# Network Backbone Discovery Using Edge Clustering

Ning Ruan† Ruoming Jin† Guan Wang†
† Department of Computer Science
Kent State University
{nruan,jin,gwang0}@cs.kent.edu

Kun Huang‡
‡ Department of Biomedical Informatics
Ohio State University
khuang@bmi.osu.edu



***Abstract*—In this paper, we investigate the problem of network backbone discovery. In complex systems, a "backbone" takes a central role in carrying out the system functionality and carries the bulk of system traffic. It also both simplifies and highlight underlying networking structure. Here, we propose an itegrated graph theoretical and information theoretical network backbone model. We develop an efficient mining algorithm based on *Kullback-Leibler* divergence optimization procedure and maximal weight connected subgraph discovery procedure. A detailed experimental evaluation demonstrates both the effectiveness and efficiency of our approach. The case studies in the real world domain further illustrates the usefulness of the discovered network backbones.**

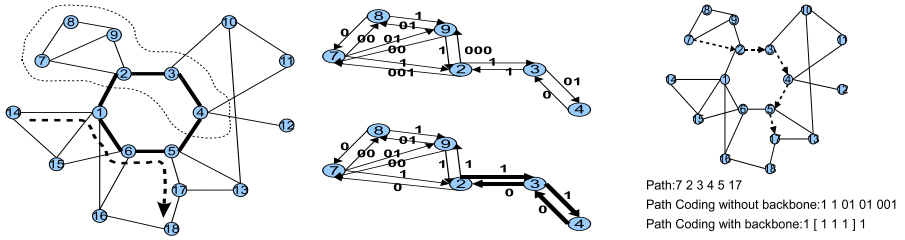
## I. Introduction

In many man-made complex networking systems, a "backbone" takes a central role in carrying out the system functionality. The clearest examples are the highway framework in the transportation system and the backbone of the Internet. When studying these systems, the backbone offers both a concise and highlighted view. Furthermore, it also provides key insight on understanding how the entire system organizes and works. Does "backbone" exist in natural or social network? What should it look like? How we can discover them efficiently? Interestingly, the backbone phenomena has been recently observed in several natural and social systems, including metabolic networks [2], social networks [11], and food webs [19]. Unfortunately, there is no formal definition of the network backbone and no goodness function defined in all existing work.

In the paper, we propose the first theoretical network backbone model under an integrated graph theoretical and information theoretical framework. Intuitively, network backbone is a connected subgraph which carries major network "traffic". It can simplify and highlight underlying structures and traffic flows in complex systems. A complex system's behavior relies on proper communication through the underlying network substrate, which invokes a sequence of local interactions between adjacent pair of vertices. These interactions thus form system-wide network traffic [17]. It is essential for a complex system to deliver such traffic in an energy efficient way [3]. Here, at least two energy costs should be considered: 1) communication cost and 2) organization cost in defining/recognizing communication path. For the first cost, energy-efficient way for one vertex communicating with another vertex naturally points to shortest path, i.e., the information flow over a network primarily follows the shortest ones [7]. The second cost can be described as a path-recognition complexity. Based on information theory, we could depict it as the shortest coding length of a given path. In general, the optimal code length for an edge $(i, j)$ can be bounded by $\log \frac{1}{p(i \to j)}$, where $p(i \to j)$ is the probability of a shortest path taking edge $(i, j)$ when it goes through vertex $i$ [6].

Minimizing these two costs gives rise to the network backbone structure: the shortest paths form a traffic flow which must efficiently travel from source to destination using the backbone. Especially, if only the first cost is considered, only the edges with high betweenness [7] (roughly speaking, edge-betweenness defines the likelihood of any shortest path going through an edge) will be selected; however, those edges are not necessarily connected [7], how they should work together in delivering the system-wide traffic is unknown. In this aspect, the second cost enables us to further constrain the backbone structure using the path-recognition complexity. A backbone that is too simple or has wrong topology is not an efficient route for vertex-vertex paths (first cost). A backbone that is too complex is expensive to describe shortest paths (second cost).

Figure 1 illustrates a backbone and its usage in reducing the description length of a shortest path. Since our computational framework implicitly evaluates path's information complexity by statistical likelihood, it does not rely on any particular coding scheme (i.e., Huffman code is used here only for illustrative purpose). Figure 1(a) shows a network with its highlighted backbone. Figure 1(b) focuses on a subgraph, showing Huffman code for each edge, where the upper part is for the edge code without utilizing backbone and the lower part is for using backbone. For instance, edge $(9, 2)$ with direction $9 \to 2$ is assigned with code $1$, and with direction $2 \to 9$ is assigned code $000$.In Figure 1(c), we show path codings using and without using the backbone. We basically list each edge code in the path consecutively. Specifically, the subpath $(2, 3, 4, 5)$ is in the backbone, and we use "[" and "]" to denote the entering and exiting of the backbone, respectively. It can also be observed that even with




(a) Network and its backbone (b) Coding of subgraph (c) Path coding example

Fig. 1: Path encoding with and without backbone

an extra coding cost for entering and exiting backbone, the lower coding cost for paths inside the backbone can still be beneficial.

We note that backbone discovery problem resonates with the recent efforts in the graph mining community on graph simplification. To deal with the scale and complexity of graph data, reducing graph complexity or graph simplification is becoming an increasingly important research topic [25], [23], [18], [1]. Generally, graph simplification focuses on sparsifying graphs by reducing non-essential edges [25], [23], [18], extracting key vertices [22], [9], or collapsing substructures into supernodes [14], [1]. These simplified structures are able to facilitate many real-world applications, such as topology visualization [16], [8] and computational speedup on various graph-centered tasks [10], [22], [18]. From this perspective, the backbone structure can potentially serve as a simplification approach, which highlights a core set of vertices and edges in the original network.

## II. Statistical Learning Framework for Backbones

In this section, we introduce and refine the backbone description under a statistical learning framework. Based on the well-known relationship between information complexity and statistical likelihood, the information complexity of each path (or a set of paths) can be equivalently represented as a likelihood function. Further, this reformulation also generalizes the notion of backbone to optimize the information complexity for any given set of paths (information pathway) beyond the shortest paths.

### A. Notations

To facilitate our discussion, we introduce the following notations. Let $G = (V, E)$ be a undirected graph with edges $E \subseteq V \times V$. Since each edge $(u, v) \in E$ in the graph may be assigned two codes, one for $(u \to v)$ and another for $(v \to u)$ (See Figure 1), it is more convenient to consider each undirected edge $(u, v)$ as two directed edges $(u \to v)$ and $(v \to u)$. Therefore, we represent undirected graph $G$ as a bidirected graph, i.e., $G = (V, \mathcal{E})$ where $\mathcal{E} = \cup_{(u,v)\in E}\{(u \to v), (v \to u)\}$. Note that when we say an edge $(u, v)$ is a backbone edge in a undirected graph, it suggests that both directed edges $(u \to v)$ and $(v \to u)$ from $\mathcal{E}$ are backbone edges in the bidirected graph. The same holds for non-backbone edges. This constraint does not hold for directed graphs, where each directed edge is evaluated independently. Though this paper mainly focuses on undirected graphs, this backbone discovery framework can be easily generalized to directed graphs.

For a vertex $u \in V$, let $\mathcal{N}(u)$ be the immediate neighbors of $u$, i.e., $\mathcal{N}(u) = \{v|(u \to v) \in \mathcal{E}\}$. Let $\mathcal{I}(u)$ be all the incoming edges of vertex $u$, i.e., $\mathcal{I}(u) = \{(w \to u)|w \in \mathcal{N}(u)\}$ and let $\mathcal{O}(u)$ be all the outgoing edges of vertex $u$, i.e., $\mathcal{O}(u) = \{(u \to v)|v \in \mathcal{N}(u)\}$. Note that, in bidirected graphs, $\mathcal{I}(u) = \mathcal{O}(u)$. A path $P$ from vertex $u$ to vertex $v$ can be defined as a vertex-edge sequence, i.e., $(u_0, e_1, u_1, e_2, \cdots, u_{k-1}, e_k, u_k)$, where $u = u_0$, $v = u_k$, and $(u_{i-1} \to u_i) = e_i$. When no confusion arises, we use only the edge sequence to describe the path as $(e_1, e_2, \cdots, e_k)$. In this paper, we only consider simple path such that no vertex appears more than once in a path. The path length is defined as the number of edges in the path. The shortest path from vertex $u$ to $v$ is the one with minimal path length which is denoted as $P_{uv}$.

Let $\mathcal{P}$ be a collection of paths in graph $G$ characterizing system-wide information flow. Without prior knowledge, we consider $\mathcal{P}$ to contain shortest paths for each pair of vertices in graph $G$, i.e, $\mathcal{P} = \{P_{uv}\}$. In case there is more than one shortest path between a pair of vertices, only one shortest path is added to $\mathcal{P}$. This is consistent with the earlier assumption that any two pairs of vertices have equal communication frequency. Alternatively, we may assign each path with an equal weight such that total weight of all shortest paths is one.

### B. Two Simple Models

In this subsection, we consider two simple statistical models for generating a collection of paths in $\mathcal{P}$. At a high level, both generative models try to assign each edge a probability (or multiple probabilities) such that the probability of each path can be derived by augmenting its edges' probabilities.

**Edge Independent Model:** In the first model, referred to as *edge independent model*, each outgoing edge $(u \to v)$ of a vertex $u$ in the graph is assigned a probability $p(u \to v)$, and the sum of the probabilities of all outgoing edges should be equal to one, i.e., $\sum_{(u\to v)\in\mathcal{O}(u)} p(u \to v) = 1$. Furthermore, we assume any two edges in a path are

independent. Given this, the probability of a path $P = (v_0, e_1, v_1, e_2, v_2, \cdots, v_{k-1}, e_k, v_k)$ is $p(P) =$

$$p(e_1)p(e_2)\cdots p(e_k) = p(v_0 \to v_1)p(v_1 \to v_2)\cdots p(v_{k-1} \to v_k)$$

Given the collection of paths $\mathcal{P}$, the overall likelihood $L_I(\mathcal{P})$ of these paths being generated from this model is

$$L_I(\mathcal{P}) = \prod_{P \in \mathcal{P}} p(P) = \prod_{e \in \mathcal{E}} p(e)^{N_e} \tag{1}$$

where $N_e$ is the number of paths in $\mathcal{P}$ passing through edge $e$. Note that if $\mathcal{P}$ includes all shortest paths in $G$, then $N_e$ is often referred to as the edge betweenness [7]. To maximize the overall likelihood $L_I(\mathcal{P})$, using the Lagrange multiplier method, it is easy to derive that the optimal parameters are

$$p(e) = p(u \to v) = \frac{N_e}{M_u},$$

where $M_u$ is the number of paths going through $u$, i.e., reaching vertex $u$ and then continue to one of its neighbors.

Clearly, there are a total of $2|E| = |\mathcal{E}|$ parameters in the edge independence model. Note that this model directly corresponds to the coding scheme where each edge $(u \to v)$ is assigned a unique code with length $\frac{1}{p(u \to v)}$. Thus, under this scheme, the overall minimal coding length for all paths in $\mathcal{P}$ is simply the negative log-likelihood, $-\log L_I(\mathcal{P})$.

**Edge Markovian Model:** The edge independent model is one of the simplest models for describing the path probability. However, it is also rather unrealistic and results in poor model performance. A path itself is a correlation between edges, so the edge probability model should consider this. In the second model, we replace the independent assumption with the Markovian property, i.e., the probability of an edge is determined by its immediately preceding edge in a path. Given this, the probability of a path $P = (e_1, e_2, \cdots, e_k)$ is rewritten as

$$p(P) = p(e_1)p(e_2|e_1)\cdots p(e_k|e_{k-1})$$

where $p(e_j|e_i)$ is the conditional probability of edge $e_j$ appearing after edge $e_i$ in the path.

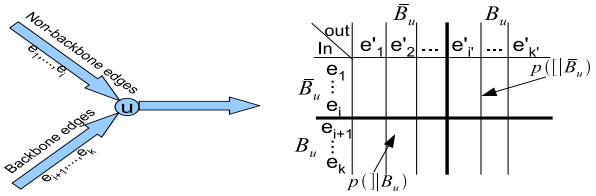
Given the path collection $\mathcal{P}$, the likelihood function for generating all these paths can be written as

$$L_M(\mathcal{P}) = \prod_{P \in \mathcal{P}} p(P) = \prod_{e \in \mathcal{E}} p(e)^{N'_e} \prod_{e,e' \in \mathcal{E}} p(e'|e)^{N_{ee'}}$$

where $N'_e$ is the number of paths in $\mathcal{P}$ starting with edge $e$, and $N_{ee'}$ is the number of paths with consecutive edges $(e, e')$. We note that this model directly corresponds to a Markov chain where each edge represents a state and the conditional probability $p(e'|e)$ represents a transition probability from state (edge) $e$ to state (edge) $e'$. However, though this model is more accurate at capturing the paths in graph $G$, it is also much more expensive in terms of number of parameters. It requires $\sum_{v \in V} |\mathcal{I}(v)| \times |\mathcal{O}(v)| = \sum_{v \in V} |\mathcal{N}(v)|^2$ parameters.

### C. Bimodal Markovian Model

We propose a new model with reduced number of parameters to improve model performance. This model is motivated by the observation on the usage of highway in the transportation system: compared to complex local traffics, less information is needed to describe highway traffic. Mapping back to modeling a path using backbone structure, this suggests that its subpaths only consisting of edges from backbone can be described in a relatively coarse manner comparing to subpaths containing non-backbone edges. Complying with this intuition's guidance, we introduce Bimodal Markovian Model utilizing *backbone* structure to reduce the number of parameters in edge markovian model while minimizing the loss of its modeling accuracy. It is termed *bimodal* because all vertices and edges would be categorized into two groups in the model.




(a) Backbone vertex $u$ (b) Probability table

Fig. 2: Example of backbone vertex $u$

Figure 2 illustrates the basic idea of this model. In figure 2(a), for each vertex $u$, all its incoming edges are divided into two categories, the *backbone edges* and the *non-backbone edges*. For each outgoing edge $e = (u \to v)$ from backbone vertex $u$, it only has two probabilities, conditional on the categories of incoming edges, denoted as $p(e|\mathcal{B} \cap \mathcal{I}(u))$ and $p(e|\overline{\mathcal{B}} \cap \mathcal{I}(u))$, where $\mathcal{B}$ is the set of all backbone edges and $\overline{\mathcal{B}}$ is the set of all non-backbone edges ($\mathcal{B} \cup \overline{\mathcal{B}} = \mathcal{E}$). In other words, multiple probabilities of edge $e$ conditional on different incoming edges are reduced to only two probabilities. From clustering viewpoint, this can be considered as performing biclustering on each vertex's incoming edges.

It is worthwhile to note that only the conditional probability of directed edge starting from backbone vertex in edge markovian model would be affected by this model. In other words, we still keep the conditional probability expressed in edge markovian model for edges starting with non-backbone vertex. For instance, consider the shortest path $P = (14, 1, 6, 5, 17, 18)$ highlighted in Figure 1(a). The probability of this path can be expressed by bimodal markovian model as $p(P) =$

$$p(14 \to 1)p(1 \to 6|\overline{\mathcal{B}})p(6 \to 5|\mathcal{B})p(5 \to 17|\mathcal{B})p(17 \to 18|5 \to 17)$$

Note the conditioning on the first edge entering the backbone and the first edge after leaving the backbone: even though $(1 \to 6)$ is a backbone edge, the probability we must use is $p(1 \to 6|\overline{\mathcal{B}})$ and though $(5 \to 17)$ is a non-backbone edge, its probability is $p(5 \to 17|\mathcal{B})$. For consecutive edges $(5 \to 17 \to 18)$ without involving backbone vertices, the conditional probabilities of $(17 \to 18)$ follows what we used in edge markovian model.

Given this, the overall likelihood for a given collection of paths $\mathcal{P}$ with respect to the backbone subgraph $B =$

$(V_B, E_B)$ is described as $L_B(\mathcal{P}) = \prod_{P\in\mathcal{P}} p(P)$

$$= \prod_{e\in\mathcal{E}} p(e)^{N'_e} \prod_{u\notin V_B \wedge e'\in\mathcal{O}(u)} p(e'|e)^{N'_{ee'}} \prod_{u\in V_B\wedge e\in\mathcal{O}(u)} p(e|\mathcal{B})^{N_{\mathcal{B}e}} \prod_{u\in V_B\wedge e\in\mathcal{O}(u)} p(e|\overline{\mathcal{B}})^{N_{\overline{\mathcal{B}}e}} \quad (2)$$

where $N'_e$ is the number of paths in $\mathcal{P}$ starting from edge $e$, $N'_{ee'}$ is the number of paths with consecutive edges $(e, e')$ while intermediate vertex connecting two edges are not backbone vertex, $N_{\mathcal{B}e}$ and $N_{\overline{\mathcal{B}}e}$ denote the number of paths passing through $e$ when its starting vertex is backbone vertex or non-backbone vertex, respectively. In connection with edge markovian model, we observe that for backbone vertex $u$ and its outgoing edge $e = (u \to v)$, $N_{\mathcal{B}e} = \sum_{e'\in\mathcal{B}\cap\mathcal{I}(u)} N_{e'e}$ and $N_{\overline{\mathcal{B}}e} = \sum_{e'\in\overline{\mathcal{B}}\cap\mathcal{I}(u)} N_{e'e}$, where $N_{e'e}$ is the number of paths in $\mathcal{P}$ containing the consecutive edges $(e'e)$.

In our framework, the overall number of parameters in bimodal markovian model is $\sum_{v\notin V_B} |\mathcal{I}(v)| \times |\mathcal{O}(v)| + \sum_{v\in V_B} 2|\mathcal{O}(v)| = \sum_{v\notin V_B} |\mathcal{N}(v)|^2 + \sum_{v\in V_B} 2|\mathcal{O}(v)|$. Compared to edge markovian model, the saving regarding the number of parameters is
$\sum_{v\in V_B}(|\mathcal{I}(v)| - 2) \times |\mathcal{O}(v)|$.

Given this, we formally define optimal backbone discovery problem based on bimodal markovian model:

*Definition 1:* **(Optimal $K$-Backbone Discovery Problem)** Given a complex network $G = (V, E)$, the targeted path set $\mathcal{P}$ and the number of backbone vertices $K$, the network backbone $B = (V_B, E_B)$ is a connected subgraph with $|V_B| = K$ such that $L_B(\mathcal{P})$ in Formula 2 is maximized.

Note that in this definition, we allow the user to define the number of vertices in the backbone structure. Alternatively, as a model selection problem, we may use a parameter penalty to help determine the optimal backbone size. For instance, if we use the Akaike information criterion (AIC), then, we simply want to optimize $-\log L(\mathcal{P}|B) + (|\mathcal{B}| + \sum_{v\in V_B} |N(v)|)$. If we use the Bayesian information criterion (BIC), then, our goal is to optimize $-2\log L(\mathcal{P}|B) + (|\mathcal{B}| + \sum_{v\in V_B} |N(v)|)\log|\mathcal{P}|$, where $|\mathcal{P}|$ is considered to be the sample size. In this paper, our decision to study the optimal backbone model problem for any given number of vertices is based on several considerations. First, though this model treats backbone discovery as a model selection problem, in many applications, the construction of backbones might involve other costs. Thus, it is more convenient and flexible to set up an adjustable number of vertices. Second, the solution of this problem forms the basis for solving the AIC or BIC criterion as we can utilize the solution with respect to different $K$ and then choose the overall optimal backbones. Finally, it is important and useful to observe how the backbone grows (shrinks) when its size increases (decreases). Therefore, in the reminder of the paper, we focus on studying the Optimal K-Backbone Discovery Problem. We will empirically investigate the relationship between backbone size and the likelihood of bimodal markovian model in subsection VI-B.

**Relationship to Path Coding Length:** Before we work towards a solution for this problem, let us first confirm its relationship to the problem of optimizing path-recognition complexity (section I). Given a backbone structure $G_B$, it is not hard to see that $-\log L_B(\mathcal{P})$ serves as the corresponding coding length of $\mathcal{P}$ in network $G$. When $L_B(\mathcal{P})$ is maximized, the optimal coding can be derived to describe network $G$.

In addition, recall that in the coding scheme, we have two special codes, representing entering the backbone ("[") and exiting the backbone ("]"). To show how their code length is represented in the probabilistic model, let us take a look at the transition probability from a non-backbone edge to a backbone edge $p(u \to v|\overline{\mathcal{B}} \cap \mathcal{I}(u)), (u \to v) \in \mathcal{B}$, and the transition probability from a backbone edge to a non-backbone edge $p(w \to z|B \cap \mathcal{I}(u)), (w \to z) \in \overline{\mathcal{B}}$:

$$p(u \to v|\overline{\mathcal{B}}) = p'(u \to v|\overline{\mathcal{B}}) \times p([|\overline{\mathcal{B}}_u), \text{ where } p'(u \to v|\overline{\mathcal{B}}) = \frac{p(u \to v|\overline{\mathcal{B}})}{p([|\overline{\mathcal{B}}_u)}, p([|\overline{\mathcal{B}}_u) = \sum_{(u,v')\in\mathcal{B}\cap\mathcal{O}(u)} p(u \to v'|\overline{\mathcal{B}})$$

$$p(w \to z|\mathcal{B}) = p'(w \to z|\mathcal{B}) \times p(]|\mathcal{B}_w), \text{ where } p'(w \to z|\mathcal{B}) = \frac{p(w \to z|\mathcal{B})}{p(]|\mathcal{B}_w)}, p(]|\mathcal{B}_w) = \sum_{(w,z')\in\overline{\mathcal{B}}\cap\mathcal{O}(w)} p(w \to z'|\mathcal{B})$$

Here $\overline{\mathcal{B}}_u = \overline{\mathcal{B}} \cap \mathcal{I}(u)$ and $\mathcal{B}_w = \mathcal{B} \cap \mathcal{I}(w)$ correspond to the set of incoming non-backbone edges to $u$ and incoming backbone edges to $w$. Table 2(b) illustrates $p([|\overline{\mathcal{B}}_u)$ and $p(]|\mathcal{B}_u)$. In other words, the code length "$[e_i$" ($e_i = (u \to v) \in \mathcal{B}$) corresponds to the transition probability from a non-backbone edge to a backbone edge: $p([|\overline{\mathcal{B}}_u)p'(e_i|\overline{\mathcal{B}}) = p(e_i|\overline{\mathcal{B}})$. Similarly, the code length of "$e_j]$" ($e_j = (w \to z) \in \overline{\mathcal{B}}$) corresponds to the transition probability from a backbone edge to a non-backbone edge: $p'(e_j|\mathcal{B}) \times p(]|\mathcal{B}_w) = p(e_j|\mathcal{B})$.

## III. Backbone Discovery based on Vertex Betweenness

In this section, we introduce a straightforward approach based on vertex betweenness and minimal steiner tree to discover backbone. This approach also serves as the basic benchmark for backbone discovery. Recall that, network backbone is a connected subgraph carrying the major traffic formed by shortest paths. In the meanwhile, vertex betweenness has been widely used to evaluate the importance of a vertex by the number of shortest paths passing through it. Given this, the straightforward solution for optimal $K$-backbone discovery problem is to consider $K$ vertices with highest betweenness as backbone vertices, since they tend to captures more information flow following shortest paths. Ideally, if these vertices are connected in the graph, its corresponding induced subgraph naturally forms the backbone, where the edges included in the induced subgraph are considered as backbone edges.

However, these $K$ vertices are not necessarily connected to each other. To obtain backbone structure, we want to build the connections among them while introducing minimal number of extra vertices. Since we focus on unweighted graph in this paper, this problem is essentially an instance of minimal steiner tree problem. The minimal steiner tree problem has been proved to be NP-hard, but an approximation algorithm has been introduced in [24]. Applying this method, we are able to gain a set of connected vertices as the superset of backbone vertices. The corresponding induced graph is treated as candidate backbone structure. To discover backbone with exactly $K$ vertices, we can utilize a refinement strategy to remove extra backbone vertices iteratively. Basically, in each iteration, we remove the vertex with smallest vertex betweenness from current graph. If remaining graph is not connected, we consider the removal of vertex with second smallest betweenness.

**Algorithm 1** BackboneDiscovery_VB($G = (V, \mathcal{E})$,$K$)

**Parameter:** $G$ is input network, $K$ is the backbone size

1: Compute vertex betweenness for each vertex using method in [5];
2: Select vertex set $V_s$ including $K$ vertices with largest vertex betweenness;
3: Construct minimal steiner tree $T = (V_T, E_T)$ on vertex set $V_s$ (i.e., $V_s \subseteq V_T$) by approximation algorithm [24];
4: $G_B$ is induced subgraph of $G$ on vertex set $V_T$;
5: $Q \leftarrow V_T$; {$Q$ is a queue which stores vertices in ascending order of their corresponding vertex betweenness}
6: **while** $|V_T| > K$ **do**
7: $u \leftarrow Q.pop\_front()$;
8: $G'_B$ is the induced subgraph of $G_B$ on vertex set $V_T \setminus \{u\}$;
9: **if** $G'_B$ is connected graph **then**
10: $V_T \leftarrow V_T \setminus \{u\}$;
11: $G_B \leftarrow G'_B$;
12: **else**
13: $Q.push\_back(u)$;
14: **end if**
15: **end while**
16: **return** $G_B$;

We sketch this approach in Algorithm 1. The algorithm mainly consists of two steps: candidate backbone discovery step (Line 1 to Line 4) and refinement step (Line 5 to Line 15). To begin with, the betweenness of each vertex is computed by fast approach in [5] (Line 1). We then select top $K$ vertices with largest betweenness and construct the minimal steiner tree $T = (V_T, E_T)$ over this vertex set (Line 2 to Line 3). Now, we have a set of connected vertices $V_T$ and its corresponding induced subgraph $G_B$ serves as the candidate backbone (Line 4 to Line 5). In the refinement step, we firstly store $V_T$ into a queue according to their corresponding betweenness in ascending order (Line 5). During main loop, in each iteration, we consider the first vertex $u$ in the queue (Line 7). If the removal of vertex $u$ and its incident edges will disconnect current graph, we push $u$ back into queue for further consideration (Line 13). Otherwise, the remaining graph $G'_B$ is used as current graph $G_B$ in next iteration. The refinement procedure proceeds until only $K$ vertices remains in the graph. This resulting graph is connected and returned as backbone.

**Computational Complexity:** In the candidate backbone discovery step, we take $O(|V||E|)$ time to compute vertex betweenness and $O(K^2|V|)$ time to build steiner tree. For refinement step, since we can remove at most one vertex from $V_T$ and connectivity checking on $G_B$ takes at most $O(|E_T|)$ in each iteration, total running time is $O((|V_T| - K)|E_T|)$ in the worst case. Putting both together, the overall computational complexity is $O(|V||E| + K^2|V|)$.

Instead of optimizing likelihood function, this straightforward approach only employs vertex betweenness to approximate the contribution made by each vertex to $L_B(\mathcal{P})$. In addition, this approach neglects the effect of edges on likelihood function by directly using the edge set of induced graph as backbone edges. Therefore, the discovered backbone may not necessarily maximize likelihood function $L_B(\mathcal{P})$. In the next section, we propose novel approaches to discover backbone by directly considering the optimality of likelihood function.

## IV. Optimizing Bimodal Markovian Model

In order to solve the optimal $K$-backbone discovery problem, two key questions should be considered: 1) Can we identify a set of $K$ connected vertices as candidate backbone vertices based on their contribution to likelihood of bimodal markovian model? 2) For a given set of $K$ vertices, how to extract an optimal backbone such that $L_B(\mathcal{P})$ is maximized? In other words, the edges in the induced subgraph of these $K$ vertices need to be classified into either backbone or non-backbone edges. It turns out the second question is more fundamental as it can directly contribute to the solution of the first one. Simply speaking, the solution of the second question offers an effective way to estimate individual vertex's contribution to the objective function $L_B(\mathcal{P})$ and thus is very helpful in selecting the backbone vertices.

**Backbone Edge Selection of $K$ Vertex Set:** In the following, we introduce a log-likelihood representation of the objective function $L_B(\mathcal{P})$ to simplify our problem in discovering the optimal backbone given a set of $K$ vertices. Formally, let $V_B$ be a subset of connected vertices in $G$ and $G[V_B]$ is corresponding induced graph of $G$. The backbone graphs $G_B = (V_B, \mathcal{B}) \subseteq G[V_B]$, i.e., the edges in the backbone can only come from the edges in the induced subgraph by vertex set $V_s$. Given path set $\mathcal{P}$, we want to extract edge set $E_B$ from $G[V_B]$ achieving maximum $L_B(\mathcal{P})$. This problem turns out to be rather challenging on undirected graph due to large search space and the edge consistent constraint that the categorization of both directed edges $(u \rightarrow v)$ and $(v \rightarrow u)$ of an undirected edge $(u, v)$ should be the same. In fact, it is even non-trivial to categorize individual backbone vertex's incoming edges into backbone and non-backbone without consistent constraint.

To facilitate maximizing $L_B(\mathcal{P})$ (Formula 2), we compare it with the likelihood of edge markovian model $L_M(\mathcal{P})$ (Formula 2). First of all, we rewrite likelihood of edge markovian model as $L_M(\mathcal{P}) =$

$$\prod_{e\in\mathcal{E}} p(e)^{N'_e} \prod_{u\notin V_B \wedge e'\in\mathcal{O}(u)} p(e'|e)^{N'_{ee'}} \prod_{u\in V_B \wedge e'\in\mathcal{O}(u)} p(e'|e)^{N'_{ee'}}$$

Now, we introduce the likelihood ratio (first two terms are canceled out for simplification) $LR(V_B) = \frac{L_B(\mathcal{P})}{L_M(\mathcal{P})} =$

$$\frac{\prod_{u\in V_B\wedge e\in\mathcal{O}(u)} p(e|\mathcal{B})^{N_{\mathcal{B}e}} \prod_{u\in V_B\wedge e\in\mathcal{O}(u)} p(e|\overline{\mathcal{B}})^{N_{\overline{\mathcal{B}}e}}}{\prod_{u\in V_B\wedge e'\in\mathcal{O}(u)} p(e'|e)^{N'_{ee'}}} = \prod_{u\in V_B} \frac{\prod_{e\in\mathcal{O}(u)} p(e|\mathcal{B})^{N_{\mathcal{B}e}} \prod_{e\in\mathcal{O}(u)} p(e|\overline{\mathcal{B}})^{N_{\overline{\mathcal{B}}e}}}{\prod_{e'\in\mathcal{I}(u),e\in\mathcal{O}(u)} p(e|e')^{N_{e'e}}} \quad (3)$$

Given graph $G$ and path set $\mathcal{P}$, $L_M(\mathcal{P})$ is a constant, assuming each of its parameters $p(e|e')$ is optimized for the maximal likelihood. Therefore, maximizing the likelihood ratio $L_B(\mathcal{P})/L_M(\mathcal{P})$ is equivalent to maximize $L_B(\mathcal{P})$. The following definition formalizes our problem:

*Definition 2:* (**Optimizing Vertex Set Likelihood Ratio Problem**) Given graph $G = (V, E)$ and path set $\mathcal{P}$, for any connected vertex set $V_B$, we would like to construct backbone subgraph $G_B = (V_B, \mathcal{B})$, where $\mathcal{B} \subseteq V_B \times V_B \cap E$ and $LR(V_B)$ (Formula 3) is maximized.

### A. Clustering Incoming Edges of Individual Vertex

To approach the problem(Definition 2), we start with relaxing the consistent constraint. In other words, for directed edges $(u \to v)$ and $(v \to u)$ with opposite direction, we assume that each of them can be determined independently to be a backbone or non-backbone edge. Then, the following rule is applied to enforce consistent constraint: we say $(u, v) \in E$ is backbone edge iff both $(u \to v)$ and $(v \to u)$ are backbone edges.

First of all, we rewrite $LR(V_B) = \sum_{u\in V_B} LR(u)$ where

$$LR(u) = \frac{\prod_{e\in\mathcal{O}(u)} p(e|\mathcal{B})^{N_{\mathcal{B}e}} \prod_{e\in\mathcal{O}(u)} p(e|\overline{\mathcal{B}})^{N_{\overline{\mathcal{B}}e}}}{\prod_{e'\in\mathcal{I}(u),e\in\mathcal{O}(u)} p(e|e')^{N_{e'e}}}$$

Based on aforementioned relaxation, we can see that $LR(u)$ is independent of $LR(v)$ if $u \neq v$, i.e., the optimality of $LR(u)$ will not affect the optimality of $LR(v)$. Therefore, optimizing each $LR(u)$ corresponding to vertex $u \in V_B$ individually is able to result in global maximization of $LR(V_B)$. Given this, our problem is converted to categorizing each vertex $u$'s incoming edges as backbone edges or non-backbone edges in order to optimize $LR(u)$. From the viewpoint of clustering, this essentially group each vertex's incoming edges into only two clusters (backbone or non-backbone).

Before proceeding to our solution, we first study how to compute optimal $L_B(u)$ assuming backbone edges are given. Essentially, we want to figure out the optimal $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$ leading to maximal $L_B(u)$. It is not hard to derive following result:

*Lemma 1:* For a backbone vertex $u$, assuming each incoming edge has been categorized as backbone or non-backbone, i.e., $\mathcal{B}$ and $\overline{\mathcal{B}}$, then the minimum of $-\log LR(u)$ is achieved when

$$p(e|\mathcal{B}) = \frac{N_{\mathcal{B}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\mathcal{B}e'}} \text{ and } p(e|\overline{\mathcal{B}}) = \frac{N_{\overline{\mathcal{B}}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\overline{\mathcal{B}}e'}} \quad (4)$$

**Proof Sketch:** The minimum of $-\log LR(u)$ is essentially finding the maximum value of likelihood function subject to certain probabilistic constraints. Using Lagrange multiplier method with two probabilistic constraints $\sum_{e\in\mathcal{O}(u)} p(e|\mathcal{B}) = 1$ and $\sum_{e\in\mathcal{O}(u)} p(e|\overline{\mathcal{B}}) = 1$, we are able to obtain the optimal values of $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$. □

**Algorithm for Edge Clustering:** We propose an iterative refinement algorithm to resolve this edge clustering problem on each vertex $u$. Initially, each incoming edge $e$ is randomly assigned to be backbone edge or non-backbone edge. Given such assignment, optimal value of $LR(u)$ can be computed based on corresponding optimal $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$ (Formula 4). In the subsequent iterations, we iteratively refine each edge's cluster membership in order to achieve a better value of $LR(u)$. The iterations terminate until no further improvement can be obtained. Interestingly, we will show that this method is essentially a K-Means under *Kullback-Leibler* divergence measure (K=2).

To further explain the algorithm, we express $LR(u)$ in negative log-likelihood format as follows (for simplicity, $\mathcal{I}$ and $\mathcal{O}$ are used to replace $\mathcal{I}(u)$ and $\mathcal{O}(u)$): $-\log LR(u) =$

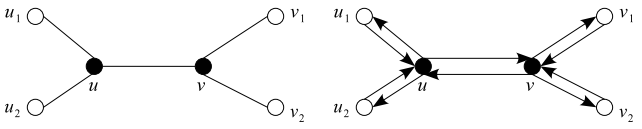
$$\sum_{e\in\mathcal{O}}\Big(\sum_{e'\in\mathcal{B}\cap\mathcal{I}} N_{e'e}\log\frac{p(e|e')}{p(e|\mathcal{B})} + \sum_{e'\in\overline{\mathcal{B}}\cap\mathcal{I}} N_{e'e}\log\frac{p(e|e')}{p(e|\overline{\mathcal{B}})}\Big) = \sum_{e'\in\mathcal{B}\cap\mathcal{I}} M_{e'}\sum_{e\in\mathcal{O}} p(e|e')\log\frac{p(e|e')}{p(e|\mathcal{B})} + \sum_{e'\in\overline{\mathcal{B}}\cap\mathcal{I}} M_{e'}\sum_{e\in\mathcal{O}} p(e|e')\log\frac{p(e|e')}{p(e|\overline{\mathcal{B}})}$$

where $M_{e'} = \sum_{e\in\mathcal{O}(u)} N_{e'e}$ is the total number of paths passing through edge $e'$ and then continue to one of its neighbors.

Indeed, $\sum_{e\in\mathcal{O}} p(e|e')\log\frac{p(e|e')}{p(e|\mathcal{B})}$ simply corresponds to the well-known *Kullback-Leibler* divergence between two distributions $(p(e_1|e'), \cdots, p(e_k|e'))$ and $(p(e_1|\mathcal{B}), \cdots, p(e_k|\mathcal{B}))$, where $e_1, \cdots, e_k \in \mathcal{O}(u)$ and $k = |\mathcal{O}(u)|$. Here, each incoming edge $e' \in \mathcal{I}(u)$ corresponds to a point with $k$ features $(p(e|e'), e \in \mathcal{O}(u))$ to be clustered. In addition, $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$ are interpreted as "centers" for the two clusters, the backbone clusters $\mathcal{B} \cap \mathcal{I}(u)$ and non-backbone clusters $\overline{\mathcal{B}} \cap \mathcal{I}(u)$. In this sense, the objective function $-\log LR(u)$ actually serves as the within-cluster distance. Now, we can utilize the K-Means type clustering to categorize incoming edges into backbone edges or non-backbone edges. In each refinement iteration, each incoming edges is assigned to the cluster who results in smallest KL-divergence. The procedure

**Algorithm 2** Bi-KL-Partition(Vertex $u$)

1: randomly partition the incoming edges of $u$ into $\mathcal{B}$ and $\overline{\mathcal{B}}$;
2: compute $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$, $e \in \mathcal{O}(u)$;
3: **repeat**
4: assign each incoming edge $e'$ to the cluster with the closest distance: $min(\sum_{e\in\mathcal{O}(u)} p(e|e') \log \frac{p(e|e')}{p(e|\mathcal{B})}$, $\sum_{e\in\mathcal{O}(u)} p(e|e') \log \frac{p(e|e')}{p(e|\overline{\mathcal{B}})})$;
5: calculate the two new centroids: $p(e|\mathcal{B}) = \frac{N_{\mathcal{B}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\mathcal{B}e'}}$ and $p(e|\overline{\mathcal{B}}) = \frac{N_{\overline{\mathcal{B}}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\overline{\mathcal{B}}e'}}$
6: **until** stop criteria is satisfied




(a) Undirected graph (b) Bidirected graph

Fig. 3: Observations related to backbone vertices

is outlined in Algorithm 2. Clearly, this algorithm will converge to a local minimum similar to classical K-Means algorithm.

### *B. Clustering Undirected Edges*

Utilizing above clustering method, we are able to convert the global optimization problem into the local problem of optimizing each vertex independently. Although the method is rather efficient, for each undirected edge $(u, v)$, both $(u \to v)$ and $(v \to u)$ cannot be guaranteed to be in the same category. Thus, such clustering only generates an upper bound of each individual vertex's benefit brought to bimodal markovian model. This upper bound will be utilized for selecting candidate backbone vertices in subsection V-A.

To address the edge consistent constraint, we further explore the property of edges incident to $V_B$. We observe that: 1) any undirected edge $e' = (u, u_1) \in E$ with only one endpoint $u$ in $V_B$ (see Figure 3(a)) is not a candidate to be a backbone edge; 2) for each undirected edge $e' = (u, v) \in E$ with both endpoints in $V_B$, we need $(u \to v)$ and $(v \to u)$ to be both in $\mathcal{B}$ or in $\overline{\mathcal{B}}$ (as shown in Figure 3(b)). To distinguish those two types of edges, we decompose the likelihood ratio $LR(V_B)$ into three parts:

$$-\log LR(V_B) = \sum_{e'\in(E\setminus V_B\times V_B)} \sum_{e\in\mathcal{O}(u)} N_{e'_v e} \log \frac{p(e|e'_v)}{p(e|\overline{\mathcal{B}})} + \sum_{e'\in V_B\times V_B\cap E} F(e'|\mathcal{B}) + \sum_{e'\in V_B\times V_B\cap E} F(e'|\overline{\mathcal{B}}) \quad (5)$$

where

$$F(e'|\mathcal{B}) = \sum_{e\in\mathcal{O}(u)} N_{e'_v e} \log \frac{p(e|e'_v)}{p(e|\mathcal{B})} + \sum_{e\in\mathcal{O}(v)} N_{e'_u e} \log \frac{p(e|e'_u)}{p(e|\mathcal{B})}$$

$$F(e'|\overline{\mathcal{B}}) = \sum_{e\in\mathcal{O}(u)} N_{e'_v e} \log \frac{p(e|e'_v)}{p(e|\overline{\mathcal{B}})} + \sum_{e\in\mathcal{O}(v)} N_{e'_u e} \log \frac{p(e|e'_u)}{p(e|\overline{\mathcal{B}})}$$

Note that, first term of Formula 5 relates to the probabilities of edges with one endpoint in $V_B$, $F(e'|\mathcal{B})$ and $F(e'|\overline{\mathcal{B}})$ consider the cases of edge $e'$ being backbone edge and non-backbone edge, respectively. Given this, we derive the following result:

*Lemma 2:* For a connected vertex set $V_B$, supporting all edges of induced graph $G[V_B]$ have been categorized as backbone or non-backbone, the minimum of $-\log LR(V_B)$ is achieved when

$$p(e|\mathcal{B}) = \frac{N_{\mathcal{B}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\mathcal{B}e'}} \text{ and } p(e|\overline{\mathcal{B}}) = \frac{N_{\overline{\mathcal{B}}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\overline{\mathcal{B}}e'}}$$

We apply the same technique based on Lagrange multiplier for obtaining Lemma 4, to gain the value of $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}}$ for optimizing $-\log LR(V_B)$. The probabilistic constraint here is the same, i.e., $\sum_{e\in\mathcal{O}(u)} p(e|\mathcal{B}) = 1$ and $\sum_{e\in\mathcal{O}(u)} p(e|\overline{\mathcal{B}}) = 1$.

The first term of Formula 5 is relatively stable on different backbone edge set as only $p(e'|\overline{\mathcal{B}})$ is involved. Therefore, the minimization of $-\log LR(V_B)$ can be approximately achieved by minimizing last two terms of Formula 5, serving as within-cluster distance. Given this, we describe our generalized *Bi-KL-Partition* algorithm (Algorithm 3) to solve it, which again has an interesting convergence property. The basic idea is to update the cluster membership of each candidate backbone edge $e' = (u, v) \in V_B \times V_B \cap E$ based on $F(e'|\mathcal{B})$ and $F(e'|\overline{\mathcal{B}})$ with the optimal $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$ for the current clusters $\mathcal{B}$ and $\overline{\mathcal{B}}$. In other words, $F(e'|\mathcal{B})$ and $F(e'|\mathcal{B})$ describe a generalized "distance" function from a point (edge) $e'$ to corresponding centroids.

*Lemma 3:* **(Convergence Property)** As we iteratively update membership of each edge $e' \in V_B \times V_B \cap E$ in Algorithm 3, the function $-\log LR(V_B)$ converges to a local optimum in finite iterations.

**Proof Sketch:** Let us use $F_i^1$ and $F_i^2$ to denote the values of objective function obtained from step 1 (Line 4) and step 2 (Line 5) at $i$-th iteration in Algorithm 3, respectively. Clearly, $F_i^2$ records the value of objective function at the end of iteration $i$. Assuming the algorithm just finishes iteration $i$, we will show that the value of $-\log LR(V_B)$ in iteration $i + 1$ is no greater than the value obtained from iteration $i$. Considering the step 1 in iteration $i + 1$, for each edge $e' \in V_B \times V_B \cap E$, its within-cluster distance is reduced (i.e., $min(F(e'|\mathcal{B}), F(e'|\overline{\mathcal{B}}))$), i.e., $F_i^2 \geq F_{i+1}^1$. Fixing backbone edge assignment, step 2 attempts to minimize the objective

**Algorithm 3** GBi-KL-Partition(Vertex Set $V_s$)

1: Assign each edge with only one end in $V_s$ to cluster $\overline{\mathcal{B}}$; randomly partition edges with both ends in $V_s$ into $\mathcal{B}$ and $\overline{\mathcal{B}}$;
2: for each vertex $u \in V_s$, compute optimal $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$, $e \in \mathcal{O}(u)$;
3: **repeat**
4: assign each edge $e'$ to the cluster with the closest distance: $min(F(e'|\mathcal{B}), F(e'|\overline{\mathcal{B}}))$;
5: calculate new centroids of two clusters for each vertex $u \in V_s$: $p(e|\mathcal{B}) = \frac{N_{\mathcal{B}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\mathcal{B}e'}}$ and $p(e|\overline{\mathcal{B}}) = \frac{N_{\overline{\mathcal{B}}e}}{\sum_{e'\in\mathcal{O}(u)} N_{\overline{\mathcal{B}}e'}}$
6: **until** stop criteria is satisfied

function by updating two clusters' centroids. In other words, we guarantee that $F_{i+1}^1 \geq F_{i+1}^2$. Considering both, we have $F_i^2 \geq F_i^1 \geq F_{i+1}^2$. In this sense, the value of $-\log LR(V_B)$ cannot be increased when the number of iteration increases. On the other hand, the number of possible edge assignment to be backbone or non-backbone is bounded by $2^{|E_s|}$ where $E_s$ is the edge set of induced graph based on vertex set $V_s$. This implies that the number of iterations in Algorithm 3 is at most $2^{|E_s|}$. Putting both together, the lemma holds. □

## V. Algorithms for Backbone Discovery

In this section, we will introduce two algorithms to discover the backbone with $K$ vertices for optimizing bimodal markovian model. The first algorithm tries to choose a set of connected vertices as backbone vertices by certain criteria, and then discover backbone edges among them in order to maximize $L_B(\mathcal{P})$. Interestingly, the first step can be converted to an instance of *maximum weight connected subgraph* (MCG) problem [12]. However, the selected vertices in first algorithm cannot guarantee to produce "good" backbone. We thus further propose second algorithm starting from above backbone and iteratively refine it to achieve better value of $L_B(\mathcal{P})$.

### A. Backbone Discovery based on Maximal Weight Connected Subgraph

The optimality of resulting backbone highly depends on the firstly selected backbone vertices. How to choose "good" backbone vertices is a challenging problem, as it is impossible to determine "goodness" of a set of backbone vertices in terms of $L_B(\mathcal{P})$ without backbone structure. To tackle it, we utilize the upper bound of their contribution to $L_B(\mathcal{P})$ in order to approximate the true "goodness". More specifically, we assign a score to each vertex which corresponds to maximal contribution by this vertex to the likelihood. For a set of connected vertices, their overall weight (sum of vertex weight) serves as an upper bound of their true likelihood. Larger upper bound potentially leads to better true value, thus we attempt to find a set of connected vertices with maximal upper bound to effectively approximate their true contribution. Then, *GBi-KL-Partition* procedure is used to discover backbone edges connecting them.

**Upper Bound:** Since most vertices will not be backbone vertices, we will rewrite our target maximal likelihood as

$$\log L_B(\mathcal{P}) = \log \frac{L_B(\mathcal{P})}{L_I(\mathcal{P})} + \log L_I(\mathcal{P})$$

where $L_I(\mathcal{P})$ is the likelihood function for the edge independence model. Given this, we introduce log-likelihood ratio $F(u)$ which represents the benefit for this vertex being a backbone vertex:

$$F(u) = \sum_{e \in \mathcal{O}(u)} \left( N_{\mathcal{B}e} \log \frac{p(e|\mathcal{B})}{p(e)} + N_{\overline{\mathcal{B}}e} \log \frac{p(e|\overline{\mathcal{B}})}{p(e)} \right) \quad (6)$$

For simplicity, we omit the benefit for edge $e$ to be the first edge in any shortest path (this portion is very small). It is easy to see that $\log \frac{L_B(\mathcal{P})}{L_I(\mathcal{P})} \approx \sum_{u \in V} F(u)$. In this case, we can invoke the *Bi-KL-Partition* procedure for each vertex $u$ and find its optimal bimodal markovian model. In the meanwhile, the values of $p(e|\mathcal{B})$, $p(e|\overline{\mathcal{B}})$ for each edge $e \in \mathcal{O}(u)$ are obtained. Now, we can calculate $F(u)$ for each vertex $u$ and assign it as corresponding vertex weight in graph $G$.

Given this, the problem of choosing a set of connected vertices with maximal sum of vertex weight is converted to an instance of *maximum weight connected graph* (MCG) problem [12].

*Definition 3:* (**Maximum Weight Connected Subgraph Problem** (MCG)) Given a graph $G = (V, E)$ where each vertex has a weight $w(v)$, and a positive integer $k$, maximum weight connected subgraph problem tries to identify a connected subgraph $G' = (V', E')$ where $|V'| = k$ and $\sum_{v \in V'} w(v)$ is maximized.

The MCG problem has been proven to be NP-hard, but an efficient heuristic algorithm has been proposed to find a *maximal* weight connected subgraph [13]. We apply this heuristic method, termed MCG, on our vertex-weighted graph to extract a set of connected vertices as backbone vertices. Following that, we utilize *GBi-KL-Partition* procedure to discover backbone edges for optimizing $L_B(\mathcal{P})$.

**Algorithm 4** BackboneDiscovery($G = (V, \mathcal{E})$,$K$)

**Parameter:** $G$ is input network, $K$ is the backbone size

1: **for each** $u \in V$ **do**
2: invoke *Bi-KL-Partition*(u);
3: compute $F(u)$;
4: **end for**
5: $V_B \leftarrow MCG(V, K)$;
6: $\mathcal{B} \leftarrow$ *GBi-KL-Partition*($V_B$);
7: $G_B \leftarrow (V_B, \mathcal{B})$;
8: **return** $G_B$;

The procedure to discover backbone based on maximal weight connected subgraph is outlined in Algorithm 4. To begin with, we invoke the *Bi-KL-Partition* procedure for each vertex $u$ to find its optimal bimodal markovian model and $p(e|\mathcal{B})$, $p(e|\overline{\mathcal{B}})$ for each $e \in \mathcal{O}(u)$ (Line 2). Then, we calculate $F(u)$ for each vertex $u$ as its weight in graph $G$ (Line 3). Following that, we use heuristic algorithm of MCG on vertex-weighted graph $G$ to identify backbone vertex set $V_B$ (Line 5). Finally, *GBi-KL-Partition* procedure is applied to extract backbone edges $\mathcal{B}$ among $V_B$ (Line 6).

**Computational Complexity:** In the main loop, procedure *Bi-KL-Partition* dominates computational cost. It takes at most $\sum_{u \in V} O(c|\mathcal{I}(u)| \times |\mathcal{O}(u)|) = O(\sum_{u \in V} c|\mathcal{N}(u)|^2) = O(c|V|d^2)$ time, where $c$ is the number of iterations repeated in *Bi-KL-Partition* and $d$ is the average degree of vertices in $G$. Moreover, the heuristic algorithm of MCG takes $O(K^2|V|^2 + K^2|V||E|)$ time. Then, the procedure *GBi-KL-Partition* costs $\sum_{u \in V_B} O(c'|\mathcal{N}(u)|^2) = O(Kd^2)$ time, where $c'$ is the number of iterations repeated in *GBi-KL-Partition*. Putting together, overall time complexity of this backbone discovery procedure is $O(|V|d^2 + K^2|V||E|)$.

However, discovered subset $V_B$ with maximal total weights $\sum_{u \in V_B} F(u)$ are not necessarily "good" backbone vertices for optimizing $L_B(\mathcal{P})$. This is because we neglect the constraint that both $(u \rightarrow v)$ and $(v \rightarrow u)$ of undirected edge $(u, v)$ should be both as backbone edges or non-backbone edges in vertex weight computation. In other words, if we apply procedure *GBi-KL-Partition*, then $\sum_{u \in V_B} F(u)$ may decrease by using the updated $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$.

### *B. Backbone Discovery by Iterative Refinement*

To address aforementioned issue, we propose a refinement strategy to improve the backbone in an iterative fashion. The basic idea is to first discover a subgraph as search starting point by Algorithm 4, then iteratively refine it by identifying a alternate backbone based on current one. Specially, in each iteration, we randomly abandon one vertex from current candidate backbone and add a neighboring vertex with maximal value of $F(u)$ (Formula 6) to form a alternate backbone. If new backbone leads to better value of $L_B(\mathcal{P})$, it would be used as current backbone for further refinement in the next iteration.

**Algorithm 5** IterativeRefinement($G = (V, \mathcal{E})$,$K$)

**Parameter:** $G$ is input network, $K$ is the backbone size
{Step 1: Preprocessing}
1: invoke *BackboneDiscovery*($G$,$K$) to obtain candidate backbone $G_B = (V_B, \mathcal{B})$;
2: **for each** $u \in V_B$ **do**
3: invoke *Bi-KL-Partition*(u);
4: compute $F(u)$;
5: **end for**
6: $W_H \leftarrow \sum_{u \in V_B} F(u)$;
7: $W_L \leftarrow \sum_{u \in V_B} F'(u)$; {$F'(u)$ is under the updated parameters of $p(e|\mathcal{B})$ and $p(e|\overline{\mathcal{B}})$}
8: $W \leftarrow W_L$;
{Step 2: Iterative Refinement}
9: **while** $|V(G)| > K \wedge W_H > W$ **do**
10: $V_s \leftarrow V_B \backslash \{v\}$ and $V \leftarrow V \setminus \{v\}$; {randomly remove one vertex $v$ from $V_B$ and $G$}
11: $u \leftarrow \arg\max_{u \in \mathcal{N}(V_B)} F(u)$;
12: $V_B \leftarrow V_B \cup \{u\}$;
13: $W_H \leftarrow W_H - F(v) + F(u)$;
14: $\mathcal{B} \leftarrow$ *GBi-KL-Partition*($V_B$);
15: $W_L = \sum_{u \in V_B} F'(u)$;
16: **if** $W < W_L$ **then**
17: $W \leftarrow W_L$;
18: $G_B \leftarrow (V_B, \mathcal{B})$; {keep the best result}
19: **end if**
20: **end while**
21: **return** $G_B$;

The overall procedure to discover backbone by iterative refinement scheme is outlined in Algorithm 5. It consists of two key steps: preprocessing step to generate candidate backbone by procedure *BackboneDiscovery*, and refinement step for improving backbone by local search. In *Step 1*, we invoke procedure *BackboneDiscovery* to provide a subgraph $G_B$ serving as starting point of refinement step (Line 1). For each vertex $u$ in $G_B$, we perform procedure *Bi-KL-Partition* to help compute $F(u)$ which is assigned as vertex weight (Line 2 to Line 5). The sum $W_H$ of those vertex weights represents the upper bound of maximal likelihood achieved by $G_B$ (Line 6). In the meanwhile, their true likelihoods are added together (i.e., $W_L$) to serve as lower bound of final backbone (Line 7). Moreover, $W$ is used to denote the true maximal benefit achieved so far. In *Step 2*, we try to seek alternate backbone vertices in an iterative manner for improving likelihood. To speed up the search process, in each iteration, we randomly remove a vertex $v$ from the current candidate backbone and graph $G$ (eliminate its further consideration) (Line 11). Then, we add neighboring vertex $u$ with maximal $F(u)$ to form new backbone vertex set (Line 12). We update upper bound $W_H$ and lower bound $W_L$ accordingly (Line 13 to Line 15). If a better backbone is found, current backbone will be kept as best result so far (Line 16 to Line 19). The refinement loop terminates until either the remaining graph is too small or the upper bound is smaller than the true likelihood achieved so far (Line 9). Note that since Step 2 involves a random search process, we can invoke it multiple times and choose the overall best backbone as our final result.

**Computational Complexity:** The cost of step 1 is dominated by procedure *BackboneDiscovery*, which takes $O(|V|d^2 + K^2|V||E|)$ times, where $d$ is the average degree of vertices in $G$. For step 2, in each iteration, it takes $O(dK)$ time to select best neighboring vertex to form a alternate backbone vertex set (Line 11), and takes $O(Kd^2)$ times to perform *GBi-KL-Partition* supposing the number of iteration required in the procedure is a small constant. Given this, assuming $c$ iterations are needed in the refinement loop, step 2 takes $O(cKd^2)$ time in total. Overall, the time complexity of iterative refinement scheme to discover backbone is $O(|V|d^2 + K^2|V||E|)$.

Finally, we note that in the above algorithm, we do not consider how to compute path set $\mathcal{P}$ and derive the basic probabilities, such as $p(e)$ and $p(e|e')$ in edge independence model and edge markovian model, respectively. In the worst case, assuming path set $\mathcal{P}$ includes the shortest path for each pair of vertices, the most straightforward way is to invoke $|V|$ times BFS procedures in $O(|V| \times (|V| + |E|))$ time. When we enumerate these paths, we can online compute $N_e$ (edge betweenness) and $N_{e'e}$, so there is no need to materialize the entire path set $\mathcal{P}$. The overall computational time is $O(|V| \times (|V| + |E|))$.

## VI. Empirical Study

In this section, we evaluate the performance of proposed backbone discovery algorithms:

1) the basic backbone discovery based on vertex betweenness (referred to it as **VB**);

2) the backbone discovery approach based on maximum weight connected graph (referred to it as **MCG**);

3) the backbone discovery approach based on iterative refinement (referred to it as **ITER**).

First, we study the performance of our methods from three aspects: modeling accuracy, parameter reduction and edge size in the backbone. Note that, modeling accuracy is measured by ratio between edge markovian model and bimodal markovian model, which is expressed as the logarithmic value of edge markovian model's (**EM**) likelihood divides the one of bimodal markovian model extracted by VB, MCG and ITER (denoted by $EM/VB$, $EM/MCG$ and $EM/ITER$). The closer to 1 modeling accuracy is, the better results our methods achieve. Then, we study the efficiency of our methods on large random graphs with power law degree distribution. Finally, we perform case studies on co-author network and human protein-protein interaction (PPI) network to further demonstrate our approaches.

We implemented all algorithms using C++ and the Standard Template Library (STL). All experiments were conducted on a 2.0GHz Dual Core AMD Opteron CPU with 4.0GB RAM running Linux.

### *A. Real Datasets*

We study the bimodal markovian model on three real-world datasets, including one biological network and two co-author networks on different research fields:

**Yeast**: The yeast protein-protein interaction network [4] includes $2361$ vertices and $6646$ edges. Each vertex indicates one protein and each edge denotes the interaction between two proteins. The network's average pairwise shortest distance is $4.4$.

**Net**: The coauthorship network [15] of researchers who work in the field of network theory and experimentation, as collected by M. Newman. An edge joins two authors if and only if these two have collaborated on at least one paper in this area. Since the entire network consists of several disconnected components, we extract the largest connected component with $379$ vertices and $1828$ edges for the experiment. The network's average pairwise shortest distance is $6.1$.

**DM**: The co-author network in the field of data mining [21] which consists of $2000$ researchers. Each of the $10615$ edges indicates that the authors have co-authored at least one paper. The network's average pairwise shortest distance is $4.6$.

### *B. Performance of Bimodal Markovian Model*

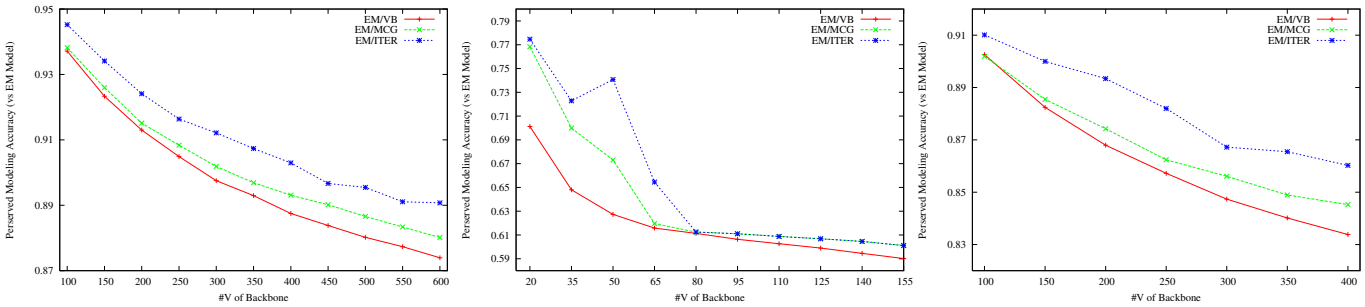
In the following experiments, we investigate the performance of bimodal markovian model from several aspects.

**Preserved Modeling Accuracy:** To verify the performance of bimodal markovian model on preserving modeling accuracy of edge markovian model while reducing its number of parameters, we apply all approaches VB, MCG and ITER on above 3 datasets. The size of backbone is supposed to be small, thus we vary the number of vertices in backbone in the range from $5\%$ to around $20\%$ of the number of vertices in original networks. Especially, for Yeast, the number of vertices in the backbone varies from $100$ to $600$. The number of vertices in the backbone are set in the range from $20$ to $155$ and the range from $100$ to $400$, respectively, for datasets Net and DM. We make the following observations:

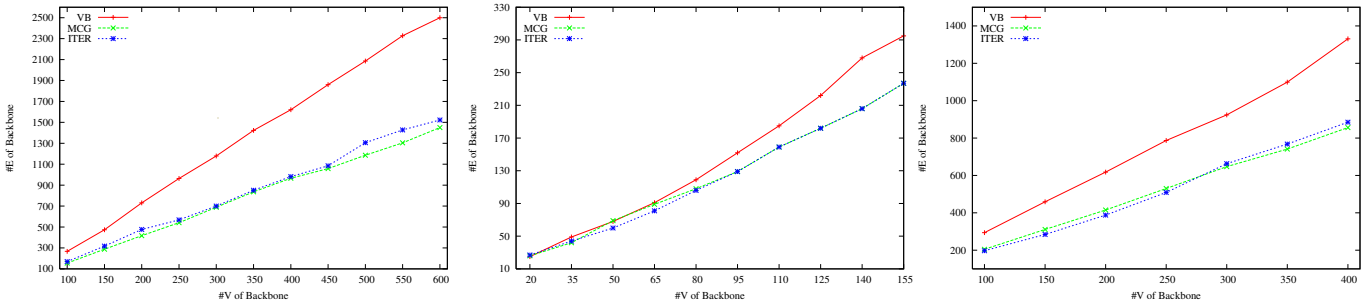
Figure 4(a), Figure 4(b) and Figure 4(c) show that bimodal markovian model based on backbones discovered by both ITER and MCG greatly preserve the modeling accuracy of edge markovian model (refer to it as **EM**). Their modeling accuracies are consistently better than the one of backbones discovered by VB. In particular, for datasets Yeast and DM, the ratio between EM and bimodal markovian model based on backbones discovered by MCG and ITER are higher or very close to $90\%$ in most of cases. Among two methods, ITER utilizing iterative refinement strategy consistently achieves better results than MCG on all datasets. As the number of vertices in the backbone increases, the ratios of all methods are slowly decreased in Yeast and DM. More vertices are considered as backbone vertices, the simpler the model becomes (this is confirmed by Figure 4(g) and Figure 4(i)). This directly leads to the coarser representation of paths and the decrease of bimodal markovian model's likelihood. Interestingly, unlike the consistently decreasing trend observed from EM/VB and EM/MCG, the results of ITER on Net do not consistently decrease with the increasing number of vertices in the backbones. This phenomena might be explained by two reasons: 1) our ITER method employing local search tries to achieve a local optimal solution while not global one; 2) larger backbone is possible to connect some important vertices which simplifies edge markovian model at the expense of less modeling accuracy. Therefore, it is reasonable to see the climbing trend from the data point corresponding to $35$-vertex backbone to $50$-vertex backbone.

**Backbone Complexity:** We evaluate the backbone complexity based on the number of edges in the discovered backbone. From Figure 4(d), Figure 4(e) and Figure 4(f), we can see that the backbones generated by MCG and ITER are rather sparse. Overall, the edge density (i.e., $|E|/|V|$) of backbones discovered by both MCG and ITER are very close to or small than $2.5$ on all three datasets. The edge density of backbones in dataset Net is rather close to $1$, which suggests that discovered backbone is tree-like structure. However, the edge density of backbone discovered by VB is much denser, which are around $4$ and $3.5$ in Yeast and DM. In addition, though ITER achieves better results than MCG regarding the number of parameters (Figure 4(g), Figure 4(h) and Figure 4(i)), the number of edges in the backbones generated by ITER is not guaranteed to be smaller than that of MCG (see Figure 4(e) and Figure 4(f)). This is reasonable because the parameter reduction relies on the number of edges incident to backbone vertices while is independent of the number of backbone edges. In other words, for each backbone vertex $v$ with immediate neighbors $N(v)$, no matter how many incident edges are backbone edges, the number of parameters in bimodal markovian model is fixed to be $2 \times |N(v)|$.

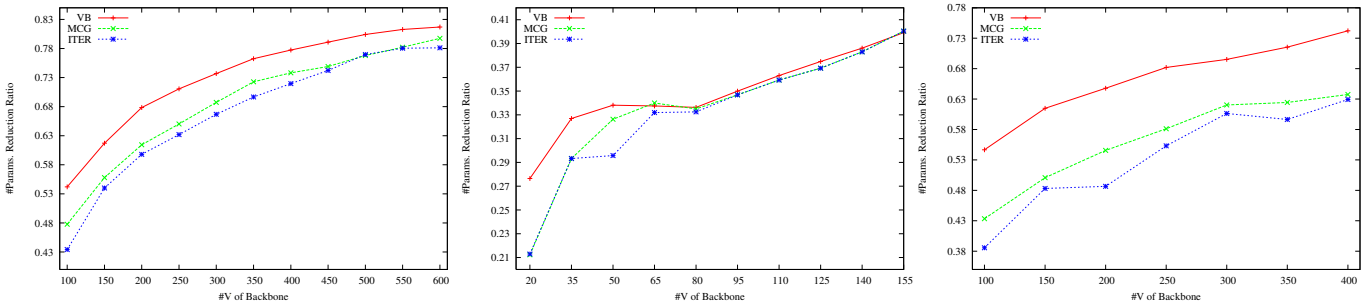
**Parameter Reduction:** For all three datasets, we compare



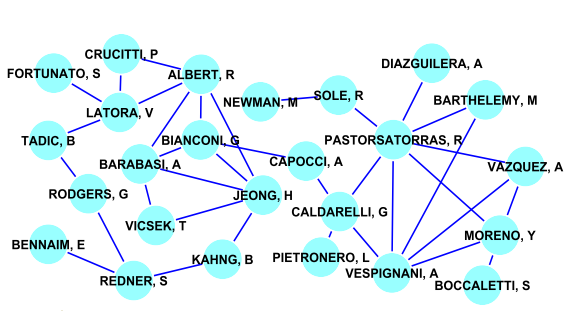

(a) Modeling accuracy on Yeast (b) Modeling accuracy on Net (c) Modeling accuracy on DM



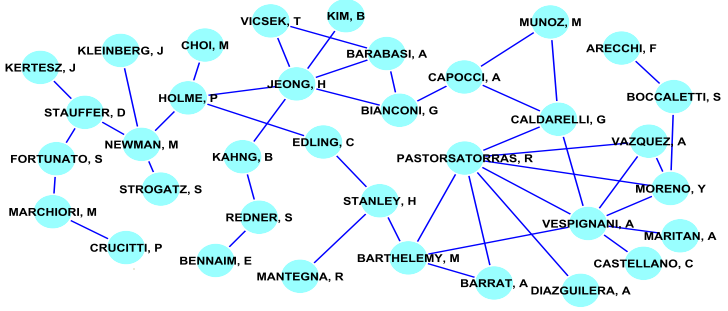

(d) #E of backbone (Yeast) (e) #E of backbone (Net) (f) #E of backbone (DM)



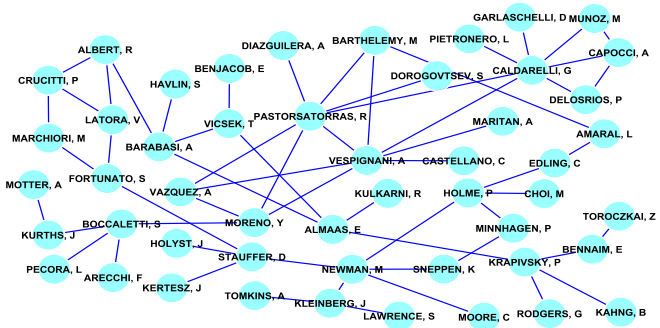

(g) #Param. reduction on Yeast (h) #Param. reduction on Net (i) #Param. reduction on DM



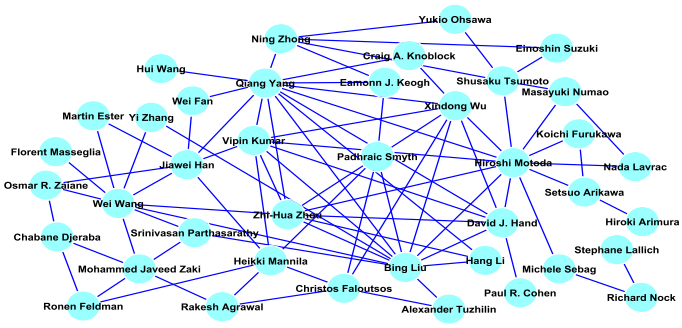

(j) 25-vertex backbone (Net)




(k) 35-vertex backbone (Net)




(l) 50-vertex backbone (Net)




(m) 40-vertex backbone (DM)

Fig. 4: Backbone discovery on real-world datasets

the parameters reduction ratio between edge markovian model and bimodal markovian models based on backbones discovered by VB, ITER and MCG in Figure 4(g), Figure 4(h) and Figure 4(i), respectively. The parameter reduction ratio is computed by $\frac{\#Param_{EM}-\#Param_{BM}}{\#Param_{EM}}$ where $\#Param_{EM}$ and $\#Param_{BM}$ denote the number of parameters used in edge markovian model and bimodal markovian model, respectively. As we can see, all three approaches VB, ITER and MCG dramatically reduce the number of parameters in edge markovian model. Among them, it is interesting to see that VB outperforms both ITER and MCG in all settings. In VB, high-degree vertices tend to be selected as backbone vertices since they have high probability to lie in many shortest paths and have greater vertex betweenness. Therefore, more conditional probabilities of edges incident to these vertices would be simplified compared to other two methods. In datasets Yeast and DM, VB on average even reduces $73\%$ and $66\%$ parameters in EM model, respectively. Both MCG and ITER also achieve good parameter reduction ratio. In Yeast and DM, even though $5\%$ of vertices in original graphs are backbone vertices, around half of parameters in EM model are reduced while large portion of modeling accuracy is preserved. Also, more parameters are reduced by MCG than that of ITER in most of cases. As mentioned before, when the number of vertices in the backbones increases, more incident edges' conditional probabilities tend to be simplified.

Finally, we note that in general, the right backbone size is application-dependent. Without any prior information, based on experimental results on those $3$ datasets, it seems that using around $10\%$ of vertices in original networks as backbone vertices is a reasonable choice. There are significant losses of modeling accuracy for larger backbones and the number of parameter reduction is not high for smaller backbones.

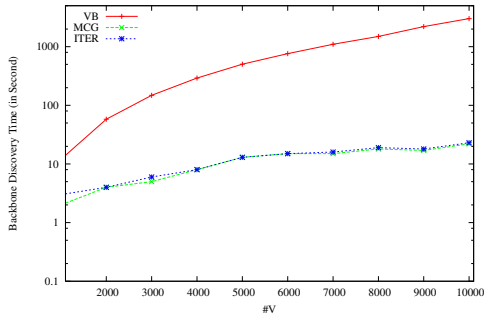



Fig. 7: Running time for power-law graphs

### C. Performance Study

To verify the scalability of our approach, we test a set of random undirected graphs with power-law degree distribution. The graphs vary in size from $10K$ to $100K$ vertices and we set the edge density to be $4$. We specified each backbone to have $100$ vertices.

We decompose the running time into two parts: preprocessing time (i.e., computing shortest paths and calculating edge or segment betweenness for basic probabilities) and backbone discovery time. Figure 7 shows the backbone discovery time of VB, ITER and MCG for random graphs with power-law degree distribution. These results clearly demonstrate the scalability of approaches MCG and ITER. In particular, the running time of ITER is very close to MCG, because the extra computational cost of ITER (Algorithm 5) compared to MCG only depends on the size of backbone which is supposed to be small. This also confirms our time complexity analysis on ITER and MCG. Both are much faster than straightforward method VB by a factor of $59$, due to the high cost of building minimal steiner tree in VB. The preprocessing step, especially computing the pairwise shortest distances, as expected is more expensive. The preprocessing time of all methods varies from $30$ seconds to $121$ minutes. We note that sampling seems to be an effective approach to avoid the full pairwise computation, thus speeding up the preprocessing time. It is beyond the scope of this paper and will be investigated in future work.

## VII. Case Studies

In this section, we report network backbones in co-author networks and the PPI network discovered by ITER method.

**Co-author Networks (Net and DM):** Figure 4(j), Figure 4(k) and Figure 4(l) show the discovered backbones with the number of backbone vertices varying from $25$ to $50$. These figures precisely depict the backbone generation and growth process. In Figure 4(j), the backbone is a sparse subgraph with only $25$ vertices. Comparing this backbone to the full connected component[1], we see that these vertices serve as the essential connectors among several "small world" components. As the backbone expands to $35$ vertices in Figure 4(k), most of the earlier vertices are retained, while several important researchers, such as J. Kleinberg and P. Holme are added. Then, the backbone is slightly expanded from Figure 4(k) to Figure 4(l). Another interesting observation is that some of the researchers in the backbone are not necessarily the best-known scientists nor do they have a high number of collaborators. For instance, *C. Edling* in the backbone only has $5$ collaborators in this network. In contrast to the traditional research which aims to discover highly correlated components, our backbone model studies complex networks from a new angle. The discovered backbone essentially captures the communication path among different highly correlated communities.

From Figure 4(m), we can see that many of the discovered researchers, like *Jiawei Han*, *Rakesh Agrawal* and *Christos Faloutsos* are prominent scientists in data mining. Compared

[1]A figure of the largest connected component can be found at http://www-personal.umich.edu/∼mejn/centrality/

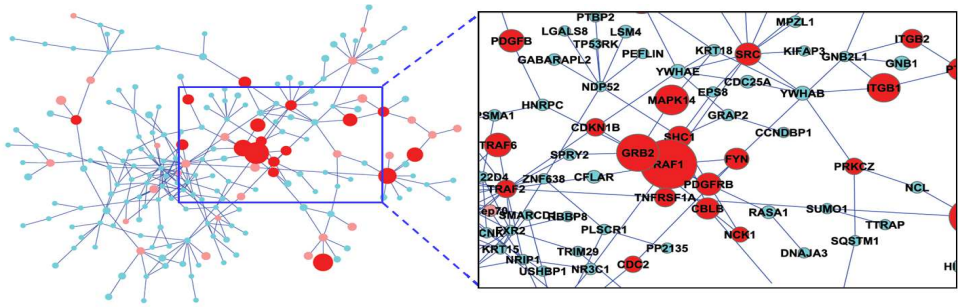



Fig. 5: Visualization of the 200 gene backbone for the PPI network. Red color indicates genes involving in at least 4 KEGG pathways. The larger the nodes, the more KEGG pathways they are involved in.

| Functions | P-value |
|---|---|
| Cell Death | 3.93E-28 |
| Gene Expression | 1.95E-17 |
| Cellular Growth and Proliferation | 2.39E-16 |
| Cell Cycle | 2.42E-16 |
| Cellular Development | 4.74E-12 |
| Cellular Movement | 6.63E-11 |
| Cellular Function and Maintenance | 9.58E-11 |
| DNA Replication, Recombination, and Repair | 8.77E-09 |
| Cell Morphology | 6.2E-08 |
| Cell-To-Cell Signaling and Interaction | 2.19E-07 |
| Hematological System Development and Function | 4.05E-07 |
| Hematopoiesis | 4.05E-07 |
| Post-Translational Modification | 1.35E-06 |
| Connective Tissue Development and Function | 3.04E-06 |
| Cell Signaling | 4.68E-06 |

| Pathways | P-value |
|---|---|
| PPAR Signaling | 4.90E-10 |
| Molecular Mechanisms of Cancer | 1.15E-09 |
| Cell Cycle: G1/S Checkpoint Regulation | 1.17E-09 |
| Ephrin Receptor Signaling | 3.80E-09 |
| JAK/Stat Signaling | 4.07E-09 |
| PAK Signaling | 8.91E-09 |
| Neuregulin Signaling | 1.15E-08 |
| PTEN Signaling | 1.48E-08 |
| Small Cell Lung Cancer Signaling | 1.86E-08 |
| Phospholipase C Signaling | 2.24E-08 |
| PI3K/AKT Signaling | 2.24E-08 |
| Cell Cycle: G2/M DNA Damage Checkpoint Regulation | 2.40E-08 |
| Role of PKR in Interferon Induction and Antiviral Response | 2.95E-08 |
| IL-15 Signaling | 6.46E-08 |
| 14-3-3-mediated Signaling | 1.12E-07 |

| Diseases and disorders | P-value |
|---|---|
| Cancer | 1.53E-14 |
| Genetic Disorder | 1.93E-13 |
| Respiratory Disease | 1.93E-13 |
| Hematological Disease | 1.08E-12 |
| Skeletal and Muscular Disorders | 9.69E-12 |
| Gastrointestinal Disease | 1.02E-11 |
| Renal and Urological Disease | 1.14E-10 |
| Immunological Disease | 1.23E-10 |
| Neurological Disease | 6E-10 |
| Reproductive System Disease | 1.78E-09 |
| Organismal Injury and Abnormalities | 8.75E-08 |
| Dermatological Diseases and Conditions | 1.38E-07 |
| Hepatic System Disease | 2.36E-07 |
| Inflammatory Diseases | 2.48E-04 |
| Cardiovascular Diseases | 4.71E-04 |

Fig. 6: Function and pathway enrichment analysis on the 200 backbone genes by IPA. Left: Top 15 enriched functional categories. Middle: Top 15 enriched canonical pathways. Right: Top 15 diseases and disorders.

to relatively sparse backbones on Net, the backbone from the data mining co-author network is denser. This indicates the different collaboration styles in different research fields. In the field of network theory and experiment, researchers tend to collaborate within small groups while a few of them have connections among different groups. However, in the data mining area, many scientists work in several different directions which results in more wide-ranging collaborations among them.

**Human PPI Network:** We applied the backbone discovery algorithm (**ITER**) on the human protein-protein interaction (PPI) dataset obtained from [20] to identify backbone of the human PPI. This dataset consists of 3133 genes and 12298 edges indicating relationships among them. Our algorithm returned the genes in the backbone with the user specified size (which is 200 in our test). As shown in Figure 5, the backbone genes contain many well known and important genes in cellular signaling transduction pathways including both kinases (e.g., *RAF1*, *MAPK14*, *SRC* and *FYN*) and receptors (e.g., *TRAF6*, *PDGFRB*) as well as signaling molecules such as *PDGFB*. Unlike traditional gene set discovery studies for which we expect to obtain a group of genes with a small set of specifically enriched functions or pathways, we expect that the backbones genes of the PPI network would be engaged in many different functions and possibly pathways. The functional and pathway analysis using tools such as the Ingenuity Pathway Analysis (IPA) indeed confirmed our expectation. As shown in Figure 6, the 200 genes are highly enriched with a wide spectrum of important biological functions and are related to many different diseases with high statistical significance. Moreover, they are involved in a large number of pathways, which is very rare for a gene list of this size. For the IPA canonical pathways, the 200 genes show enrichment with p-values (of the Fisher's exact test used by IPA) less than 0.0001 for more than 70 different pathways. These observations suggest that many backbone genes may involve in more than one pathways. Indeed we found that out of the 200 genes (of which 195 can be mapped to KEGG gene ids) 47 are involved in at least *four* KEGG pathways. This is a highly significant enrichment comparing to the fact that a total 1,100 such genes can be found among the entire genome of 19,076 annotated human genes in the KEGG database ($p < 1.9 \times 10^{-17}$ for hypergeometric test). It can be conceived that perturbation on these genes can lead to serious disruption of important biological functions, which implies the involvement in diseases in human. This is also confirmed as shown in Figure 6. Therefore, our experimental study on the PPI network backbone discovery demonstrated the effectiveness of our approach and its great potential as a new gene ranking tool.

## VIII. Conclusion

In this paper, we introduce a new backbone discovery problem and propose novel discovering approaches based on vertex betweenness and $KL$-divergence. We believe the backbone approach opened a new way to study complex networks and systems, and also presents many new research

questions for both data mining and complex network research: How do network backbone and modularity coexists and how they affect each other? How robust is the backbone, and how will it change? What information is carried in the backbone? We plan to work on these fascinating questions in the future.

## REFERENCES


[1] Charu C. Aggarwal and Haixun Wang. A survey of clustering algorithms for graph data. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40, pages 275–301. Springer US, 2010.

[2] E. Almaas, B. Kovacs, T. Vicsek, Z. N. Oltvai, , and A.-L. Barabasi. Global organization of metabolic fluxes in the bacterium escherichia coli. *Nature*, 427:839–843, 2004.

[3] Remo Badii and Antonio Politi. *Complexity : Hierarchical Structures and Scaling in Physics (Cambridge Nonlinear Science Series)*. Cambridge University Press, 1999.

[4] Vladimir Batagelj and Andrej Mrvar. Pajek datasets http://vlado.fmf.uni-lj.si/pub/networks/data/, 2006.

[5] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[6] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley and Sons, 1991.

[7] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, June 2002.

[8] Daniel Hennessey, Daniel Brooks, Alex Fridman, and David Breen. A simplification algorithm for visualizing the structure of complex graphs. In *Proceedings of the 2008 12th International Conference Information Visualisation*, pages 616–625, 2008.

[9] David Hutchinson, Anil Maheshwari, and Norbert Zeh. An external memory data structure for shortest path queries. *Discrete Appl. Math.*, 2003.

[10] Chinmay Karande, Kumar Chellapilla, and Reid Andersen. Speeding up algorithms on compressed web graphs. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 272–281, 2009.

[11] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The structure of information pathways in social communication networks. In *KDD, To appear*, 2008.

[12] Heungsoon Felix Lee and Daniel R. Dooly. Algorithms for the constrained maximum-weight connected graph problem. *Naval Research Logistics*, 43(7):985–1008, 1996.

[13] Heungsoon Felix Lee and Daniel R. Dooly. Decomposition algorithms for the maximum-weight connected graph problem. *Naval Research Logistics*, 45(8):817–837, 1998.

[14] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. In *SIGMOD '08*, pages 419–432, 2008.

[15] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[16] Davood Rafiei and Stephen Curial. Effectively visualizing large networks through sampling. In *IEEE Visualization*, page 48, 2005.

[17] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105:1118–1123, 2008.

[18] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local graph sparsification for scalable clustering. In *SIGMOD'11*, 2011.

[19] Ageles Serrano, Maria Boguna, and Alessandro Vespignani. Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106(16):6483–6488, 2009.

[20] Ulrich et. al. Stelzl. A human protein-protein interaction network: A resource for annotating the proteome. 122(6):957–968, September 2005.

[21] Jie Tang, Ruoming Jin, and Jing Zhang. A topic modeling approach and its integration into the random walk framework for academic search. In *ICDM '08*, pages 1055–1060, 2008.

[22] Yufei Tao, Cheng Sheng, and Jian Pei. On k-skip shortest paths. In *SIGMOD '11*, pages 421–432, 2011.

[23] Hannu Toivonen, Sébastien Mahler, and Fang Zhou. A framework for path-oriented network simplification. In *IDA '10*, pages 220–231, 2010.

[24] B.Y. Wu and K.M. Chao. *Spanning trees and optimization problems*. Discrete mathematics and its applications. Chapman & Hall/CRC, 2004.

[25] Fang Zhou, Hannu Toivonen, and Sébastien Mahler. Network simplification with minimal loss of connectivity. In *ICDM '10*, 2010.