Detecting Communities in Biological Bipartite Networks

Paola Pesantez-Cabrera
School of EECS
Washington State University
Pullman, Washington 99164
p.pesantezcabrera@wsu.edu

Ananth Kalyanaraman
School of EECS
Washington State University
Pullman, Washington 99164
ananth@eecs.wsu.edu

ABSTRACT

Methods to uncover and extract community structures are required in a number of biological applications where networked data and their interactions can be modeled as graphs, and observing tightly-knit groups of vertices ("communities") can offer insights into the structural and functional building blocks of the underlying network. While classical applications of community detection have focused largely on detecting molecular complexes from protein-protein networks and other similar graphs, there is an increasing need for extending the community detection operation to work for heterogeneous data sets — i.e., networks built out of multiple types of data. In this paper, we address the problem of identifying communities from biological bipartite networks — networks where interactions are observed between *two different types* of vertices (e.g., genes and diseases, drugs and protein complexes, plants and pollinators). Toward detecting communities in such bipartite networks, we make the following contributions: i) we define a variant of the bipartite modularity function defined by Murata to overcome one of its limitations; ii) we present an algorithm (*b*iLouvain), building on an efficient heuristic that was originally developed for unipartite networks; and iii) we present a thorough experimental evaluation of our algorithm compared to other state-of-the-art methods to identify communities on bipartite networks. Experimental results show that our *b*iLouvain algorithm identifies communities that have a comparable or better quality (bipartite modularity) than existing methods, while significantly reducing the time-to-solution between one and three orders of magnitude.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms; I.5.3 [**Clustering**]: Algorithms, Similarity measures

General Terms

Algorithms

BCB'16, October 2–5, 2016, Seattle, WA, USA.
 ISBN 978-1-4503-4225-4/16/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2975167.2975177.

Keywords

Heterogeneous biological data, biological bipartite networks, graph algorithms, community detection

1. INTRODUCTION

The increasing identification and characterization of genes, protein complexes, diseases, and drugs have highlighted a need to incorporate *heterogeneity* while analyzing complex biological data. Identifying "community" structures that transcend data boundaries could provide new insights that may not be readily visible by examining only a specific data type in isolation. For instance, identifying a group of genes that have been implicated across a set of diseases could possibly reveal hidden links among seemingly different diseases or disease conditions, and in the process help identify new drugs and therapies [8]. Similarly, identifying active gene clusters across different subsets of brain regions could provide new insights into brain function [13].

Graph-theoretic representations offer a natural way to model networks built out of heterogeneous data. In this paper, we focus on *bipartite networks* as a way to model the interplay between two different data types. Bipartite networks are those which have two types of vertices such that edges exist only between vertices of the two different types. Some examples of biological bipartite networks include (but are not limited to) gene-disease [19], gene-drug [11], plants-pollinators [7], and host-pathogen [22]. Once modeled as a bipartite network, we can view the problem of identifying cluster structures between the two different data types as a problem of *community detection in bipartite networks*.

Given a graph, the goal of community detection is to partition the set of vertices into "communities" such that vertices that are assigned to the same community have a higher density of edges among them than to the vertices in the rest of the network. Community detection can be used to reveal hidden substructures within real world networks, without any known prior knowledge on either the number or sizes of the output communities.

Community detection is a well studied problem in literature [9]. However, the treatment of the problem on bipartite networks has been sparse. Because edges connect vertices of two different types, the classical definition of communities [18] does *not* directly apply. Instead, the notion of communities needs to be redefined so that a community of vertices of one type is formed on the basis of the strength of its shared connections to vertices of the other type (as shown in the example in Figure 1).

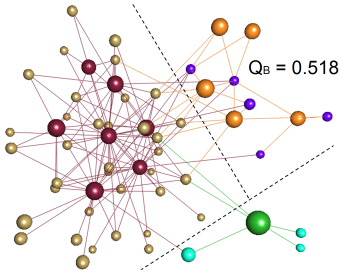

Figure 1: Communities detected by our *biLouvain* algorithm, in the *motten1982* bipartite network [7] of 13 spring wildflowers (larger nodes) vs. 44 insects visitors (smaller nodes). Nodes of the same color correspond to the same community. Q_B stands for bipartite modularity.

To unveil important new associations, evaluation of the goodness of a community-wise division of a bipartite network also becomes critical. To this end, a measure called *bipartite modularity* can be defined, extending the classical measure for unipartite networks [18]. Modularity is a statistical measure which calculates the difference between the observed fraction of intra-community edges to an expected fraction in an equivalent random graph — i.e., null model. For bipartite networks, multiple formulations of bipartite modularity have been proposed [1, 12, 16, 21]. Of these, Murata's definition is known to overcome some of the limitations of the other definitions (see Sections 2 and 4).

Contributions: We present a direct method to optimize a modified version of Murata's modularity (which we call *Murata+*). Specifically, the contributions are as follows:

- *Murata+ modularity:* We initially consider the classical definition of Murata's modularity [16] and discuss its suitability for community detection in Section 4. In the process of evaluation, we identify an inconsistency in the classical formula and subsequently propose a simple variant.
- *biLouvain algorithm:* We present an algorithmic heuristic for optimizing *Murata+* modularity in bipartite networks (Section 5). Our approach extends the Louvain algorithm [4], which is one of the most efficient and widely used heuristic for unipartite networks. Consequently, we call our algorithm *biLouvain*. As part of our algorithm, we provide ways to calculate the modularity gain resulting from vertex migrations — a step that constitutes the core of the Louvain heuristic.
- *Experimental evaluation:* We present a thorough experimental evaluation of our algorithm using both synthetic and real world networks and in comparison to other state-of-the-art methods (Section 6). Experimental results show that our algorithm identifies communities that have a comparable or better quality (bipartite modularity) than existing methods, while significantly reducing the time-to-solution between one and three orders of magnitude.

2. RELATED WORK

Community detection has been extensively studied in unipartite graphs [9]. Most of these algorithms use the modularity measure, as defined by Newman [18]. Multiple efforts have extended the classical definition of modularity to bipartite networks:

i) Guimerà *et al.* [12] defines bipartite modularity as the cumulative deviation from the random expectation of the number of edges between vertex members of the same bipartite community. The main weakness of this definition is that it focuses on connectivity from the perspective of only one vertex type.

ii) Barber [1] extends the scope to include connectivity information from both vertex types. However, this definition has a limitation of assuming a one-to-one correspondence between the communities from both vertex types — i.e., the number of communities should be equal on both sides.

iii) Murata's definition [16] overcomes the above limitations by not enforcing a one-to-one mapping between the communities of either side (see Section 4.1).

iv) Suzuki-Wakita [21] uses a "prominence factor" that averages the volume of connections to all communities.

Table 1: A comparison of bipartite community detection methods.

Feature	**DIRTLPAwb+ [3]**	**biSBM [14]**	***biLouvain***
Objective function	Barber's modularity	Maximum Likelihood	*Murata+ modularity*
Output determinism	No	No	*Yes*
Number of communities known *a priori*	No	Yes	*No*
Binary & weighted	Yes	Yes	*Yes*

While there is no general consensus on the modularity definition to use for bipartite networks, we chose to adopt the Murata's definition (with modifications - Section 4.2). Experimental results shown in Section 6 will demonstrate the high quality of results produced by this definition.

As for community detection in bipartite networks, there have been a handful of efforts so far. In 2010, Liu and Murata [15] proposed a way to detect communities in bipartite networks using label propagation. This approach was extended and improved by Beckett [3] in a tool called *DIRTLPAwb+*. In 2014, Larremore *et al.* [14] developed an approach (*biSBM*) that uses Stochastic Block model to maximize a likelihood function. This approach assumes that the number of output communities should be known *a priori*. Multiple executions of *DIRTLPAwb+* and *biSBM* on the same input could produce different outputs. In 2012, Nacher *et al.* [17] used a simulated annealing algorithm to maximize Guimerà's modularity and also calculated modularity using projections. The method proposed in *this* paper, *biLouvain*, compares to the above methods as summarized in Table 1.

3. BASIC NOTATION AND TERMINOLOGY

We will use $G(V_1 \cup V_2, E, \omega)$ to denote an undirected bipartite graph. Here, V_1 and V_2 represent the two sets of vertices, and E represents the set of edges such that each edge $e = (i, j) \in E$ is such that $i \in V_1$ and $j \in V_2$. Each edge e is also associated with a numerical weight $\omega(e)$. We will assume that the edge weights are non-negative values that reflect the strength of the relation between any two

vertices. The sum of the weights of all edges incident on a vertex i is said to be its *weighted degree* (denoted by $\gamma(i)$). Additionally, we use the term *binary networks* to describe networks whose edges are unweighted. In such cases, all edges that exist are assumed to have a unit weight.

Let n_1 and n_2 denote the number of vertices in V_1 and V_2 respectively, and m denote the number of edges. Let $M = \sum_e \omega(e)$. For sake of consistency, we use i's to denote vertices in V_1, and j's for vertices in V_2.

A *community* represents a subset of either V_1 or V_2. For ease of exposition, we use C's to denote communities taken from V_1 and D's to denote communities taken from V_2. Let $P_1 = \{C_1, C_2, \ldots, C_{k_1}\}$ denote a set of communities in V_1 such that it represents a partitioning of V_1. Similarly, let $P_2 = \{D_1, D_2, \ldots, D_{k_2}\}$ represent a set of communities in V_2 such that it represents a partitioning of V_2. Throughout this paper, we assume k_1 need *not* be equal to k_2.

We denote the community containing any vertex $i \in V_1$ as $C(i)$. Similarly, we denote the present community containing any vertex $j \in V_2$ as $D(j)$.

4. BIPARTITE MODULARITY

4.1 Murata's Bipartite Modularity

In what follows, we describe Murata's modularity [16] although using our own notation for convenience.

Given a pair of communities, $C \in P_1$ and $D \in P_2$, let $E_{C,D}$ denote the set of all edges that connect a vertex in community C with a vertex in community D, and let $e \in E_{C,D}$. Consequently, we define:

$$e_{C,D} = \frac{1}{2M} \sum_e \omega(e) \tag{1}$$

For binary networks, this quantity represents the fraction of all edges that connect communities C and D[1]. Note that by this definition, $e_{C,D} = e_{D,C}$. Also note that the term $2M$ corresponds to the sum of the weighted degree of all vertices — i.e., $2M = \sum_i \gamma(i)$. We will use the term a_C to denote the fraction of this term contributed by community C.

$$a_C = \frac{1}{2M} \sum_D e_{C,D} \tag{2}$$

Furthermore, we define the *co-cluster* of a community C be a community $D_C \in P_2$ to which C has the most concentration of its edges — i.e.,

$$D_C = \underset{D}{\operatorname{argmax}} \left(e_{C,D}\right) \tag{3}$$

Similarly, the *co-cluster* of a community D is defined as follows:

$$C_D = \underset{C}{\operatorname{argmax}} \left(e_{D,C}\right) \tag{4}$$

Definition 1. Given a bipartite graph $G(V_1 \cup V_2, E, \omega)$, and two sets of communities P_1 in V_1 and P_2 in V_2, Murata's bipartite modularity Q_B is defined as follows [16]:

$$Q_B = \sum_C (e_{C,D_C} - a_C \times a_{D_C}) + \sum_D (e_{D,C_D} - a_D \times a_{C_D}) \tag{5}$$

Intuitively, Murata's modularity is calculated by pairing every community from one side with a community on the other side that it has maximum connections to. The first term inside the two summations in Eqn. 5 corresponds to the fraction of such "intra-cocluster" edges. The second term inside each of the summations is the expected fraction of such edges in a randomly generated bipartite graph with an identical vertex degree sequence. As in Newman's modularity [18] for unipartite networks, the idea is to encourage a partitioning that maximizes intra-cocluster edges while discouraging a partitioning from grouping unrelated vertices together.

[1]The factor 2 in the denominator is a result of each edge getting counted twice — once in each direction.

4.2 *Murata+*: Proposed Bipartite Modularity

From Eqns. 3, 4 and 5 we make the following two observations:

Observation 1. If a community C picks a community D as its co-cluster (by Eqn. 3), D need *not* necessarily pick C (by Eqn. 4).

Observation 2. The statistical terms $a_C \times a_{D_C}$ and $a_D \times a_{C_D}$ are used in the final modularity calculation of Eqn. 5, but they are *not* used while picking the co-clusters (Eqns. 3 and 4).

Observation 1 implies that the co-cluster relationship is *nonsymmetric*. This relaxation is necessary to avoid a method-enforced one-to-one mapping between communities and their co-clusters. However, the relaxation could also lead to an undesirable effect of a lack of cohesion between communities and their co-clusters. For bipartite networks, we typically attempt to explain the grouping of a community based on its co-cluster. While one-to-one mapping would make it too restrictive for this purpose, it is also important *not* to make it excessively many-to-many. For most practical inputs, a middle ground is more desirable where the expected mapping remains closer to an one-to-one mapping. For instance, we can expect a strong (if not strict) two-way correlation between a set of genes and the set of diseases they impact.

Observation 2 indicates a matter of inconsistency because a community C picks a co-cluster solely based on the positive term, while the final modularity is calculated taking into account also the negative term. At best, this can lead to an underestimated value of modularity when Q_B is calculated. More importantly, we argue that the negative term is in fact essential as otherwise it could potentially lead to a scenario where a community and its co-cluster could be of vastly different sizes. This is shown in Figure 2.

To address the above inconsistency problem within the classical definition, we propose a variant of bipartite modularity by redefining the *co-cluster* equations as follows:

$$D_C = \underset{D}{\operatorname{argmax}} \left(e_{C,D} - a_C \times a_D\right) \tag{6}$$

Similarly, the *co-cluster* of a community D is defined as follows:

$$C_D = \underset{C}{\operatorname{argmax}} \left(e_{D,C} - a_D \times a_C\right) \tag{7}$$

The modularity expression to be used is identical to that of Eqn. 5.

It should be clear that this revised definition would make the choice of co-clusters consistent with the modularity calculation — i.e., fixing the problem with Observation 2.

In addition, the revised definition preserves the nonsymmetry property (Observation 1), while being better positioned than the classical definition to encourage one-to-one

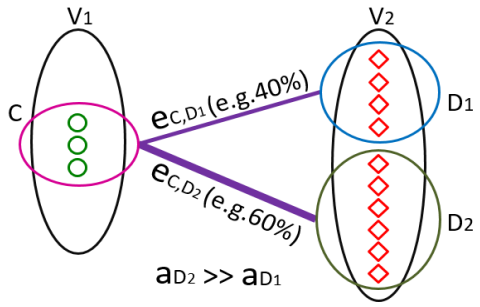

Figure 2: An illustrative example of a case where a community C has 40% of its edges connected to community D_1 and the remaining 60% of its edges connected to community D_2. Under this scenario, C has two choices for its co-cluster, D_1 and D_2, with e_{C,D_1} being only marginally smaller than e_{C,D_2} but the sizes of C and D_2 are vastly different as shown. In scenarios like this, ignoring the negative term may have the undesirable effect of picking a co-cluster that is significantly different in size. A better choice of co-cluster for C is D_1, not only because of its comparable size, but also because it is likely to lead to a better modularity.

mapping, wherever possible, without strictly enforcing it. This is because of the reduced degree of freedom that a community is likely to have (with the introduction of the negative term) when picking its co-cluster.

Henceforth, we refer to our revised version of the Murata's modularity as *Murata+* (Eqns. 5, 6, and 7), and use it as our primary objective function for detecting communities in a bipartite network.

5. *BILOUVAIN:* AN ALGORITHM FOR BIPARTITE COMMUNITY DETECTION

In this section, we present our *biLouvain* algorithm for community detection in bipartite networks. We adapt the widely used *Louvain* heuristic [4] to work for bipartite networks. While *biLouvain* follows the same algorithmic template provided by Louvain, it differs in the objective function (by using *Murata+*) and in the way all the key steps are computed, as will be elaborated below.

Like the original algorithm, *biLouvain* is a multi-phase, multi-iterative algorithm, where each *phase* is a series of *iterations*, and the algorithm moves from one phase to another using a graph compaction step. Within every iteration, each vertex makes a local decision on its community based on a net modularity gain function. The main steps of the *biLouvain* algorithm are as follows (see Figure 3):

1. Given a bipartite graph $G(V_1 \cup V_2, E, \omega)$, initialize a set of n_1+n_2 communities, where $n_1 = |V_1|$ and $n_2 = |V_2|$, by placing each vertex in its own community.
2. At every iteration, all vertices in V_1 and V_2 are scanned linearly (in an arbitrary order). For each vertex i:
   a) Acquire a list of its **candidate communities** for it to move to;
   b) Evaluate the **modularity gain** that would result from moving i from its current community to each of the candidate communities.
   c) Move vertex i to a candidate community that maximizes the modularity gain, only if that gain is positive (otherwise, no change).
3. A phase ends when the net modularity gain achieved between two consecutive iterations is negligible — i.e., below a certain threshold τ_i, which we refer to as the *iteration cutoff*.
4. Once a phase terminates, a new graph $G'(V_1' \cup V_2', E', \omega')$ is generated through a **compaction step**, which collapses each community to a vertex, and edges and their weights in the new graph corresponds to the strength of edges connecting any two communities.
5. The new compacted graph is input to the next phase (step 1). The algorithm terminates when any two consecutive phases result in a negligible modularity gain, defined by a threshold τ_p, which we refer to as the *phase cutoff*.

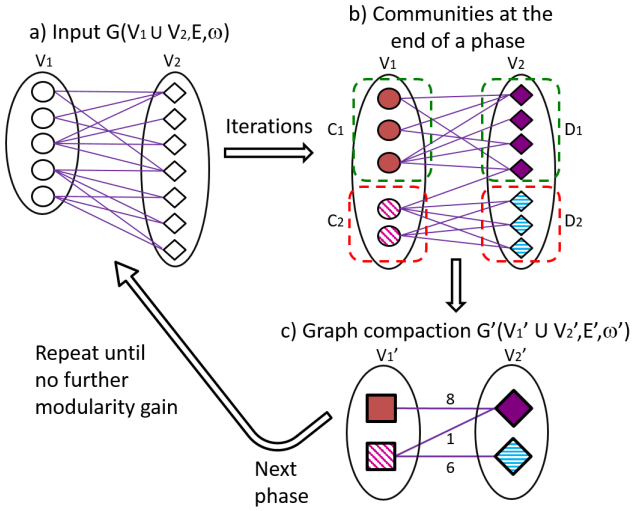

Figure 3: An example to illustrate the main steps of our algorithm for community detection: *biLouvain*. a) The input bipartite graph. Vertices of the two different types are shown in two different shapes. b) After a sequence of iterations, the biLouvain algorithm converges (based on net modularity gain) and a phase is completed. The figure shows the vertices and the communities they belong to at the end of the phase. In this simple example, communities C_1 and D_1 are co-clusters to one another, and C_2 and D_2 are co-clusters to one another. c) The graph is compacted by collapsing each detected community into a new "vertex" and collapsing inter-community edges between every pair of communities into new "edges" with corresponding weights. This compacted graph is input to the next phase until convergence.

In what follows, we describe how the key steps that are impacted by the bipartite structure are implemented in *biLouvain*. More specifically, the classical definition for candidate communities and their computation (step (2a)), the expression for calculating the modularity gain resulting from a vertex migration, and the algorithm to calculate the modularity gain (step (2b)) need to be redefined taking into account the bipartite structure.

5.1 Computing Candidate Communities

A *candidate community* of a vertex is a community to which that vertex can potentially migrate at any given iteration of the *biLouvain* algorithm, with a realistic chance of accruing a positive modularity gain.

For ease of exposition, we explain the process of computing candidate communities from the point of view of a vertex i in V_1. It should be easy to see that the same approach works for any vertex $j \in V_2$.

For a given vertex $i \in V_1$, let $\Gamma(i)$ denote the set of neighbors of i in V_2 — i.e.,

$$\Gamma(i) \;=\; \{j \mid (i,j) \in E\}$$

Let $\Gamma'(i)$ denote the set of vertices in V_1 that the neighbors of i are connected to.

$$\Gamma'(i) \;=\; \{k \mid (k,j) \in E, \text{ where } j \in \Gamma(i)\}$$

Consequently, the set of candidate communities for i, $Cand(i)$ is given by:

$$Cand(i) \;=\; \bigcup_{k \in \Gamma'(i)} C(k)$$

Intuitively, a vertex $i \in V_1$ can only migrate to communities in V_1, within which it has at least one 2-hop neighbor (i.e., via its vertex neighbors in V_2). Moving to any other community in V_1 (i.e., not in this candidate set) will result in a decrease in modularity.

5.2 Calculating Modularity Gain

In the case of unipartite networks [4], calculating the expected modularity gain resulting from moving a vertex from one community to another can be executed in constant time if appropriate data structures are maintained. In the case of bipartite networks, this is not the same because of the following two lemmas (as shown in Figure 4):

LEMMA 1. *If vertex i moves from C to C', then the choice of co-clusters for either community could change.*

PROOF. The migration of vertex i from C to C' affects the values $e_{C,D}$, $e_{C',D}$, a_C and $a_{C'}$, for any D, in Eqn. 6. □

Note that the lemma applies to migrations of vertices in V_2 as well.

Define the community set $N(C)$ as follows:

$$N(C) \;=\; \bigcup_{j \in \Gamma(i)} D(j)$$

The above lemma leads to the following corollary:

COROLLARY 1. *If vertex i moves from C to C', then:*

a) any of $N(C)$'s co-clusters may change; and

b) any of $N(C')$'s co-clusters may change.

PROOF. Due to the migration of i from C to C', any community D_x of V_2 to which either of these two communities is connected (see Figure 4) could potentially change its co-cluster preference with changes to the right hand side values of its corresponding C_{D_x} — i.e., Eqn. 7. This implies that $D_x \in N(C) \cup N(C')$. □

It should be intuitively clear why these two lemmas should be true. In short, the two equations for co-cluster choices (Eqns. 6 and 7) depend on both the positive and the negative terms, and either of those two terms could change for the communities covered in Lemma 1 and Corollary 1 as a result of i moving out of C into C'. It should also be clear that the co-cluster choices for no other communities in the rest of the graph are affected by this move.

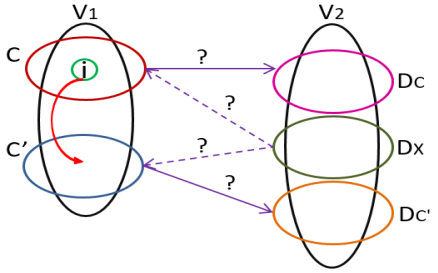

Figure 4: Illustration of vertex i evaluating its migration from community C to community C'. If this move were to be executed, the co-cluster choices for both C and C' (presently, D_C and $D_{C'}$ respectively) could change (Lemma 1). Furthermore, the co-cluster choice for an arbitrary community D_x which has at least one edge incident from either C or C' could also change (Corollary 1). A question mark indicates that the corresponding co-cluster affiliation needs to be re-evaluated.

Based on the above lemma and corollary, we constitute a set containing all affected communities — i.e., $S = \{C, C'\} \cup N(C) \cup N(C')$, and compute the overall net modularity gain resulting from $i's$ migration from C to C' as follows:

We calculate the contribution of a community D and its co-cluster C_D to the summation in the modularity Equation 5 as follows:

$$f(D, C_D) \;=\; e_{D,C_D} - a_D \times a_{C_D}$$

Consequently, the change in a community D's contribution to the overall modularity Q_B due to the change in its co-cluster (from D_C to D_C^{new} imposed by vertex i's move) is given by:

$$\Delta Q_B(D) \;=\; f(D, C_D) - f(D, C_D^{new}) \qquad (8)$$

We calculate this term for all affected communities (which includes C, C', all communities in $N(C)$ and $N(C')$) in order to derive the modularity gain for a single vertex move. This modularity gain is calculated for each possible vertex move into one of its candidate communities; and finally, vertex i is moved to that candidate community which maximizes the gain (assuming it is positive).

Vertex Ordering: In the *biLouvain* algorithm, the order in which vertices are processed could potentially impact the performance of the algorithm and the quality of community-wise partitioning to varying degrees, depending on the input and on the ordering scheme used.

To understand the impact of vertex ordering on performance, note that the community assignment made for a vertex on one partition (say V_1) at any given iteration is dependent on the community states of its neighboring vertices in the other partition (V_2), and also on the community states in the same partition (for selecting candidate communities). Also note that initially the number of communities on each partition is equal to its number of vertices. Coupled together, these observations imply that if vertex decisions are all made, say sequentially, within one partition prior to the other partition, then the time taken for processing ver-

tices on the first partition is likely to be significantly higher than for the vertices in the second partition during the same iteration. However, this performance impact is expected to diminish in the later iterations of the algorithm as communities get larger. Other vertex ordering schemes might alter this performance behavior.

As for quality, the impact of ordering is likely to be relatively less. It can be expected that for real world inputs with well-defined community structures, the state of communities typically converges faster in the first few iterations of the algorithm and largely remain stable in the later iterations[2]. However, the final output quality could still differ based on the vertex ordering used.

To evaluate this quality-time tradeoff imposed by vertex ordering, we implement these vertex ordering schemes:

1. **Sequential:** Within each iteration, the vertices in one partition (say, V_1) are all processed (in some arbitrary order) prior to vertices in the other partition.
2. **Alternate:** Within each iteration, the processing of vertices from the two partitions is interleaved — i.e., alternating between the two partitions, until one of the partitions is exhausted at which point the algorithm defaults to the sequential mode to cover the remaining vertices of the larger partition.
3. **Random:** Within each iteration, the order of processing vertices in $V_1 \cup V_2$ is randomized. By fixing the random seed, one can ensure that the output remains deterministic across multiple runs on the same input.

Another contributing factor to dictate an ordering scheme's impact on performance is the constituent vertex partition sizes. For instance, if the two partitions are of skewed sizes (e.g., $n_1 << n_2$) then the Random scheme is expected to have an advantage in run-time over the other two schemes, whereas if the two partitions have comparable sizes, then all three schemes can be expected to behave similarly.

5.3 Complexity Analysis

The run-time within an iteration is dominated by the time taken to calculate the modularity gain for all the vertices (Section 5.2). In our implementation we keep efficient data structures to enable us to calculate each community's contribution in constant time. The worst-case run-time complexity for calculating the best modularity gain for a given vertex i at any given iteration is $\mathcal{O}(n_2 \times n_1)$ (this assumes that all C communities are connected to all D communities).

As all vertices are linearly scanned within each iteration, the worst-case run-time complexity is $\mathcal{O}((n_1 + n_2)n_1n_2)$ per iteration — which makes our exact algorithm a cubic algorithm. In practice; however, we can expect inputs to be sparse — that would imply a quadratic behavior in the initial iterations; but as the algorithm progresses, the number of communities shrinks and with that also the run-time per iteration.

Algorithmic Extensions: The performance of the algorithm can be further improved by observing and taking advantage of certain structural motifs within bipartite networks and their relation to the modularity expression. Such structural motifs can include simple topological features such as a star (i.e., vertex i in V_1 is connected to k vertices in V_2), and a chain (i.e., a linked list) inside a bipartite network, or more complex attributes such as a k-core or inexact versions of chains and stars embedded within a larger subgraph.

In this paper, we prove two such lemmas, one for the simple star case (Figure 10(a)), and another for the simple chain case (Figure 10(b)). The lemmas and condensed versions of the proofs are shown in the Appendix.

The idea is that, based on these provable properties, the input graph can be preprocessed so as to detect such motifs and compact them into their corresponding community structures (as dictated by the lemmas). Thus, reducing the initial number of vertices will have a direct impact on the overall work of the *biLouvain* algorithm.

From testing, we find that identifying simple stars and chains can reduce the number of input vertices by a factor of up to 8%. While this may not in itself represent a significant work reduction, we are extending this idea to more prevalent structural motifs, such as the inexact versions of stars and chains, which could yield larger factors of improvement.

6. EXPERIMENTAL RESULTS

6.1 Experimental Setup

Test Platform: We used compute nodes of the Cori supercomputer at the National Energy Research Scientific Computing Center (NERSC). Each node has a 16-core Intel "Haswell" processor at 2.3 GHz, and 128GB RAM. Since our current implementation is serial, we used only one core. For graph visualization, we used the *Mango* software: Manipulation and Analysis of Networks and Gene Ontology [5].

Test inputs: We used real world and synthetic data sets for our testing (see Table 2). The real world data sets are as follows: a) *Southern Women [2]:* a women vs. social events bipartite graph, where edges represent the attendance of a woman at a social event;[3] b) *Plant-Pollinator [7]:* 4 of the 23 pollinator networks, binary and weighted, where each edge represents the frequency of a pollinator's visit to a plant[4]; c) *Malaria [14]:* a network showing a mapping between subsequences and the genes that contain them, in the malaria parasite *P. falciparum*; d) *Drug-Complexes [17]:* a network showing drug-protein target interactions; and e) *Gene-Drug [11]:* a network showing gene-drug interactions.

We also used two synthetic networks (*Synthetic1* and *Synthetic2*) generated using a simulation code with predefined probabilities for links and target community structures. More specifically, *Synthetic1* corresponds to a bipartite network with a well characterized community structure (probability of 0.9 for intra-cocluster edges and probability of 0.1 for inter-cocluster edges), whereas *Synthetic2* represents a random bipartite network with uniform degree distribution.

***biLouvain* setting:** All modularity results presented use the *Murata+* formulation defined in this paper (Section 4). Recall that *b*iLouvain has two parameters — the iteration and phase cutoffs (τ_i, τ_p respectively), as described in Section 5. We experimented with multiple values of τ_i in the interval $[10^{-6}, 10^{-2}]$ on different inputs. These preliminary experiments consistently showed that: a) the final output modularities hardly changed within the interval tested; whereas b) as τ_i is decreased, the number of iterations per phase increased, thereby increasing run-time to completion. Thus,

[2]This is confirmed in our experimental results (Section 6).

[3]This is a bipartite graph with a known community structure and we use this as a benchmark for validation.

[4]We experimented on all 23 networks, and select only the top 4 largest networks for presentation in this section.

Table 2: Input statistics, and *biLouvain*'s modularity and run-time performance on the inputs.

Input Data Set	Nodes n_1	Nodes n_2	Edges m	Modularity (Q_B)	Time t(sec)
SouthernWomen	18	14	89	0.575	0.1
memmott1999	25	79	299	0.431	0.9
kevan1970	30	114	312	0.540	1.4
junker2013	56	257	572	0.560	5.9
kato1990	91	679	1,206	0.636	48.6
Malaria	297	806	2,965	0.630	34.8
Drug-Complex	680	739	3,690	0.839	22.4
Gene-Drug	3,090	14,311	29,389	0.822	3,402
Synthetic1	21	180	216	0.816	1.3
Synthetic2	67	10	424	0.505	1.5

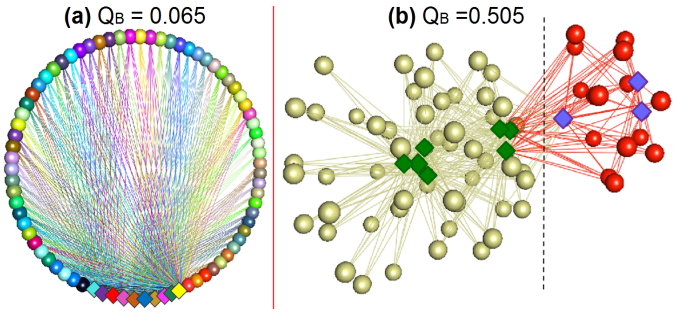

Figure 5: (a) *Synthetic2* bipartite network: Circles represent V_1 vertices, and diamonds V_2 vertices. (b) Bipartite communities output by *biLouvain*: Yellow and red vertices form two communities in V_1, while green and purple vertices form two communities in V_2. The dashed line shows the division between the two co-clusters.

we set the default value of $\tau_i = 10^{-2}$ throughout our experiments. We set the phase cutoff τ_p to 0.0 in all our experiments. Note that this represents a conservative setting where the algorithm is allowed to terminate only when two consecutive phases produce *no* change in the overall modularity. We also evaluated the quality-time tradeoff among different vertex ordering schemes in Section 6.3. As our default setting, we used the *Random* ordering scheme.

6.2 Qualitative Assessment

Validation: First, we validate our *biLouvain* algorithm using the Southern Women benchmark and the two synthetic networks. For the Southern Women, our algorithm was able to reproduce the expected communities [10] *identically* (not shown due to lack of space).

For both synthetic networks, the results were along expected lines. In Figure 5, we show the *Synthetic2* input, and the bipartite community division output from *biLouvain*. Recall that this is a random network with uniform degree distribution. Yet, our algorithm was able to achieve a modularity of 0.505. On the *Synthetic1* input, which was configured to have a stronger community structure, the output modularity was 0.816 and the expected community structure was recovered (results not shown due to space).

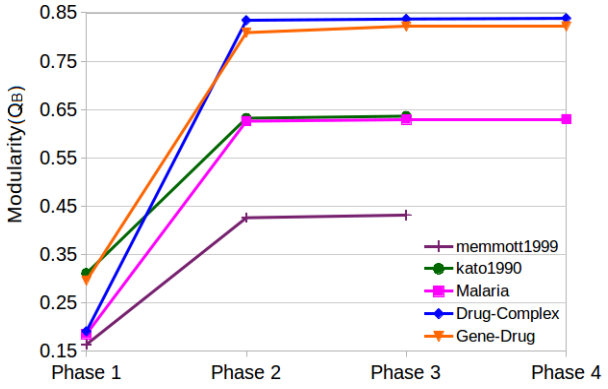

Figure 6: *biLouvain* bipartite modularity evolution by phase on different inputs.

Modularity evolution: We studied the increase in modularity as a function of the number of iterations and phases. As can be seen in Figure 6, the first phase typically contributed to the maximum gain in modularity indicating the minimal role of graph compaction on the inputs tested. Graph compaction can prove more effective on networks which are likely to result in local maximal solutions at the end of the first phase (e.g., inputs with bicliques). We also observed that the number of iterations per phase was at most 3 for the inputs tested, illustrating the effectiveness of the first few iterations.

6.2.1 *Cluster Assessment*

We assessed the significance of the bipartite communities output by the *biLouvain* algorithm, on the Gene-Drug network, which was the largest real world network tested. For assessment, we computed a Gene Ontology (GO)-based significance for each gene cluster detected by our algorithm.

The *biLouvain* algorithm detected 1,092 gene clusters from the Gene-Drug network, of which 505 clusters consisted of two or more genes. We computed the GO significance for these 505 clusters using *gProfileR* [20]. The analysis resulted in 428 clusters with GO term annotations. The significance of a particular GO term, associated with a group of genes, is given by its p-value. We used the conservative approach of assigning the *maximum* p-value from within each cluster to be the cluster's p-value. Based on this conservative scheme, we found that *all* of the 428 clusters have a p-value of 0.05 or less — indicating a minimum confidence level of 95%.

6.2.2 *Effect of Edge Weights*

We studied the effect of introducing edge weights using the plant-pollinator networks. Figure 7 shows results of our analysis. As can be observed, the resulting community structures for the weighted vs. binary inputs are different and are consistent with the weight distribution of edges.

6.3 Performance of vertex ordering schemes

In Section 5.2 we described three vertex ordering schemes and their potential impact on *biLouvain*'s quality and performance. We studied this quality-time tradeoff on two of the larger real world networks: the Drug-Complex network, which represents the case of an even size distribution between the two vertex partitions — i.e., $n_1 \simeq n_2$; and the

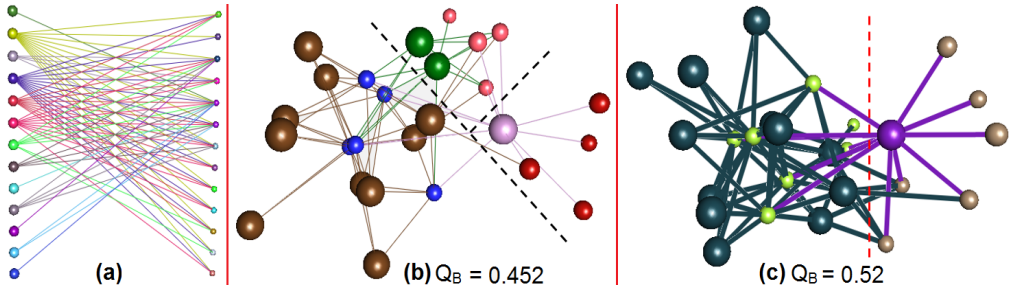

Figure 7: The set of bipartite communities resulting from analyzing the *bezerra2009* network (shown in (a)). The communities (and co-clusters) resulting from the binary network and the weighted network are shown in parts (b) and (c) respectively. The large nodes represent 13 Malpighiaceae oil-flowers, and the small nodes represent 13 oil-collecting bees. Nodes belonging to the same community are shaded with the same color.

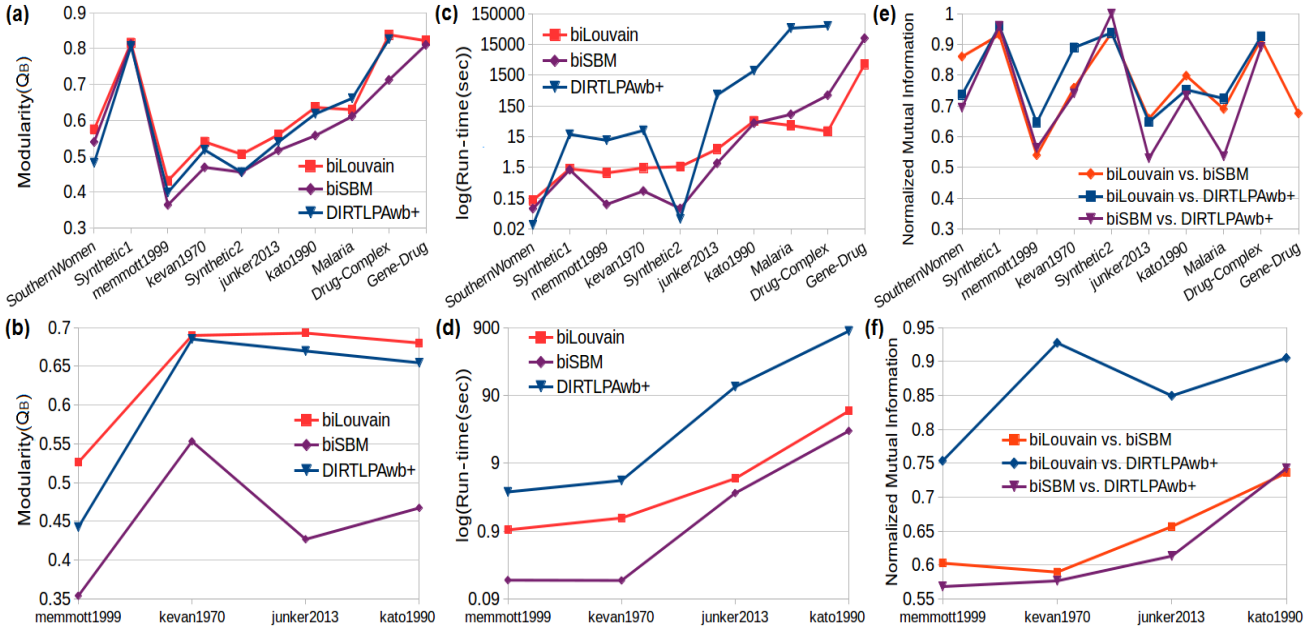

Figure 8: Comparison among *biLouvain*, biSBM, and DIRTLPAwb+. Modularity, run-time and NMI results for binary networks (charts (a),(c),(e)), and for weighted networks (charts (b),(d),(f)).

Gene-Drug network, which represents the case of a skewed size distribution between the partitions (here, $n_1 \ll n_2$.

Figure 9 depicts this quality-time tradeoff for these two cases. For Drug-Complex (chart (a)), we observed that all three schemes behave similarly (both by quality and performance). For the Gene-Drug (chart (b)), we observed that the Random scheme demonstrated the best tradeoff, showing a reduction of 0.05 in modularity relative to Sequential, while improving the performance by a factor of 3.43×. These results confirm the expected efficacies of the ordering schemes. These results provide a guide to the choice of the ordering scheme based on the input.

6.4 Comparative Evaluation

We compared our *biLouvain* algorithm with two other tools — the Label Propagation (*DIRTLPAwb+*) method [3], and the Stochastic Block Model (*biSBM*) [14]. These two methods represent the state-of-the-art in bipartite community detection. For the purpose of comparatively evaluating the quality of the communities generated by the different methods, we use the *Murata+* modularity. We also computed a pairwise Normalized Mutual Information (NMI) [6] of the outputs generated between any two methods.

While running both tools (*DIRTLPAwb+* and *biSBM*), we observed slight variations in the outputs across multiple executions on the same input. For the purpose of our comparison, we selected an arbitrary output from each of them.

Furthermore, for *biSBM*, the tool also requires the user to input the number of output communities on either side (k_1 for V_1 and k_2 for V_2). Therefore, to obtain the best possible results from *biSBM*, we ran the *biSBM* under multiple configurations and used the results from the configuration that produced the *best* modularity. The four configurations tested are as follows — the numbers of communities (k_1 and k_2) are set to the numbers output by: a) *DIRTLPAwb+* [3], and b) our *biLouvain* algorithm run in each of the vertex ordering schemes (Sequential, Alternate and Random).

Figure 8 shows the results of our comparative study on both binary and weighted network inputs. In terms of the

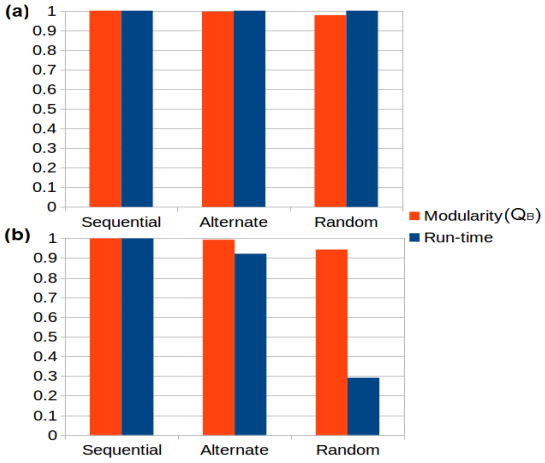

Figure 9: Evaluation of performance of *biLouvain* vertex ordering schemes. (a) Drug-Complex data set ($n_1 \simeq n_2$). (b) Gene-Drug data set ($n_1 \ll n_2$).

output quality (modularity; charts (a) and (b)), our *biLouvain* algorithm produces the best modularity for all but one input.

As for performance (run-time; charts (c) and (d)), there is a variance in performance for the smaller inputs; but for the larger inputs (Malaria, Drug-Complex, and Gene-Drug), *biLouvain* is orders of magnitude faster than the other two tools. For the Drug-Complex network, *biLouvain* is 14.8× faster than *biSBM* and 2,655× faster than *DIRTLPAwb+*. For Gene-Drug, *biLouvain* completes in 56 minutes, while *biSBM* had to be run using small values of $k1 = k_2 = 30$ in order to complete in reasonable time (6 hours 35 minutes). The *DIRTLPAwb+* tool failed to complete in 48 hours.

We compared the community compositions produced by the three methods, and the results (charts (e) and (f)) show that the pairwise NMI between methods can vary anywhere between 0.5 and 1 for the inputs tested. While there is hardly a consensus, our *biLouvain* algorithm's outputs tend to be generally closer to the outputs from *DIRTLPAwb+* than to *biSBM*. This can be attributed to the fact that, methodology-wise, the label propagation and *biLouvain* approaches are more similar to one another, as both scan local neighborhoods to determine a vertex's community.

In [17], modularity of the Drug-Complex network was calculated using a simulated annealing algorithm. The results yielded a modularity of 0.824 with 48 communities for drugs and a modularity of 0.762 with 42 communities for complexes. In comparison to the above, the *biLouvain* algorithm yields a bipartite community structure with $Q_B = 0.839$. We also ran the *biSBM* tool setting $k_1 = 48$ and $k_2 = 42$ (as per [17]), and this yielded an output modularity of 0.788. Running *DIRTLPAwb+* gives a modularity of 0.826.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an efficient algorithm, *biLouvain*, for the problem of community detection in bipartite networks. Our method uses a revised definition of the classical Murata's modularity definition, and extends the Louvain heuristic to work for bipartite networks. Experimental results show that our *bi*Louvain algorithm identifies communities that have a comparable or better quality (bipartite modularity) than two state-of-the-art methods, while significantly reducing the time-to-solution between one and three orders of magnitude for large input networks.

Multiple future extensions have been planned. These include: i) Parallelization to further reduce the time to solution and enhance problem size reach; ii) Algorithmic optimizations to fully take advantage of structural motifs within the input graph; iii) Incorporation of intra-type edge information, where available, in addition to inter-type edges as part of modularity computation; iv) Comparative evaluation with other modularity formulations and with other existing methods such as spectral methods; and v) Large-scale application to networks obtained from multiple domains and subsequent scientific analysis.

8. ACKNOWLEDGMENTS

We wish to thank the Mango graph visualization team at Iowa State University. This research was supported by US Department of Energy grant DE-SC-0006516. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

9. REFERENCES

[1] M. J. Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6):066102, 2007.

[2] V. Batagelj and A. Mrvar. Pajek datasets. `http://vlado.fmf.uni-lj.si/pub/networks/data/`.

[3] S. J. Beckett. Improved community detection in weighted bipartite networks. *Royal Society Open Science*, 3(1):140536, 2016.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[5] J. Chang and H.-H. Chou. Mango: Manipulation and analysis of networks and gene ontology. `http://www.complex.iastate.edu/download/Mango/`.

[6] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[7] C. F. Dormann, J. Frueund, N. Bluethgen, and B. Gruber. Indices, graphs and null models: analyzing bipartite ecological networks. *The Open Ecology Journal*, 2:7–24, 2009.

[8] J. T. Dudley, T. Deshpande, and A. J. Butte. Exploiting drug–disease relationships for computational drug repositioning. *Briefings in bioinformatics*, page bbr013, 2011.

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

[10] L. C. Freeman. Finding social groups: A meta-analysis of the southern women data. *Dynamic social network modeling and analysis*, pages 39–97, 2003.

[11] M. Griffith, O. L. Griffith, A. C. Coffman, J. V. Weible, J. F. McMichael, N. C. Spies, J. Koval, I. Das, M. B. Callaway, J. M. Eldred, et al. Dgidb: mining the druggable genome. *Nature methods*, 10(12):1209–1210, 2013.

[12] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral. Module identification in bipartite and directed networks. *Physical Review E*, 76(3):036102, 2007.

[13] S. Ji. Computational genetic neuroanatomy of the developing mouse brain: dimensionality reduction, visualization, and clustering. *BMC bioinformatics*, 14(1):1, 2013.

[14] D. B. Larremore, A. Clauset, and A. Z. Jacobs. Efficiently inferring community structure in bipartite networks. *Physical Review E*, 90(1):012805, 2014.

[15] X. Liu and T. Murata. An efficient algorithm for optimizing bipartite modularity in bipartite networks. *JACIII*, 14(4):408–415, 2010.

[16] T. Murata. Detecting communities from bipartite networks based on bipartite modularities. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 50–57. IEEE, 2009.

[17] J. C. Nacher and J.-M. Schwartz. Modularity in protein complex and drug interactions reveals new polypharmacological properties. *PloS one*, 7(1):e30028, 2012.

[18] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[19] J. Piñero, N. Queralt-Rosinach, À. Bravo, J. Deu-Pons, A. Bauer-Mehren, M. Baron, F. Sanz, and L. I. Furlong. Disgenet: a discovery platform for the dynamical exploration of human diseases and their genes. *Database*, 2015:bav028, 2015.

[20] J. Reimand, M. Kull, H. Peterson, J. Hansen, and J. Vilo. g: Profiler—a web-based toolset for functional profiling of gene lists from large-scale experiments. *Nucleic acids research*, 35(suppl 2):W193–W200, 2007.

[21] K. Suzuki and K. Wakita. Extracting multi-facet community structure from bipartite networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 312–319. IEEE, 2009.

[22] M. Wardeh, C. Risley, M. K. McIntyre, C. Setzkorn, and M. Baylis. Database of host-pathogen and related species interactions, and their global distribution. *Scientific data*, 2, 2015.

APPENDIX

A. STAR PROPERTY

LEMMA 2. *A connected component of $G(V_1 \cup V_2, E, \omega)$ that is a vertex $i \in V_1$ connected to k vertices $\{j_1, j_2, \ldots, j_k\}$ in V_2, is guaranteed to collapse into a single co-cluster.*

PROOF. Figure 10(a) illustrates a star input. Let us consider unit weight edges for simplicity. Vertex i is guaranteed to be in a community of its own when the algorithm terminates, because all its neighbors (j's) in V_2 are of unit degree. As for the community assignment for all the j's, we consider two possible cases:

Case a) All the k vertices in V_2 are merged into one community. For this case, the contribution (Q'_B) of all these $k+1$ vertices, including i, to the overall modularity (Q_B) is as follows (by Eqn. 5):

$$Q'_B = \frac{1}{2M}\left(2k - \frac{2k^2}{2M}\right) \tag{9}$$

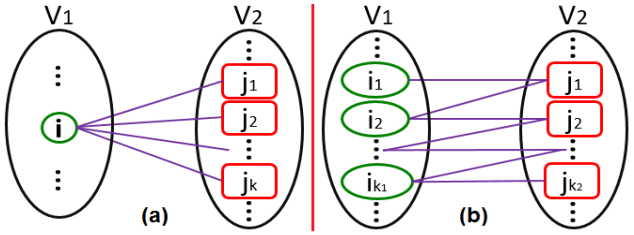

Figure 10: Illustration of a) the star case, and b) the chain case.

Case b) $k-1$ vertices in V_2 are merged into one community, the remaining one is left in its own community. The corresponding modularity contribution is:

$$Q'_B = \frac{1}{2M}\left((2k-1) - \frac{2k^2-k}{2M}\right) \tag{10}$$

A comparison of Eqn. 9 and Eqn. 10 indicates that the contribution of case (a) can be shown to be always greater than the contribution of case (b), as $M \geq k$ (expression not shown). □

B. CHAIN PROPERTY

LEMMA 3. *A connected component within $G(V_1 \cup V_2, E, \omega)$ that is a linear chain of length ℓ (in the number of edges), can collapse into one co-cluster if and only if $\ell < 2\sqrt{M}$.*

PROOF. Figure 10(b) illustrates a chain input. For convenience we assume unweighted edges. We consider two possible cases to determine the optimal length of a chain for modularity maximization.

Case a) All vertices forming the chain on each partition, are assigned to the same community. For this case, the contribution of the chain (Q'_B) to the overall modularity (Q_B) is as follows:

$$Q'_B = \frac{1}{2M}\left[2\ell - \frac{\ell^2}{M}\right] \tag{11}$$

Case b) The chain is split in two parts, such that community pairs of lengths (C_0, D_0) and (C_1, D_1), form two co-clusters. The corresponding modularity contribution is:

$$Q'_B = \frac{1}{2M}\left[2\ell - 2 - \frac{\ell}{M}\left(\ell^2 - \ell D_0 - \ell C_0 + 2C_0 D_0\right)\right] \tag{12}$$

Maximizing the negative term above:

$$\begin{aligned} \frac{\partial y}{\partial C_0} &= -\ell + 2D_0 = 0 \\ \frac{\partial y}{\partial D_0} &= -\ell + 2C_0 = 0 \\ \Rightarrow C_0 = D_0 &= \frac{\ell}{2} \end{aligned} \tag{13}$$

Substituting Eqn. 13 in Eqn. 12:

$$Q'_B = \frac{1}{2M}\left[2\ell - 2 - \frac{1}{M}\left(\frac{\ell^2}{2}\right)\right] \tag{14}$$

For case (a) to be preferred over case (b):

$$\begin{aligned} \Delta Q'_B &= \frac{1}{2M}\left[2\ell - 2\ell + 2 - \frac{\ell^2}{M} + \frac{\ell^2}{2M}\right] > 0 \\ \Rightarrow \ell &< 2\sqrt{M} \end{aligned} \tag{15}$$

□