevel approach are represented by functions: *coarsener*, *partitioning* and *refiner*.

Algorithm 2.1: Multilevel graph partitioning

Input: graph G_0, reduction factor rf
1 $G_i \leftarrow G_0$, i $\leftarrow$ 0;
2 **repeat**
3 $\quad G_{i+1} \leftarrow$ coarsener(G_i, rf);
4 $\quad$ i $\leftarrow$ i+1;
5 **until** G_i *not coarse enough*;
6 clustering $\leftarrow$ partitioning(G_i);
7 **while** G_i *!=* G_0 **do**
8 $\quad$ project clustering from G_i to G_{i-1};
9 $\quad$ clustering $\leftarrow$ refiner(G_{i-1}, clustering);
10 $\quad$ i $\leftarrow$ i+1;
11 **return** *clustering*

2.2 Coarsening Phase

In this phase, the graph G_0 is recursively reduced in a sequence of smaller graphs $G_1, G_2, ..., G_N$, such that $|V_0| > |V_1| > ... > |V_N|$, i.e. decreases the number of vertices and edges. The G_N, is called a coarse graph, which can be achieved in various ways. In most coarsening schemes a set of vertices of G_i are merged, forming a super-vertex sV in the coarser graph G_{i+1}. For the graph G_{i+1} to be a good representation compared to its previous coarser versions, the weight of $sV = \{v, u\} \in V_{i+1}$ is given by the sum of weights v and $u \in V_i$. Furthermore, in order to preserve the connectivity information, the edges of sV are obtained by the join of edges into $u \in V_i$. The set of pairs of candidate vertices for merging is denominated **matching** (show Figure 3).



(a) G_i (b) G_{i+1}

Figure 3: The coarsening graph process uses the matching concept. The dashed ellipses represent the set of pairs of candidate vertices to merged by matching. In G_i, the all edges weight is 1

The coarsening phase is repeated until a graph with desired size is attained. The size of a coarse graph may be defined by the absolute number of vertices, a reason related to the original size, or a number of iterations. Algorithm 2.2 summarizes the coarsening process in pseudo-code. In this algorithm, G_i is the graph to be coarsened. The *rf* limits the number of pairs of vertices merged by determining the size of the matching. When $rf = 0.5$ (maximum value), the number of vertices in the graph is reduced to half. Based on this, the size of the matching (*mergeCount*) is defined by the reduction factor (*rf*) times by the number of vertices in G_i, therefore each iteration reduces the number of vertices of a constant factor to the logarithmic scale.

The *selector* represents the merge selection quality to use. The function *getVertex* selects the vertex by their selection quality. Next, the function *bestNeighbor* stores the best neighbor identified by the selector. One of the most popular selection algorithms is the Heavy-edge Matching (HEM) [13]. The HEM searches the set of pairs of vertices that reduces the most of the edge weights in the graph. Thus, HEM selects the vertex u adjacent to v, whose edge (u, v) is the heaviest. We notice that, the HEM complexity is $O(|E|)$.

Algorithm 2.2: Coarsener

Input: graph G_i, reduction factor rf
1 *mergeCount* $\leftarrow rf *$ number of vertices in $G_i + 1$;
2 $i \leftarrow 0$;
3 **while** *mergeCount* > 0 **do**
4 $\quad$ a $\leftarrow$ selector.bestVertex(G_i);
5 $\quad$ b $\leftarrow$ selector.bestNeighbor(a, G_i);
6 $\quad$ **if** *a and b were not merged* **then**
7 $\quad\quad$ matching[i] $\leftarrow$ merge pair (a,b);
8 $\quad\quad$ mergeCount $\leftarrow$ mergeCount - 1;
9 $\quad\quad$ $i \leftarrow i + 1$;
10 **return** *matching*

2.3 Initial Partitioning Phase

This phase computes the partition C in the coarser graph G_N. Given a graph G_N as the input parameter and a vector C as output (representing the partitioning of G_N), different algorithms for graphs partitioning can be used. Depending on the setting of the coarsening phase, the graph G_N can be very small, therefore it is possible to use computationally expensive graph partitioning algorithms without a impact on general performance. [13]. This phase represents the *partition()* function in the Algorithm 2.1.

2.4 Uncoarsening Phase

During the uncoarsening phase, the partitioning graph G_N is recursively expanded until the original graph. For each iteration, each super-vertex sV from the graph G_{i+1} is divided in its origin vertices u and v. The edges which link two sV are distributed between u and v keeping original structure. In the same way, the partitioning C is projected by the intermediate levels $(G_{N-1}, G_{N-2}, ..., G_0)$ until the original graph. By decomposing a sV, the original vertices u and v are added to partition C_i, so that $sV \in C_i$. In this phase, it is possible to use refinement heuristics in order to improve the quality of the solution.

Refinement polices. As stated previously, refinement algorithms may be used in the uncoarsening phase. This process is done through local operations by moving vertices among partitions (see Figure 2). Given a partitioning $C = \{C_1, C_2\}$, the goal is to select $C'_1 \subset C_1$ and $C'_2 \subset C_2$, so that the partitioning $(C_1 - C'_1) \cup C'_2$ and $(C_2 - C'_2) \cup C'_1$ optimize an objective function, for instance, the graph cut [14].

In the research literature most refinement approaches are based on the Kernighan-Lin (KL) algorithm [16]. KL performs continuous changes of vertices among partitions in order to reduce the graph cut and increase the balancing of the partitions. The KL complexity is $O(n^3)$ but it can be done in $O(|E|log|E|)$. Fiduccia and Mattheyses [8] proposed a modification (FM) of KL with linear computational cost $O(|E|)$. The FM algorithm uses accurate data structures

that makes KL computationally more efficient. The KL algorithm has also been modified in order to maximize the modularity [23].

These algorithms are efficient but are limited to improve a bisection[3]. In many contexts, the partitioning is not limited to a bisection, however refining k-partitions is more complex than refining a bisection due to vertices can move to many partitions [27].

One of the first adaptations of the KL algorithm to refine k-partitions was proposed by Sanchis [28] in the context of hypergraphs. Hendrickson and Leland [12] proposed an algorithm for the k-partitioning of graphs using a strategy of local refinement, however the algorithm complexity depends of the number of partitions, i.e. $O(k|E|)$, therefore this algorithm can be used only when k is small. Karypis and Kumar [15] proposed an algorithm (KK) where the complexity is independent of the number of partitions (simplifying the KL). By using the gain concept based on the degree of vertices to move boundary vertices among adjacent partitions.

Schuetz and Caflisch [30] moves vertices among adjacent partitions based on the variation of modularity (ΔQ). The refinement algorithm fastgreedy (RFG) analyses the list of vertices in order of increasing degree and each vertex is moved to the adjacent partition with maximum ΔQ. The algorithm complexity is $O(m\ log\ n)$. Ye *et al.* [35] also presented an approach based on the maximization of modularity. Rotta and Noack [27] empirically analyzed heuristics of coarsening and refinement of graphs, and proposed an algorithm that prioritizes the modularity when perform any operation of edge merges (coarsening) or refinement vertices (uncoarsening).

Almeida and Lopes [1] presented a multilevel approach by optimization of modularity. In the partitioning phase is used the fastgreedy algorithm [5], however this approach does not use any refinement method, due to its objective is only to verify the performance of the fastgreedy algorithm in a multilevel context.

3. PROPOSED METHOD

In this section, we present a refinement algorithm based on neighborhood similarity, RSim, which explores social networks characteristics to face the refinement process. For this, we review the similarity measures used as the basis for our proposal, we then present the RSim algorithm and an analysis of its properties and complexity.

3.1 Similarity measures

In the context of complex networks, similarity measures quantify common characteristics between two vertices. When the similarity between vertices is based solely on network structure, it is called structural similarity. Structural similarity measures can be classified in different ways, such as the based on local or global information, refer to [20] for details.

Liben-Nowell and Kleinberg [18] and Zhou *et al.* [37] systematically compared a number of structural similarity measures on real networks. According to the authors, methods based on global information use all available information from the network, therefore it can provide higher accuracy than the measures based on local information. However, the global measures computation is very time-consuming and usually infeasible for large-scale networks. Alternatively, local measures use only information about pair of vertices, therefore are generally faster, but provide lower accuracy compared to the global ones. There are different similarity measures based on local information for example: Common Neighbors (CN), Salton (Sal), Jaccard (Jac), Sorensen (Sor), Hub Promoted (HP), Hub Depressed (HD), Leicht-Holme-Newman (LHN), Adamic-Adar (AA) and Resource Allocation (RA), summarized in Table 1.

Recently have been proposed some approaches which use other network informations such as the behavior of vertices in communities. Valverde-Rebaza and Lopes [32] proposed hybrid similarity measures based on community information. The WIC measure and the W-measures which are based on the relationship of belonging of the neighborhood of a pair of vertices to certain partitions.

The basic definition is the set of common neighbors, denoted by $\Lambda_{v,u} = \Gamma(v) \cap \Gamma(u)$. Consider that $\Lambda_{v,u} = \Lambda^W_{v,u} \cup \Lambda^I_{v,u}$, where $\Lambda^W_{v,u} = \{z \in \Lambda_{x,y} \mid x, y, z \in C_i\}$ is the set of within-community common neighbors (W) and the complement $\Lambda^I_{v,u} = \Lambda_{v,u} \ / \ \Lambda^W_{v,u}$ is the set of inter-community common neighbors (I). Therefore, $\Lambda^W_{v,u} \cap \Lambda^I_{v,u} = \emptyset$.

WIC measure. The WIC measure uses information of the common neighbors inter and intra-communities of the evaluated pair (v, u). Considering each vertex belonging to a unique partition, the WIC is defined by [34]:

$$S^{WIC}_{v,u} = \begin{cases} |\Lambda^W_{v,u}| & \text{if } \Lambda^W_{v,u} = \Lambda_{v,u} \\ |\Lambda^W_{v,u}| \ / \ |\Lambda^I_{v,u}| & \text{otherwise} \end{cases} \quad (2)$$

W-measures. Correspond to the reformulation of the local-similarity measures using information considering the common neighbors within-communities instead of compute all the common neighbors of the evaluated pair (v, u). The common neighbors within-community (W) capture the existent relations between pairs of vertices that belong to the same communities. However, the common neighbors inter-community (I) capture the existent relations between pairs of vertices that belong to different communities. Therefore, the similarity measures were reformulated using only the neighborhood set W, $\Lambda^W_{v,u}$ (summarized in Table 1).

Table 1: Local similarity measures and their corresponding W forms. Adapted from [32]

Local measure	W-measure
$S^{CN}_{v,u} = \lvert\Gamma(v) \cap \Gamma(u)\rvert = \lvert\Lambda_{v,u}\rvert$	$S^{CN-W}_{v,u} = \lvert\Lambda^W_{v,u}\rvert$
$S^{Sal}_{v,u} = \frac{\lvert\Lambda_{v,u}\rvert}{\sqrt{k(u)\times k(y)}}$	$S^{Sal-W}_{v,u} = \frac{\lvert\Lambda^W_{v,u}\rvert}{\sqrt{k(u)\times k(y)}}$
$S^{Jac}_{v,u} = \frac{\lvert\Lambda_{v,u}\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$	$S^{Jac-W}_{v,u} = \frac{\lvert\Lambda^W_{v,u}\rvert}{\lvert\Gamma(x)\cup\Gamma(y)\rvert}$
$S^{Sor}_{v,u} = \frac{2\lvert\Lambda_{v,u}\rvert}{k(x)+k(y)}$	$S^{Sor-W}_{v,u} = \frac{2\lvert\Lambda^W_{v,u}\rvert}{k(x)+k(y)}$
$S^{HP}_{v,u} = \frac{\lvert\Lambda_{v,u}\rvert}{min\{k(x),k(y)\}}$	$S^{HP-W}_{v,u} = \frac{\lvert\Lambda^W_{v,u}\rvert}{min\{k(x),k(y)\}}$
$S^{HD}_{v,u} = \frac{\lvert\Lambda_{v,u}\rvert}{max\{k(x),k(y)\}}$	$S^{HD-W}_{v,u} = \frac{\lvert\Lambda^W_{v,u}\rvert}{max\{k(x),k(y)\}}$
$S^{LHN}_{v,u} = \frac{\lvert\Lambda_{v,u}\rvert}{k(u)\times k(y)}$	$S^{LHN-W}_{v,u} = \frac{\lvert\Lambda^W_{v,u}\rvert}{k(u)\times k(y)}$
$S^{AA}_{v,y} = \sum_{z\in\Lambda_{v,u}} \frac{1}{\log k(z)}$	$S^{AA-W}_{v,y} = \sum_{z\in\Lambda^W_{v,u}} \frac{1}{\log k(z)}$
$S^{RA}_{v,y} = \sum_{z\in\Lambda_{v,u}} \frac{1}{k(z)}$	$S^{RA-W}_{v,y} = \sum_{z\in\Lambda^W_{v,u}} \frac{1}{k(z)}$

[3]A bisection is a 2-partition.

3.2 RSim

RSim is a refinement algorithm for k-partitions based on optimization of similarity measures between vertices. This algorithm explores social networks characteristics to do refinement process unlike others literature methods which are based on the general properties of complex networks. For this, we use as base the WIC and W-measures. However, these measures have been set in the link prediction context, whose score $S_{v,u}$ is calculated to any pair $(v,u) \notin E$. Once our goal is not predicting future connections, but improve the initial partitioning solution based on the existent connections, the similarity is calculated to any pair $(v,u) \in E$.

Algorithm 3.1 describes the RSim steps. In each step i of the uncoarsening phase, the algorithm moves boundary vertices among adjacent partitions. A score is calculated for each partition considering each one of their boundary vertices. The scoring is calculated with base on the average similarity between the vertex v and its neighborhood in the partition. Lastly, the vertex v is moved to partition with higher scoring.

Algorithm 3.1: Refiner

Input: graph, clustering
repeat
  foreach *boundary vertex* v **do**
    set ws;
    $C \leftarrow$ partitions where v has connection;
    foreach C_i *of* C **do**
      $ws(C_i) \leftarrow \frac{1}{k_{C_i}(v)} \sum_{v,u|u \in C_i} S_{v,u}$
    $C_i \leftarrow$ best cluster from ws;
    if $v \notin C_i$ **then**
      move vertex v to cluster C_i;
      update clustering;
until *no improve clustering*;
return *clustering*

The *clustering* indicates the original partitioning. The S indicates the similarity measure used, therefore RSim allows the use of different similarity measures but it is limited to measures based on the behavior of vertices in partitions. This behaviors allows us to calibrate the RSIM into ten variants using WIC and W-measures. Each similarity measure produces one RSim variant. For instance, RSim-CN uses the CN-W similarity measure.

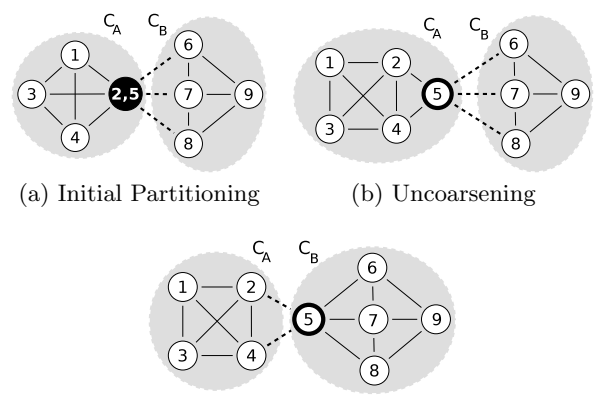
Considering that the algorithm performs operations only in boundary vertices and their operations are done in constant time using a vector mapping. For each boundary vertex v is computed the similarity to their neighborhood in each partition, i.e all neighbors of v will be visited independent of the number of partitions. In the worst case, in which all the vertices are boundary, all the edges will be checked. Therefore, the algorithm complexity does not depend on the number of partitions and it has a linear time of execution in relation with the number of edges, i.e. $O(|E|)$.

Figure 4 illustrates the refinement process for the boundary vertex 5 using RSim-CN. Given the initial partitioning $C = \{C_A, C_B\}$, where $C_A = \{1,2,3,4,5\}$ and $C_B = \{6,7,8,9\}$, illustrated by Figure 4(a), and after the uncoarsening phase, illustrated by Figure 4(b), we have:

$$ws(C_A) = \frac{1}{k_{C_A}(v)} \sum_{5,u|u \in C_A} S^{CN-W}_{5,u} = \frac{|\Lambda^{C_A}_{5,2}| + |\Lambda^{C_A}_{5,4}|}{k_{C_A}(v)} = \frac{|\{4\}| + |\{2\}|}{2} = \mathbf{1}$$

$$ws(C_B) = \frac{1}{k_{C_B}(v)} \sum_{5,u|u \in C_B} S^{CN-W}_{5,u} = \frac{|\Lambda^{C_B}_{5,6}| + |\Lambda^{C_B}_{5,7}| + |\Lambda^{C_B}_{5,8}|}{k_{C_B}(v)} = \frac{|\{7\}| + |\{6,8\}| + |\{7\}|}{3} = \mathbf{1.33}$$

as $ws(C_B) > ws(C_A)$, the vertex 5 is moved from partition C_A to C_B. Thus, $C_A = \{1,2,3,4\}$ and $C_B = \{5,6,7,8,9\}$, Figure 4(c).



(a) Initial Partitioning (b) Uncoarsening

(c) Refinement

Figure 4: Example of the refinement process using RSim with CN-W. During the refinement process the vertex 5 is moved from the partition C_A to C_B

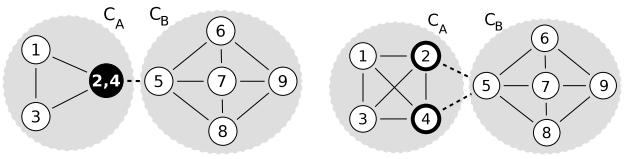
RSim has numerous variants based on the set of common neighbors, it is possible that all of the lead to the same decisions. For instance, Table 2 shows values obtained by three RSim variants to refine the vertex 5. In this case, all variants have moved the vertex 5 to C_B partition.

Table 2: Values by three RSim variants to refine the vertex 5 from $sV = \{2,5\}$ (Figure 4) and refine the vertices 2 and 4 from $sV = \{2,4\}$ (Figure 5)

Variant	$sV = \{2,5\}$		$sV = \{2,4\}$			
	refining 5		**refining** 2		**refining** 4	
	$w(C_a)$	$w(C_b)$	$w(C_a)$	$w(C_b)$	$w(C_a)$	$w(C_b)$
RSim-CN	1.00	**1.33**	**2.00**	0.00	**2.00**	0.00
RSim-HP	0.25	**0.38**	**0.66**	0.00	**0.66**	0.00
RSim-HD	0.20	**0.26**	**0.55**	0.00	**0.55**	0.00

Analogous, Figure 5 illustrates another scenario in which $sV = \{2,4\}$. In this case, the three variations also have

take the same decisions to refine the vertices 2 and 4, as shown in Table 2. We can conclude from the information in this table that the variants of RSim have produced the same solutions, any measure of quality of partitioning will show equalize performance for these variants.



(a) Initial Partitioning (b) Refinement

Figure 5: Example of the refinement process using RSim. During the refinement process the vertices 2 and 4 remain in the partition C_B

As stated previously, our algorithm explores social networks characteristics to perform the refinement process. For this, we use an approach based on neighborhood and degree. This characteristic are relevant at networks characterized by high clustering coefficient (presence of cycles with three vertices) and assortativity (tendency of vertices to connect with other vertices with similar degrees). In addition, our proposal is fast and feasible for large-scale networks.

4. EXPERIMENTS

We evaluated the RSim algorithm performance in twelve real networks from different domains: information networks, technological networks, biological networks, and social networks. The selected networks are commonly used in the research literature. Their topological characteristics are summarized in Table 3.

We evaluated ten RSim variants. Each variant uses a different similarity measure (WIC or W-measures). We have compared RSIM with two methods from the research literature, KK [15] and RFG [30, 35, 27]. Furthermore, in order to evaluate the efficiency of these algorithms we used a baseline (also called no-refinement) that performs pure multilevel partitioning (HEM+fastgreedy), i.e. does not undertake the refinement process. The baseline method is described in detail in [1].

The algorithms were implemented in `C++` and compiled with `GCC 4.7.3` and Linux Ubuntu `kernel 3.8.0-34-generic`. The structure of data is based in the platform LP-made [19]. Our experiments were performed on a computer with an `Intel(R) Core(TM) i5-2430M` processor CPU `@ 2.40GHz` and `4 GBytes` of memory.

Figure 6 summarizes our methodology with a flowchart. The experiments performed three steps: (i) the original graph G_0 is reduced recursively using the HEM algorithm, (ii) the coarse graph G_N is partitioned using the fastgreedy algorithm [5], (iii). the partition of the graph G_N is projected back to the intermediates levels and refined by an refinement algorithms. This step is repeated individually for each evaluated refinement algorithm.

Table 4 summarizes the accuracy results, measured by modularity on the twelve networks. First, we analyze the difference between RSim and two refinement algorithms (KK and RFG). Second, we analyze the statistical performance of all the analyzed refinement algorithms. The highest accuracies are highlighted in bold.

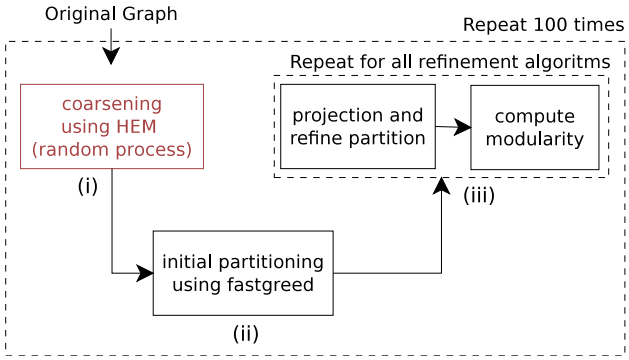


Figure 6: Flowchart of our experimental methodology

We observed that the RSim algorithm has the best performance in seven of the twelve evaluated networks. Also, we observed that the RSim performance is better in networks with a high clustering coefficient (**C**) and assortativity (**r**), as observed in AL, ZK, DBLP, NS, IM, HT and AP networks.

Many real networks have a high clustering coefficient. This concept can be generalized by assuming that object similarity behaves transitivity, i.e most of the neighbors of a vertex are neighbors of each other. Based on this intuition, RSim-CN and RSim-WIC maximize the proportion of pairs of vertices linked with one another among all the neighbors of a vertex, i.e. maximize the number of cycles of length three, therefore, these variants have best performance in networks with a high clustering coefficient, as observed in AL and DBLP networks. The other RSim variants have best performance in networks with high assortativity coefficient, such as NS, MI, HT and AP networks. Alternatively, RSim is less efficient in networks with lower clustering and assortativity coefficient.

To analyze the difference between refinement algorithms, we use a post-hoc test [6]. The results of Table 4 and analysis is shown in two critical difference diagrams in Figure 7. In the diagram of Figure 7(a), we analyze the refinement algorithms in a general context, i.e. using all networks. In the diagram of Figure 7(b), we analyze the top five refinement algorithms and baseline by the overall ranking obtained by diagram of the Figure 7(a) for social networks. The critical difference (CD) is shown in the top of each diagram. In the axis of each diagram is plotted the average ranks of the refinement algorithms analyzed. In each axis, the lowest (best) ranks are in the left side. Algorithms that have no significant difference are connected by a black continuous line.

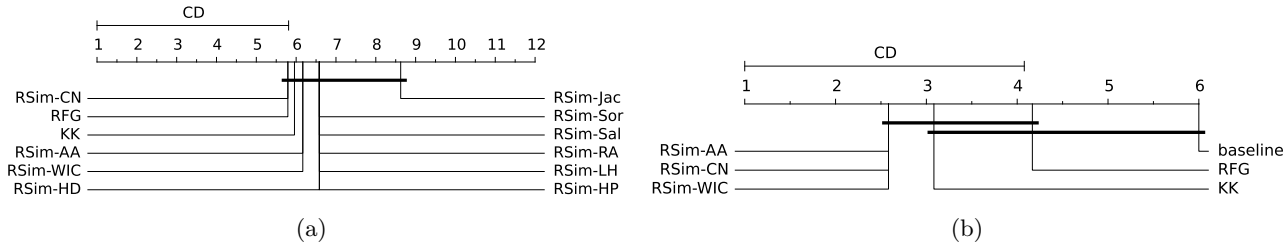
In the diagram of Figure 7(a) although there is no significant difference between these measures, observe that the RSim-CN, RFG, KK, RSim-AA and RSim-WIC have the best average rank. We noticed that two groups of RSim variants have the same performance. The first group is formed by RSim-WIC and RSim-AA. The second group is formed by RSim-HD, RSim-HP, RSim-Sor, RSim-Sal, Rsim-LH and Rsim-RA. Lastly, RSIM-Jac have the worst overall average rank. In the diagram of the Figure 7(a) also there is no significant difference between the top five algorithms, however observe that the RSim variants have a better average

Table 3: The basic topological features of twelve networks. Where $|V|$ and $|E|$ are the number of vertices and links. C and r are clustering and assortative coeficient, respectively. H denotes the heterogeneity degree

Domain	Nets	Acronym	$\|V\|$	$\|E\|$	C	r	H
Technological	Airline [3]	AL	332	2126	0.7494	-0.2079	3.4639
	Power [24]	PW	4941	6594	0.0801	0.0034	1.4504
	Router [31]	RT	5022	6258	0.0116	-0.1384	5.5031
Biological	Yeast [24]	YT	2362	7182	0.2443	-0.0587	2.7643
Information	Political Blogs [24]	PB	1224	16716	0.3203	-0.2211	2.9749
	Industry [7]	ID	2189	11666	0.3297	0.1842	3.4122
Social	Zachary Karate [24]	ZK	34	78	0.5879	-0.4756	1.6933
	DBLP [11]	DB	1011	5754	0.8677	0.0651	2.5485
	Imdb [21]	IM	1441	20317	0.5843	0.3492	2.0982
	NetScience [24]	NS	1461	2742	0.6937	0.4616	1.8486
	High-energy theory [24]	HT	8361	7875	0.2939	0.3402	2.3057
	Astrophysics [24]	AP	16706	121251	0.2355	0.4305	3.0946

Table 4: Accuracy measured by modularity on twelve networks for ten RSim variants, RFG, KK, and the baseline. Each modularity value is obtained by averaging over 100 times

Algorithm	Network											
	AL	PW	RT	YT	PB	ID	ZK	DBLP	NS	IM	HT	AP
RSim-AA	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	**0.6522**	0.6243
RSim-CN	**0.3165**	0.9165	0.8650	0.6869	0.4218	0.4801	0.3560	**0.9130**	**0.9553**	0.5912	0.6479	0.6128
RSim-HD	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	0.6479	0.6243
RSim-HP	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	0.6479	0.6243
RSim-Jac	0.2930	0.9144	0.8599	0.6860	0.4151	0.4592	**0.3828**	0.9117	0.9548	**0.6474**	0.6479	**0.6471**
RSim-LH	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	0.6479	0.6243
RSim-RA	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	0.6479	0.6243
RSim-Sal	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	0.6479	0.6243
RSim-Sor	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	0.9128	**0.9553**	0.5912	0.6479	0.6243
RSim-WIC	0.3160	0.9165	0.8650	0.6869	0.4218	0.4801	0.3553	**0.9130**	**0.9553**	0.5912	**0.6522**	0.6128
RFG	**0.3165**	0.9301	0.8840	0.6920	0.4236	**0.4871**	0.3667	0.9123	0.9549	0.5869	0.6429	0.6120
KK	0.2632	**0.9311**	**0.8870**	**0.7081**	**0.4257**	0.4693	0.3798	0.9128	0.9459	0.6385	0.6422	0.6128
baseline	0.2884	0.9228	0.8611	0.6607	0.4204	0.4782	0.3483	0.9113	0.9403	0.5785	0.5864	0.6058



(a) (b)

Figure 7: Critical difference diagrams for results from Table 4. In (a), we show the statistical analyses for all refinement algorithms in the twelve networks analyzed. In (b), is show statistical analyses for the top five refinement algorithms for only the six social networks analyzed (ZK, DBLP, NS, IM, HT, AP)

rank, which in turn is considerably larger than the average rank of the KK and RFG. This fact indicates that RSim surpasses the usual refinement algorithms in social networks and is competitive with the methods evaluated for general domains. The critical value for comparing mean ranking of two different algorithms at 95 percentile is 4.81 in the diagram of the Figure 7(a) and 2.49 in the diagram of the Figure 7(b). Mean rank differences above this value are significative.

The results of post-hoc values, shown in Figure 7(b), also indicated that there is significant difference between the baseline and RSim, as opposed, the algorithms RFG and KK have no statistically significant difference, providing evidence that the methods of literature have no clear advantage over the baseline.

Finally, we noted from results of Table 4, some RSim variants have exactly the same results. This ties as discussed in Subsection 3.2, occur due to the all RSim variants are based on the set of common neighbors.

Figure 8 shows the execution time of the refinement algorithms considering the size of the all analyzed networks. The upper x-axis shows the acronym of the evaluated network and the lower x-axis shows the size of the network by the number of edges.

The longest runtime of the RSim was 153.33 seconds, alternatively KK and RFG they took around of 700 seconds. Also, we notice that RSim variants have the lower growth curve than KK and RFG. This is due to RSim: performing only operations from boundary vertices. In addition the similarity measures used have a low computationally com-

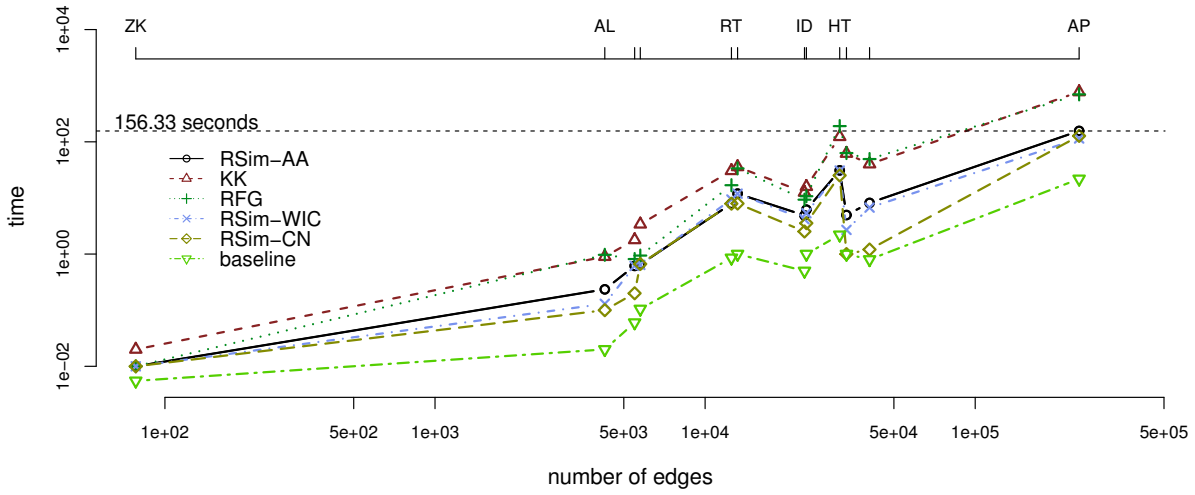


Figure 8: Runtime ratio the refinement algorithms considering the graph size at 50% reduction factor. The dashed line marks the longest runtime of the RSim. Baseline shows the runtime to projection partitions without perform the refinement process

plexity, unlike: RFG which uses the computationally expensive modularity increase (ΔQ) and KK which requires: 1. extra operations to check balancing and 2. additional computations to calculate the gain based on the degree of vertices. Therefore, independently of the type of network analyzed our proposal is faster than refinement algorithms compared.

5. CASE STUDY

In this section, we conducted a case study addressed the problem of ambiguous citations in scientific cooperation networks. The ambiguous citations problem affects the performance and quality of scientific data recovery from digital libraries (Springer[4], ACM[5], DBLP[6] and CiteSeer[7]). The ambiguous citations problem is where citations are incorrectly assigned and therefore gives undue credit to authors. This problem can be classified in different ways, such as: (i) **split citation** problem, in which citations of the same author appears under different name variations, such as "Ronald W. Williams" and "R. W. Williams"; (ii) **mixed citation** problem, in which different authors have the same name spellings, such as "Mohammed Zaki" (Al-Azhar University, Egypt) and "Mohammed Zaki" (Rensselaer Polytechnic Institute, USA). Recently, some studies used cooperation networks and partitioning algorithms to solving the split/mixed citation problem [11, 2]. However, the solution scalability and quality is a critical factor for large-scale bibliographic databases which may make some approaches unfeasible. For instance, the computer science citation databases DBLP contains approximately 2 million records. In order to increase scalability and accuracy in scientific cooperation networks, a multilevel partitioning algorithm may be appropriate. The idea is to match the multilevel strategy and the refinement algorithm that optimizes inherent characteristics of social networks.

In this experiment, we used the collection of authorship records extracted from the DBLP digital library. The registers were manually labeled using the authors information, the collection is available in [11]. We choose to treat the problem of the author "A. Gupta", which is similar to "A. Gupta" and "Amit Gupta" and related to "Ajay K. Gupta", "Amit Gupta" and "Anoop Gupta". The set of citations has 567 registers and 26 different authors.

The experiment was made in two steps: (i) the scientific cooperation network[8] is partitioned using a multilevel approach from different refinement algorithms, and (ii) each partition is processed individually using similarity between strings[9]. For this, we use the Levenshtein distance at 40% threshold. This value was selected due to the citation variants that take place mainly by abbreviations. For instance, two citations to the same author can be very different, such as "A. Gupta" and "Ashish Kumar Gupta". If there are more spelling mistakes (as "A. Gupta" and "A. Gopta") than abbreviations it is possible to set a lower threshold, for instance 10%.

Table 5 summarizes the results measured by modularity, F-measure and runtime on the DBLP network from the top five refinement algorithms. From entries highlighted in bold, we observe that RSim-CN and RSim-WIC have better performance towards from modularity and f-measure. This is due to DBLP having a high clustering coefficient (0.8677), consequently, it is more likely that common neighbors (cooperators) are connected among themselves. As we previously mentioned, using CN-W and WIC as refinement criteria is an efficient strategy in this scenario, once maximized

[4]http://www.springer.com/
[5]http://dl.acm.org/
[6]http://www.informatik.uni-trier.de/ ley/db/
[7]http://citeseer.ist.psu.edu/

[8]Two authors have a link if they producing a scientific writing together.
[9]Strings contained author's name.

the proportion of pairs of vertices linked with one another among all the neighbors of a vertex. Another observation is that RSim-AA have better f-measure than KK, although both have the same modularity, therefore, it is important to note that, the selection of the refinement algorithm will influence the final quality of the solution. Lastly, RSim variants surpasses KK and RFG as to the runtime.

Table 5: Efficacy and efficiency measured by modularity, F-measure and runtime in ambiguous citation problem in the DBLP network

Algorithm	modularity	F-measure	time/ms
RSIM-CN	**0.9130**	**0.6730**	06.81
RSIM-WIC	**0.9130**	**0.6730**	06.91
RSIM-AA	0.9128	0.6628	**06.79**
KK	0.9128	0.6501	12.02
RFG	0.9123	0.6501	11.02

6. SUMMARY AND CONCLUSION

This article focuses on the multilevel refinement process to improve the partitioning solution in a network. We propose a refinement algorithm based on optimization of similarity between vertices, called RSim. This algorithm exploits the tendency of vertices to be related to other vertices with similar characteristics. RSim can be applied to problems involving complex networks, but it can have a better performance in social networks.

Our proposal was compared with two algorithms from the research literature and one baseline method. The analysis on twelve real networks from different domains demonstrates that there was no single clear winner but our proposal achieve better accuracies. For networks which have a high clustering and assortativity coefficients, RSim surpasses the usual refinement algorithms. When we address the duplicate problem in scientific cooperation networks, the RSim outperforms methods based on minimum cut and modularity. In summary, this research produces evidences for efficiency of RSim in problems involving social networks.

Finally, the performance is an important matter when we work in large scale network, and the main advantage of multilevel approach is the reduction of the execution time. Based on this, our analysis shows that RSim is faster than the competing methods due to it uses operations with low computationally cost. Future work will be focus on the studies of heuristics for matching using different coarsening configurations.

7. ACKNOWLEDGEMENT

This work was partially supported by CNPq grant: 151836-/2013-2, FAPESP grants: 2011/22749-8, 11/20451-1 and 2013/12191-5 as well as by the CAPES funding agency.

8. REFERENCES

[1] L. J. Almeida and A. A. Lopes. An ultra-fast modularity-based graph clustering algorithm. *Proceedings 14th Portuguese Conference on Artificial Intelligence (EPIA) - Web and Network Intelligence Track*, pages 1–9, 2009.

[2] D. R. Amancio, O. N. Oliveira Jr., and L. d. F. Costa. Topological-collaborative approach for disambiguating authors' names in collaborative networks. *CoRR*, abs/1311.1266, 2013.

[3] V. Batagelj and A. Mrvar. Pajek datasets, 2006. visited on 2014-02-30.

[4] C.-E. Bichot. *A Partitioning Requiring Rapidity and Quality: The Multilevel Method and Partitions Refinement Algorithms*, pages 27–63. John Wiley & Sons, Inc., 2013.

[5] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004.

[6] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[7] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 53–62, New York, NY, USA, 1999.

[8] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, DAC '82, pages 175–181, Piscataway, NJ, USA, 1982.

[9] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[10] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

[11] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '04, pages 296–305, New York, NY, USA, 2004.

[12] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, Supercomputing '95, New York, NY, USA, 1995.

[13] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, Supercomputing '95, New York, NY, USA, 1995.

[14] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In *ICPP (3)*, pages 113–122, 1995.

[15] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96–129, 1998.

[16] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[17] B. Krishnamurthy and J. Wang. On network-aware clustering of web clients. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 97–110, New York, NY, USA, 2000.

[18] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the*

Twelfth International Conference on Information and Knowledge Management, CIKM '03, pages 556–559, New York, NY, USA, 2003.

[19] R. N. Lichtenwalter and N. V. Chawla. Lpmade: Link prediction made easy. *J. Mach. Learn. Res.*, 12:2489–2492, Nov. 2011.

[20] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):1150–1170, 2011.

[21] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 625–630, New York, NY, USA, 2003.

[22] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701, 2002.

[23] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[24] M. E. J. Newman, Dataset 2013. Available at `http://www-personal.umich.edu/~mejn/netdata/`; accessed 2013-30-03.

[25] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004.

[26] A. Noack and R. Rotta. Multi-level algorithms for modularity clustering. In *Proceedings of the 8th International Symposium on Experimental Algorithms*, SEA '09, pages 257–268, Berlin, Heidelberg, 2009.

[27] R. Rotta and A. Noack. Multilevel local search algorithms for modularity clustering. *J. Exp. Algorithmics*, 16:2.3:2.1–2.3:2.27, 2011.

[28] L. A. Sanchis. Multiple-way network partitioning. *IEEE Trans. Comput.*, 38(1):62–81, 1989.

[29] L. A. Sanchis. Multiple-way network partitioning with different cost functions. *IEEE Trans. Comput.*, 42(12):1500–1504, 1993.

[30] P. Schuetz and A. Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Phys. Rev. E*, 77:046112, 2008.

[31] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *SIGCOMM Comput. Commun. Rev.*, 32(4):133–145, 2002.

[32] J. Valverde-Rebaza and A. A. Lopes. Link prediction in complex networks based on cluster information. In *Advances in Artificial Intelligence - SBIA 2012*, volume 7589, pages 92–101. Curitiba, PR, Brazil, 2012.

[33] J. Valverde-Rebaza and A. A. Lopes. Exploiting behaviors of communities of twitter users for link prediction. *Social Network Analysis and Mining*, 3(4):1063–1074, 2013.

[34] J. C. Valverde-Rebaza and A. de Andrade Lopes. Structural link prediction using community information on twitter. In *CASoN*, pages 132–137. IEEE, 2012.

[35] Z. Ye, S. Hu, and J. Yu. Adaptive clustering algorithm for community detection in complex networks. *Phys. Rev. E*, 78:046115, 2008.

[36] Z. Yin, M. Gupta, T. Weninger, and J. Han. A unified framework for link recommendation using random walks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '10, pages 152–159, Washington, DC, USA, 2010.

[37] T. Zhou, L. Lü, and Y.-C. Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.