Visualization to Assist Interpretation of the Multilevel Paradigm in Bipartite Graphs

Diego S. Cintra, Alan Valejo, Alneu A. Lopes and Maria Cristina F. Oliveira

Istituto de Ciências Matemáticas e de Computação (ICMC), University of São Paulo (USP), CP 688, São Carlos SP, 13560-970, Brazil

Keywords: Visualization, Bipartite Graphs, Multilevel Paradigm, Graph Summarization.

Abstract: Multilevel methods refer to a general framework for solving optimization problems in large graphs considering a hierarchy of contracted representations of the target graph. A recent extension to bipartite graphs has been introduced and successfully employed in diverse applications, but experience suggests the method is highly susceptible to the choice of vertex matching strategy for graph contraction and on whether the super-vertices are relevant generalizations to the problem addressed. Although the flexibility in obtaining contracted representations of an original graph is a potential advantage, appropriate choice and parameterization of the contracting algorithms is challenging. Experts would benefit from solutions capable of assisting them in assessing alternatives and making informed decisions. In this work we describe a visualization system that creates an interactive graphical representation of a multilevel graph hierarchy obtained as a result of executing a multilevel method on bipartite graphs. We provide illustrative case studies showing the proposed visualization can support algorithm developers in inspecting and interpreting how different parameter choices in the coarsening stage impact the resulting multilevel hierarchies.

1 INTRODUCTION

Multilevel methods provide a framework for solving optimization problems in large graphs (Walshaw, 2004). They gradually reduce a large scale graph by coarsening it into a hierarchy of successively smaller graphs; employ a target algorithm to solve the problem in the smallest graph instance and project the solution thus obtained backwards into the hierarchy of increasingly complex graph models, up to the original one. An approximate solution may be thus obtained in situations where executing the target algorithm directly on the original graph might be unfeasible. This general framework has been instantiated in a variety of combinatorial problems in graphs, e.g., sparse matrix factorization (Gupta et al., 1997), graph partitioning (Padmavathi and George, 2014) and graph drawing (Hachul and Jünger, 2004). Solving a problem on a small graph requires searching a reduced solution space rather than the possibly cost-prohibitive solution space associated with the full graph. Thus, a solution is computed at a reduced cost and then generalized to the original graph. The quality of the final solution depends both on the quality of the initial solution and the generalization capability of the graph contraction strategy.

Originally defined on homogeneous networks, in which the vertices represent entities of a single type, the multilevel approach has recently been extended to bipartite graphs and networks (Valejo et al., 2018). Vertices in bipartite graphs represent two types of entities, so that the vertex set is split into two non-overlapping subsets and edges can only connect vertices in different subsets. Such graphs occur frequently in data analysis scenarios, e.g., document-word and protein-ligand networks are two illustrative examples in connection with relevant real-world problems (Rossi et al., 2016; Jeong et al., 2000).

The framework introduced by Valejo et al. (2018) to support the multilevel approach on bipartite graphs is powerful, conceptually simple and highly flexible, with potential applicability in a wide variety of data analysis problems. Yet, successful application of multilevel strategies involves difficult choices, e.g., the choice and parameterization of vertex matching algorithm required for coarsening may strongly impact the method’s performance and solution quality. Making informed choices when handling novel applica-
tion problems is particularly challenging. This work addresses this gap by introducing a visualization solution aimed at supporting algorithm developers in inspecting and interpreting the graph hierarchy resulting from an execution of the multilevel method on bipartite graphs. Our solution relies on a novel graphical metaphor for depicting a multilevel bipartite graph hierarchy, which is the key component of an interactive interface implemented as a front-end to the aforementioned framework.

This paper is organized as follows. The multilevel strategy is detailed in Section 2, and related work is briefly reviewed in Section 3. In Section 4 we describe the components of the visualization system developed and in Section 5 we present case studies illustrating how it can support developers interpreting the behavior of the multilevel method and understanding the implications of alternative choices. Concluding remarks are presented in Section 6.

2 MULTILEVEL METHOD ON BIPARTITE GRAPHS

A graph \( G = (V, E) \) is defined by a set of vertices \( V \) and a set of edges \( E \) that indicate relationships between pairs of vertices. The set \( V \) in a bipartite graph \( G = (V_1, V_2, E) \) is split into two disjoint subsets (layers) \( V_1 \) and \( V_2 \), \( V_1 \cap V_2 = \emptyset \), and edges connect vertices in different layers (Bondy et al., 1976). Edges or vertices may have weights or other attributes. Vertices \( u \) and \( v \) are neighbors (adjacent) if edge \((v, u)\) \( \in \ E \). The degree of \( v \in V \), \( d_G(v) \), is equal to the total weight of its adjacent edges. The vertex \( h \)-hop neighborhood consists of the vertices in set \( \Gamma_h(v) = \{w \mid \text{there is a path of length } h \text{ between } v \text{ and } w\} \) (Valejo et al., 2018).

Figure 1 illustrates the stages of a multilevel method applied to a bipartite graph. Initially a coarsening algorithm creates a sequence of simplified (coarsened) versions of the input graph at gradually increasing contraction levels. Coarsening comprises the steps of matching, which selects which vertices will be collapsed; and contraction, which builds the reduced representation, collapsing the matched vertices and their incident edges into so-called super-vertices and super-edges, respectively. A super-vertex is called a successor of its originating vertices, which are themselves called predecessors of the super-vertex. An initial solution to the target problem is computed on the coarsest graph instance. In the final uncoarsening stage this initial solution is projected backwards and refined through the intermediate sequence of coarsened graphs, until obtaining the final solution in the original input graph.

The vertex matching algorithm is a key component of an effective multilevel solution. Inadequate matchings may yield poor initial solutions, or poor performance of the target algorithm, perhaps both. Formally, a matching \(^1 M = \{V_i\}_{i = 0}^k \) is a partition of \( V \) into a set of \( K \) non-empty and disjoint subsets \( V_i \in V \), \( 1 \leq |V_i|, \forall i \) and \( i \neq j \Rightarrow V_i \cap V_j = \emptyset \). A vertex \( u \in V_i \) is called matched, otherwise, if \( \exists v \in M \mid u \in V_i, u \) is called unmatched. Algorithms rely on a user-provided function to assess vertex similarity, e.g., Common Neighbors (Watts, 2004) considers the pair of vertices with more neighbors in common as the most similar; Adamic Adar (Adamic and Adar, 2003) relies on a logarithm function to define a "relevance factor" in assessing vertex similarity; Preferential Attachment (Newman, 2001) considers the adjacent vertex of highest degree as the most similar.

Many matching algorithms have been developed relying on different policies to establish how vertices will be collapsed. We briefly review two recent algorithms targeted at bipartite graphs, namely GMb (Greedy Matching for bipartite graphs) (Valejo et al., 2018) and MLPb (Multilevel Label Propagation for bipartite graphs) (Valejo et al., 2019a). Both take as input parameters the initial graph \( G_0 \); the layers to be coarsened, i.e., if layer \( V_1 \), layer \( V_2 \) or both; and the similarity function \( S(u, v) \), \( u, v \in V \). Input parameters are informed with indexes 1 or 2 to indicate they refer respectively to layer \( V_1 \) or \( V_2 \).

GMb matching merges vertices pairwise relying on a priority queue of candidate pairs \( v, u \) wherein \( \Gamma_2(v) = u \), organized according to their similarity as computed by the given function \( S_j(u, v), j \in \{1, 2, \ldots, \} \), so that highly similar pairs are matched first. Addi-

---

\( ^1 \)Note, in the mathematical discipline of graph theory, a matching is defined as a set of independent edges.
tional inputs are the number of coarsening levels \( l_j \) and the graph reduction factors \( r_{f_j} \). For instance, setting \( l_1 = l_2 = 3 \) and \( r_{f_1} = r_{f_2} = 0.3 \) creates a 3-level graph hierarchy \( G_1, G_2, G_3 \) with each graph layer at a level \( k \) having 30% of its vertices paired into super-vertices, relative to the previous level \( k - 1 \). A maximum reduction factor of 50% is attained in a single iteration. Coarsening large graphs will require many iterations, incurring in high processing and memory costs. Moreover, matching quality tends to degrade towards the final iterations, as vertices considered for matching later are less likely to find good candidate pairs.

\( \text{MLPb} \) (Multilevel Label Propagation for bipartite graphs) has been introduced to overcome these limitations. It relies on label propagation to collapse vertex groups, rather than pairs. Every vertex is initially assigned a unique label. At each iteration, each vertex updates its label with the most frequent label in its neighboring vertices. Intuitively, a densely connected group of vertices will converge to a single dominant label, and upon convergence vertices with the same label will be collapsed into a single super-vertex. Besides \( G_0 \) and \( S_j(u,v) \), \( \text{MLPb} \) takes as input parameters the desired layer sizes in the coarsest graph, \( \zeta_j \), and a maximum number of iterations \( T \). E.g., given \( G_0 \) with \( |V_1| = |V_2| = 200, |E| = 10,000 \) and setting \( \zeta_1 = \zeta_2 = 20 \) \( \text{MLPb} \) will attempt to execute \( T \) iterations to create \( G_1 \) with \( |V_1| = |V_2| = 20 \). If this is not feasible, it will coarsen \( G_1 \) to yield a new graph \( G_2 \) and again execute up to \( T \) iterations. The process continues until the target sizes are met, yielding a multilevel graph hierarchy.

The bipartite multilevel framework \( \text{MOB} \) (Valejo et al., 2018) has been developed in Python as a flexible solution that admits plugging alternative matching algorithms and can be instantiated to solve multiple categories of problems. It provides the backend to our multilevel visualization front-end, called \( \text{MOBViewer} \). The aim is to facilitate interpretation of the behavior and limitations of matching algorithms and the impact of parameter choices on the outcome of a multilevel process applied to a bipartite network.

3 RELATED WORK

Multilevel methods have been applied, for instance, to reduce the computational cost of computing node-link layouts (Harel and Koren, 2000; Gajer and Kobourov, 2001; Hachul and Jünger, 2004), and an experimental evaluation of their usage in association with energy-based layout algorithms has been reported (Bartel et al., 2011).

Wong et al. (2009) describe a visualization solution which relies on the hierarchy of coarsened graphs displayed in interactive node-link views, which users can browse when executing analysis tasks. Providing graph views at varying coarsening levels addresses the visual clutter typical of node-link layouts. Indeed, whilst node-link representations are intuitive to convey global topological structures (Ghoniem et al., 2004), rendering large graphs is computationally expensive and may yield highly cluttered views on which exploratory tasks can be severely impaired (Von Landesberger et al., 2011).

Visual aggregation of nodes and edges is often employed to mitigate clutter. Edge bundling is possibly the best established solution for implicit edge aggregation (Lhuillier et al., 2017), whereas vertex aggregation strategies are usually explicit and domain-driven. Most solutions rely on clustering or community detection to group vertices into meta-vertices, super-vertices, clusters or communities (Von Landesberger et al., 2011), resulting in a hierarchical representation useful for visualization purposes.

Another category of related work comprises recent solutions for exploratory visualization of data modeled as large scale bipartite graphs. Several systems employ the biclustering algorithm (Heinrich et al., 2011) to generate vertex aggregations in this context (Zhao et al., 2018; Xu et al., 2016; Steinbock et al., 2018). Yet, other approaches are possible, e.g., the system WAOW-Vis (Pezzotti et al., 2018) adopts a hierarchical dimensionality reduction technique to create hierarchical representations of bipartite graphs and users may interact to expand a particular region.

The previous contributions deal with issues related to visualization of large graphs, focusing mostly on supporting data analysis. Our contribution is different in two ways. First, we deal with the problem of depicting a hierarchy of graphs resulting from coarsening in the context of the multilevel method. Second, we aim at assisting developers of multilevel solutions in assessing the effect of distinct choices of coarsening strategy. We are not aware of previous efforts of developing interactive visualizations to improve interpretation of the multilevel method or to support navigation in multilevel graph hierarchies, for the case of either homogeneous or bipartite graphs. Our goal is to investigate to which extent a visualization of the multilevel graph hierarchy can enhance user interpretation of the graph generalization process yielded by the multilevel method, in the particular case of bipartite graphs.
4 MOBViewer

Figure 2 shows MOBViewer's graphical metaphor for a multilevel graph with four hierarchical levels, where a vertex has been user selected (outlined in red). Vertices and super-vertices are shown as circles of varying sizes, with circle size proportional to the number of predecessors of the super-vertex. The two innermost circle rows depict the layers of the coarsest graph, while moving outwards from the central rows, the outer rows represent the remaining hierarchy of intermediate levels for each layer. The two outermost rows, rendered in gray, depict the layers of the input graph. By convention, the hierarchy relative to layer \( V_1 \) is displayed in the bottom area, whereas the hierarchy relative to layer \( V_2 \) is shown at the top.

Edges are rendered on-demand, upon vertex selection: once the user selects a vertex its corresponding circle is outlined in red and its incident edges are rendered. This choice has the benefit of avoiding line clutter and overdrawing. There are two types of edges: adjacency edges, rendered in blue as full lines, and hierarchy edges, rendered as green dotted lines. The first define the actual graph topology, i.e., how vertices in layers \( V_1 \) and \( V_2 \) are connected. The second show a vertex connection to its successor or predecessor (super-)vertices in the multilevel hierarchy. Adjacency edges are only displayed at the coarsest graph level, i.e., between the two inner rows/layers. Since graph layers are stacked on top/bottom of each other, the vertices are lined-up placing all predecessors of a super-vertex side-by-side, which prevents edge crossing and allows users to easily track which vertices have been grouped into a super-vertex. The complete thread of edges connecting a selected vertex to its predecessor/successor vertices is shown, with the circles depicting vertices in the hierarchical path outlined in green; and the circles depicting adjacent vertices to the selected one outlined in blue. For weighted graphs the weights of adjacency edges are mapped to color intensity, so darker shades of blue indicate heavier edges.

MOBViewer has been implemented as a client-server web application, with the back-end running MOB and the front-end running the visualization module. We used Javascript for implementation, with Node.js on the server-side; Express.js, a middleware for server-client data routing and communication; Three.js to render the graph metaphor and D3.js (Bostock et al., 2011) functions for visualization. MOBViewer is available to interested readers at https://github.com/diego2337/MOBViewer.

5 RESULTS

We illustrate how MOBViewer can support developers of multilevel solutions. The software has been executed on a Linux Mint 18.2 with 32GB RAM, Intel Core i7-3770 3.40 GHz and video card GeForce GTX 660. We considered scenarios of interest to developers of multilevel solutions, and report results from an empirical investigation of the following questions representative of issues faced by users of matching and coarsening algorithms:

Q1 How does the choice of similarity function affect preservation of the community structures in the multilevel hierarchy?

Q2 Is it possible to identify at which point in coarsening community structures start to degrade, i.e., super-vertices are created with predecessors vertices from multiple communities?

Q3 How does parameter reduction factor impact the construction of the multilevel hierarchy?

Q4 Is the matching algorithms’ capability of preserving community structures affected by the number of communities in the input graph?

MOBViewer allows inspecting how the matching algorithms behave with distinct parameter settings, and thereby comparing the outcome of different choices and assessing the convenience of adopting alternative strategies. Indeed, the ability to ‘see’ a multilevel graph hierarchy and its properties is an important facility for developers. Here we present illustrative scenarios of how each question could be investigated, taking as example a particular combination of algorithm, parameter settings and input graph. Of course, these questions are just a sample of what might be asked and they could be further investigated considering other settings.

In order to study the method’s behavior on graphs with diverse properties we synthesized weighted bi-
partite graphs with arbitrary community structures and topological properties using a benchmarking tool called BNOC (Benchmarking weighted Bipartite Networks with Overlapping Community structure) (Valejo et al., 2019b). Table 1 summarizes the synthetic networks considered. Graphs $SG_1$ and $SG_2$ are sparse with an equal number of communities in both layers, whereas graph $SG_3$ is denser with 3 communities in layer $V_1$ and 5 in layer $V_2$. Graphs $SG_3$ and $SG_5$ have vertex sets of the same size, but $SG_4$ is denser and the graphs differ in the number of communities in each layer.

Table 1: Synthetic graphs used in case studies.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Vertices</th>
<th>Edges</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>Communities</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SG_1$</td>
<td>100</td>
<td>555</td>
<td>50</td>
<td>50</td>
<td>14.4</td>
</tr>
<tr>
<td>$SG_2$</td>
<td>1,000</td>
<td>6,603</td>
<td>500</td>
<td>500</td>
<td>7.7</td>
</tr>
<tr>
<td>$SG_3$</td>
<td>1,819</td>
<td>23,652</td>
<td>542</td>
<td>1,277</td>
<td>3.5</td>
</tr>
<tr>
<td>$SG_4$</td>
<td>15,000</td>
<td>189,234</td>
<td>10,000</td>
<td>5,000</td>
<td>15.15</td>
</tr>
<tr>
<td>$SG_5$</td>
<td>15,000</td>
<td>19,102</td>
<td>10,000</td>
<td>5,000</td>
<td>40.40</td>
</tr>
</tbody>
</table>

In order to answer questions Q1 and Q2 we considered three configurations of the multilevel method using $GMb$ matching with distinct similarity functions: “Common Neighbors” in configuration C.CN, “Adamic Adar” in C.PA and “Preferential Attachment” in C.CN. In the figures, glyph colors map the corresponding vertex communities; glyphs representing super-vertices may have multiple colors reflecting the community distribution of their predecessors.

Figure 3 shows the hierarchy of coarsened graphs obtained executing $MOb$ on $SG_2$ with the three configurations. The hierarchy yielded by configuration C.CN, Figure 3(a), includes a few super-vertices with predecessors from mixed communities, outlined in red. Configuration C.PA, Figure 3(b), did not preserve the community structure, which was best preserved with the settings of configuration C.CA, Figure 3(c). Equivalent analyses were conducted on $SG_1$ and $SG_3$ (not shown), with similar conclusions.

Addressing related question Q2, Figures 4(a) and 4(b) show two closer views of the multilevel hierarchies of graph $SG_1$ yielded by configurations C.PA and C.CN, respectively. In the multilevel graph in Figure 4(a) one notices vertices from distinct communities merging into a single super-vertex at the second hierarchical level, see the zoomed-in views. However, in the multilevel hierarchy computed with configuration C.CN, shown in Figure 4(b), super-vertices with mixed communities only appear at the final levels. Community structure was best preserved with Common Neighbors similarity, Figure 4(b).

In order to address question Q3 we executed the multilevel method again with the $GMb$ matching algorithm and the “Common Neighbors” similarity, with 4 distinct settings of parameters reduction factor and number of levels (applied to both layers), informed in Figure 5, which shows the corresponding coarsest graphs $SG_1$. The number of super-vertices is strongly affected by the choice of $rf$, i.e., as reduction factors decrease fewer super-vertices are created and they are more imbalanced in size; notice, for instance, the super-vertices in Figures 5(a) and (d). Figure 6 shows an equivalent analysis on graph $SG_2$ considering two extreme configurations. Unlike the coarsest graph in (a), where most super-vertices have predecessors originating from a single community, the graph in (b) has just 2 super-vertices (outlined in red) with predecessors from mixed communities. Small reduction factors imply in few super-vertices, very likely formed by predecessor vertices from multiple communities, impairing community preservation.

Question Q4 was investigated on $SG_3$, a sparse graph with 40 communities in each layer, again using the “Common Neighbors” similarity. We applied two different matchings, $MLPb$ with $ζ_1 = 3.900$, $ζ_2 = 1.200$ and $T_1 = 1$, and $GMb$ with $r_{f_1} = r_{f_2} = 0.5$ and $l_1 = l_2 = 3$ (settings chosen to obtain a coarsest level graph with layers of equivalent sizes to $MLPb$). Figure 7 shows the coarsest level graphs, which for the most part preserved the community structures in merging vertices. This example illustrates that both algorithms, despite adopting very distinct strategies, can preserve the community structures even in graphs with many communities. Other configurations were investigated, yielding similar results (not shown).

6 CONCLUSIONS

We introduced MObViewer, a visualization tool that supports interpreting the outcome of executing the multilevel method on bipartite graphs. It relies on a visual metaphor designed to depict multilevel bipartite graphs hierarchies computed with the MOb framework. Illustrative case studies confirmed its capability of conveying the behavior of a matching algorithm applied with distinct parameter settings. Thus, developers of multilevel solutions can quickly assess alternative configurations in order to interpret and understand the impact of choices of matching algorithm, similarity function, and reduction factors.

Scalability is a critical issue in MObViewer. In spite of using state-of-the-art technologies the current implementation does not scale to graphs with roughly over 20,000 vertices or 30,000 edges. Performance is also hindered on multilevel hierarchies with many intermediate coarsening levels. On-demand network processing could be considered to overcome scalability limitations on interactive visual data analytics sce-
Figure 3: Multilevel hierarchies resulting from coarsening $SG_2$ with $GM_b$ matching considering 3 distinct similarity functions $S$: community structures were best preserved using the Adamic Adar and Common Neighbors functions (a, b).

Figure 4: User-selected super-vertex in graph $SG_1$ and the thread of edges to its predecessors in 2 multilevel hierarchies obtained with 2 distinct choices of $S$. (a) shows super-vertices with predecessors from mixed communities at level 2 of the hierarchy; in (b) such vertices only appear at the coarsest level (zoomed-in views of the areas outlined in blue are shown).
Figure 5: Coarsest graphs of the multilevel hierarchies computed for $SG_1$ with 4 distinct parameter settings in $GMb$.

(a) $C1\{r f_1 = r f_2 = 0.4, l_1 = l_2 = 3\}$
(b) $C2\{r f_1 = r f_2 = 0.3, l_1 = l_2 = 5\}$
(c) $C3\{r f_1 = r f_2 = 0.1, l_1 = l_2 = 16\}$
(d) $C4\{r f_1 = r f_2 = 0.2, l_1 = l_2 = 7\}$

Figure 6: Coarsest graphs of the multilevel hierarchy obtained from graph $SG_2$ with two settings of $GMb$.

(a) $C.1\{r f_1 = r f_2 = 0.4, l_1 = l_2 = 4\} S = \text{Common Neighbors.}$
(b) $C.2\{r f_1 = r f_2 = 0.1, l_1 = l_2 = 20\} S = \text{Common Neighbors.}$

Figure 7: Coarsest level graphs after coarsening $SG_5$ with different matching algorithms and $S = \text{Common Neighbors.}$

(a) Coarsest graph from $MLPb$ matching with $\zeta_1 = 3.900, \zeta_2 = 1.200, T = 1$.
(b) Coarsest graph from $GMb$ matching with $r f_1 = r f_2 = 0.5$ and $l_1 = l_2 = 3$. 
narios. Rather than processing the full network in a batch-like operation mode, the system could build and process a sub-network reflecting a current user focus on-the-fly, where these sub-networks satisfying a user requirement would likely have manageable sizes. Additional interaction facilities could be incorporated, such as supporting interaction with selected levels of the multilevel hierarchy. Further validation on additional scenarios is also necessary.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support of FAPESP grants 2016/25107-0 and 2017/05838-3 and CNPq grants 134806/2016-6 and 301847/2017-7.

REFERENCES


