2011 11th IEEE International Conference on Data Mining

Distance Preserving Graph Simplification

Ning Ruan
Department of Computer Science
Kent State University
Kent, OH, USA
nruan@cs.kent.edu

Ruoming Jin
Department of Computer Science
Kent State University
Kent, OH, USA
jin@cs.kent.edu

Yan Huang
Department of CSE
University of North Texas
Denton, TX, USA
huangyan@unt.edu

***Abstract*—Large graphs are difficult to represent, visualize, and understand. In this paper, we introduce "gate graph" - a new approach to perform graph simplification. A gate graph provides a simplified topological view of the original graph. Specifically, we construct a gate graph from a large graph so that for any "non-local" vertex pair (distance greater than some threshold) in the original graph, their shortest-path distance can be recovered by consecutive "local" walks through the gate vertices in the gate graph. We perform a theoretical investigation on the gate-vertex set discovery problem. We characterize its computational complexity and reveal the upper bound of minimum gate-vertex set using VC-dimension theory. We propose an efficient mining algorithm to discover a gate-vertex set with guaranteed logarithmic bound. The detailed experimental results using both real and synthetic graphs demonstrate the effectiveness and efficiency of our approach.**

***Keywords*-Graph Simplification; Gate Vertices; Gate Graph; VC-Dimension; Set Cover Problem;**

I. Introduction

Reducing graph complexity or graph simplification is becoming an increasingly important research topic [1, 17, 21, 19, 15]. It can be very challenging to grasp a graph even with thousands of vertices. Graph simplification targets at reducing edges, vertices, or extracting a high level abstraction of the original graph so that the overall complexity of the graph is lowered while certain essential properties of the graph can still be maintained. It has been shown that such simplification can help understand the underlying structure of the graph [8, 1]; better visualize graph topology [13, 8]; and speed up graph computations [2, 12, 5, 11, 18, 15].

In this paper, we investigate how to extract a set of vertices from a graph such that the vertex locations and relationships not only help to preserve the distance measure of the original graph, but also provide a simplified topological view of the entire graph. Intuitively, these vertices can be considered to distribute rather "evenly" in the graphs in order to reflect its overall topological structure. For any "non-local" vertex pair (distance higher than some threshold), their shortest-path distance can be recovered by consecutive "local" walks through these vertices. Basically, these vertices can be viewed as the key intermediate highlights of the long-range (shortest-path distance) connections in the entire graph. In other words, for any vertex to travel to another vertex beyond its local range, it can always use a sequence of these discovered vertices (each one being in the local range of its predecessor) to recover its shortest path distance to the destination. Thus, conceptually, this set of vertices form a "wrap" surrounding any vertex in the original graph, so that any long range (shortest-path) traffic goes through the "wrap". From this perspective, these vertices are referred to as the *gate* vertices and our problem is referred to as the *gate-vertex set discovery* problem. Furthermore, these gate vertices can be connected together using only "local" links to form a *gate graph*. A gate graph not only reveals the underlying highway structure, but also can serve as a simplified view of the entire graph. Gate-vertex set and gate graph have many applications in graph visualization [9, 4] and shortest path distance computation [18].

A. Problem Definition

Let $G = (V, E)$ be an unweighted and undirected graph, where $V = \{1, 2, ..., N\}$ is the vertex set and $E \subseteq V \times V$ is the edge set of graph G. We use (u, v) to denote the edge from vertex u to vertex v, and $P_{v_0,v_p} = (v_0, v_1, ..., v_p)$ to denote a simple path from vertex v_0 to vertex v_p. The length of a simple path in unweighted graph is the number of edges in the path. For weighted graph, each edge $e \in E$ is assigned a weight $w(e)$. The length of a simple path in a weighted graph is the sum of weights from each edge in the path. The distance from vertex u to vertex v in the graph G is denoted as $d(u, v)$, which is the minimal length of all paths from u to v.

Given a user-defined threshold $\epsilon > 0$, for any pair of *connected* vertices u and v, if their distance is strictly less than ϵ ($d(u, v) < \epsilon$), we refer to them as a **local pair**, and their distance is referred to as a **local distance**; if their distance is higher than or equal to ϵ but finite, we refer to them as a **non-local pair**, and their distance is referred to as a **non-local distance**. In addition, we also refer to ϵ as the *locality* parameter or the *granularity* parameter.

Definition 1: (**Mimimum Gate-Vertex Set Discovery (MGS) Problem**) Given an unweighted and undirected graph $G = (V, E)$ and user-defined threshold $\epsilon > 0$, vertex set $V^* \subseteq V$ is called a gate-vertex set if V^* satisfies the following property: for any **non-local pair** u and v ($d(u, v) \geq \epsilon$), there is a vertex sequence formed by consecutive **local pairs** from u to v, $(u, v_1, v_2, \cdots, v_k, v)$ where $v_1, v_2, \cdots, v_k \in V^*$, such that $d(u, v_1) < \epsilon$, $d(v_1, v_2) < \epsilon$, $\cdots$, $d(v_k, v) < \epsilon$, and $d(u, v_1) + d(v_1, v_2) + \cdots + d(v_k, v) = d(u, v)$. The gate vertex

1550-4786/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDM.2011.57

set discovery problem seeks a set of gate vertices with smallest cardinality.

In other words, the gate-vertex set guarantees that the distance between any *non-local pair* u and v can be recovered using the distances from source vertex u to a gate vertex v_1, between consecutive gate vertices, and from the last gate vertex v_k to the destination vertex v. These are all local distances. Here, the local distance requirement for recovering any non-local distance enables the gate vertices to reflect enough details of the underlying topology of the original graph G. Based on the gate-vertex set, we can further define the *gate graph*.

Definition 2: **(Mimimum Gate Graph Discovery (MGG) Problem)** Given an unweighted and undirected graph $G = (V, E)$ and a gate-vertex set V^* ($V^* \subseteq V$) with respect to parameter ϵ, the gate graph $G^* = (V^*, E^*, W)$ is a weighted and undirected graph where W assign each edge $e \in E^*$ a weight $w(e)$, such that for any non-local pair u and v in G ($d(u, v) \geq \epsilon$), we have $d(u, v) =$

$$min_{d(u,x)<\epsilon \wedge d(y,v)<\epsilon \wedge x,y \in V^*} d(u, x) + d(x, y|G^*) + d(y, v);$$

Here $d(x, y|G^*)$ is the distance between x and y in the *weighted* gate graph. The gate graph discovery problem seeks the gate graph with the minimum number of edges. Note that the edges in the gate graph may not belong to the original graph.

The gate vertices and gate graph are related to the recently proposed k-skip graph [18] which represents the latest effort in using highway structure to reduce the search space of the well-known shortest distance computation method, *Reach* [6]. Basically, each shortest path is succinctly represented by a subset of vertices, namely k-skip shortest path, such that it should contain at least one vertex out of every k consecutive vertices in the original shortest path. In other words, k-skip shortest path compactly describes original shortest path by sampling its vertices with a rate of at least $1/k$. Tao *et al.* show that those sampled vertices can be utilized to speed up the distance computation. Following the similar spirit, gate-vertex set and gate graph directly highlight the long-range connection between vertices, and can also serve as a highway structure in the general graph.

We note that the k-skip cover and gate vertices are conceptually close but different. The k-skip cover intends to uniformly sample vertices in shortest paths, whereas the gate-vertex set tries to recover shortest-path distance using intermediary vertices (and local walks). More importantly, the k-skip cover focuses on the road network and implicitly assume *there is only one shortest path between any pair of vertices* [18]. In this work, we study the generalized graph topology, where there may exist more than one shortest path between two vertices which is very common in graphs such as a social network. Our goal is to recover the non-local distance between any pair of vertices using only one shortest path. Finally, in this paper, we focus on developing methods to discover minimum gate-vertex set, whereas [18] only targets at a random set of vertices which forms a k-skip cover, i.e., the minimum k-skip cover problem is not addressed.

II. Properties of Gate-Vertex Set and Problem Transformation

Based on the definition of the gate-vertex set, to verify that a given set of vertices V^* is a gate-vertex set, the naïve approach is to explicitly verify that the distance between every **non-local pair** (u, v) can be recovered through some sequence of consecutive **local pairs**: (u, v_1), (v_1, v_2), $\cdots$, (v_k, v), where all intermediate vertices $v_1, v_2, \cdots, v_k \in V^*$. Clearly, this can be expensive and difficult to directly apply to discover the minimum number of gate vertices. In Subsection II-A, we first discuss an alternative (and much simplified) condition, which enables the discovery of gate-vertex set using only local distance, and reveal the NP-hardness of minimum gate-vertex set discovery problem. In addition, we utilize the VC-dimension theory to bound the size of gate vertices.

A. Local Condition and Problem Reformulation

In order to design a more efficient and feasible algorithm, we explore the properties of gate vertices and observe that gate-vertex set can be efficiently checked by a very simple condition. Let $G = (V, E)$ be an unweighted and undirected graph. For any vertex $u \in V$, its **ϵ-neighbors**, denoted as $N_\epsilon(u)$ is a set of vertices such that their distances to u is no greater than ϵ, i.e., $N_\epsilon(u) = \{v \in V | 0 < d(u, v) \leq \epsilon\}$. Let L be a set of vertices and $S = \{(u_0, v_0), ..., (u_k, v_k)\}$ be a set of vertex pairs in the graph G. We say that L **covers** S if for each vertex pair $(u_i, v_i) \in S$ there is at least one vertex $x \in L$ such that $d(u_i, v_i) = d(u_i, x) + d(x, v_i)$.

Now, we introduce the following key observation:

Lemma 1: **(Sufficient Local Condition for MGS)** If for each vertex x in the graph G, there is a subset of vertices $L(x) \subseteq N_{\epsilon-1}(x)$ which covers all vertex pairs $\{(x, y_i) | d(x, y_i) = \epsilon\}$, then $\bigcup_{x \in V} L(x)$ is a gate-vertex set of graph G. In other words, a vertex set which covers any pair of vertices with distance ϵ is a gate-vertex set.

The proof of this lemma is in the full technical report [14]. Interestingly, this local condition specified in Lemma 1 is also a necessary one for a gate-vertex set.

Lemma 2: **(Necessary Local Condition for MGS)** Given an unweighted and undirected graph G and its gate-vertex set V^* with respect to parameter ϵ, for any vertex $s \in V$, we have $L(s) = \{x \in V^* | 0 < d(s, x) < \epsilon\}$ such that for any vertex t with distance ϵ to s (i.e., $d(s, t) = \epsilon$), there is $x \in L(s)$ with $d(s, t) = d(s, x) + d(x, t)$.

This lemma directly follows the definition of gate-vertex set and is omitted for simplicity.

Putting this together, given parameter ϵ, checking whether a subset of vertices $V^* \subseteq V$ is gate-vertex set is equivalent to checking the following condition: for any vertex pair (u, v) with distance ϵ, there is a vertex $x \in V^*$ such that $d(u, v) = d(u, x) + d(x, v)$. Similarly, we can rewrite the minimum gate-vertex set discovery problem in the following equivalent local condition (only covering vertex pairs with distance ϵ).

Definition 3: (**Minimum Gate-Vertex Set Problem using Local Condition**) Given unweighted undirected

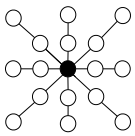


Figure 1: Gate-vertex set ($\epsilon = 3$)

graph $G = (V, E)$ and user-defined threshold ϵ, we would like to seek a set of vertices V^* with minimum cardinality, such that any pair of vertices (u, v) with distance ϵ is covered by at least one vertex $x \in V^*$: $d(u, x) + d(x, v) = d(u, v)$.

Theorem 1: **(NP-hardness of MGS using Local Condition provided Shortest Paths)** Given a collection P of vertex-pair (u, v) with $d(u, v) = \epsilon$ denoting a set of shortest paths from unweighed undirected graph $G = (V, E)$, finding minimum number of vertices $V^* \subseteq V$ such that any vertex-pair (u, v) is covered by at least one vertex $x \in V^*$ is NP-hard.

We prove Theorem 1 by reducing the 3SAT problem to this problem (see detailed proof in [14]).

B. Size of Minimum Gate-Vertex Set

In the following, using the theory of VC-dimension, we derive an upper bound of the cardinality of minimum gate-vertex set.

VC-dimension and ϵ-net: We start with a brief introduction of the VC-dimension of set systems and ϵ-net. The notion of VC-dimension originally introduced by Vladimir Vapnik and Alexey Chervonenkis in [20] is widely used to measure the expressive power of a set system. Let U be a finite set and R a collection of subsets of U, the pair (U, R) is referred to be a set system. A set $A \subseteq U$ is *shatterable* in R if and only if for any subset S of A, there is always a subset $X \in R$ where $X \cap A = S$. In other words, X contains the "exact" S with no element in $A \backslash S$. Then, we say the VC-dimension of set system (U, R) is the largest integer d such that no subset of U with size $d + 1$ can be shattered. In addition, given parameter $\epsilon \in [0, 1]$, a set $N \subseteq U$ is an ϵ-net on (U, R) if for any subset $X \in R$, X has size no less than $\epsilon|U|$, the set N contains at least one element of X. For the set system with bounded VC-dimension d, the ϵ-net theorem states that there exists an ϵ-net with size $O(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$ [7].

Using the VC-dimension and ϵ-net theorem, we can bound the size of minimum gate-vertex set.

Theorem 2: Given graph $G = (V, E)$ with parameter ϵ, the size of minimum gate-vertex set is bounded by $O(\frac{|V|}{\epsilon-1} \log \frac{|V|}{\epsilon-1})$.

The proof of this theorem can be found in the full technical report [14].

Lower Bound: The lower bound of the minimum gate-vertex set can be arbitrarily small. For example, in Figure 1, minimum gate-vertex set is only central vertex, and no gate vertex is needed for any graph with diameter less than ϵ. In this case, even a gate-vertex set of size $O(\frac{|V|}{\epsilon-1} \log \frac{|V|}{\epsilon-1})$ is obtained, we still cannot decide *how good it is compared to the minimum gate-vertex set.*

III. Algorithms for Gate-Vertex Set Discovery

Based on Theorem 2 and ϵ-net theorem [7], we observe that any random sample with size $O(\frac{|V|}{\epsilon-1} \log \frac{|V|}{\epsilon-1})$ has high probability to form a gate-vertex set but does not have a guarantee. An *adaptive sampling* method [18] is introduced to guarantee to find a k-skip cover. The guarantee is achieved by choosing a vertex using the information gained from previously sampled vertices. Since a k-skip cover can serve as a candidate for the gate-vertex set with $\epsilon = k + 1$ (as stated in Lemma 3), we can utilize the adaptive sampling method to discover gate-vertex set. However, since the lower bound of the minimum gate-vertex set can be arbitrarily small, the approximation ratio between the size of the gate-vertex set discovered by this method and the minimum one is not bounded. In other words, this method does not necessarily produce tight gate-vertex set.

Lemma 3: Given graph $G = (V, E)$, if parameters ϵ of gate-vertex set and k of k-skip cover satisfy condition $k = \epsilon - 1$, k-skip cover V^* is a gate-vertex set.

We can prove this lemma by contradiction. For simplicity, we omit it here. Note that a gate-vertex set with locality parameter $\epsilon = k+1$ may not be a k-skip cover. Also, as we mentioned earlier, the k-skip cover focuses on the unique shortest path system, and since there may exist more than one shortest path between two vertices, the adaptive sampling method chooses one of such paths arbitrarily.

Set-Cover Based Approach: We propose an effective algorithm based on set cover framework to discover gate-vertex set with logarithmic bound. Specially, we transform the minimum gate-vertex set discovery problem (MGS) to an instance of set cover [3] problem: Let $U = \{(u, v) | d(u, v) = \epsilon\}$ be the ground set, which includes all the non-local pairs with distance equal to ϵ. Each vertex x in the graph is associated with a set of vertex pairs $C_x = \{(u, v) | d(u, v) = d(u, x) + d(x, y) = \epsilon\}$, where C_x includes all of the non-local pairs with distance equal to ϵ and there is a shortest path between them going through vertex x. Given this, in order to discover the minimum gate-vertex set, we seek *a subset of vertices $V^* \in V$ to cover the ground set, i.e., $U = \bigcup_{v \in V^*} C_v$, with the minimum cost $|V^*|$.* Basically, V^* serves as the index for the selected candidate sets to cover the ground set.

Theorem 3: The minimum solution V^* for the above set-cover instance is a minimum gate-vertex set of graph G with parameter ϵ.

Its proof can be easily followed by Definition 3. The minimum set cover problem is NP-hard, and we can apply the classical greedy algorithm [3] for this problem: *Let R records the covered pairs in U (initially, $R = \emptyset$). For each possible candidate set $C_x = \{(u, v) | d(u, v) = d(u, x) + d(x, y) = \epsilon\}$ discussed above, we define the price of C_x as:*

$$\gamma(C_x) = \frac{1}{|C_x \setminus R|}$$

At each iteration, the greedy algorithm picks the candidate set C_x with the minimum $\gamma(C_x)$ (the cheapest price) and put its corresponding vertex x in V^*. Then, the algorithm will update R accordingly, $R = R \cup C_x$. The process continues until R completely covers the ground set U ($R = U$), which contains all non-local pairs with distance equal to ϵ. It has been proved that the approximation ratio of this algorithm is $ln(|U|) + 1$ [3].

Fast Transformation: In order to adopt the aforementioned set-cover based algorithm to discover the gate-vertex set, we first have to generate the ground set U and each candidate subset C_x associating with vertex x. Though we only need the non-local pairs with distance ϵ, whose number is much smaller than all non-local pairs, the straightforward approach still needs to precompute the distances of each pair of vertices with distance no greater than ϵ, and then apply such information to generate each candidate set. For large unweighted graphs, such computational and memory cost can still be rather high.

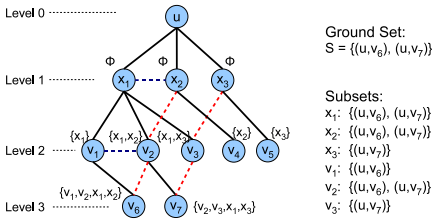
Here, we introduce an efficient procedure which performs a local BFS for each vertex to visit only its ϵ-neighborhood and during this process to collect all information needed for constructing the set-cover instance (U and C_x, for $x \in V$). Specifically, for each local BFS starting from a vertex u, it has the following two tasks: 1) it needs to find all the vertices which is exactly ϵ distance away from u, and then add them into U; and 2) for each such pair (u, v) ($d(u, v) = \epsilon$), it needs to identify all the vertices x which can appear in a shortest path from u to v. In order to achieve these two tasks, we again utilize the basic recursive property of the shortest-path distance: *Let vertex y to u's distance be d and $z \neq u$ be vertex whose distance to u is $d - 1$, we know all the intermediate vertices appearing in at least one shortest path from u to z (denoted as $I(z)$). Then, all the intermediate vertices on shortest paths from u to y can be written as* $\bigcup_{(z,y)\in E}(I(z) \cup \{z\})$. Based on this property, we can easily maintain $I(x)$ for each vertex x such that $1 < d(u, x) < \epsilon$; when $d(u, x) = 1$, $I(x) = \emptyset$. Since BFS visits u's ϵ-neighborhood in a level-wise fashion, when it reaches the ϵ level, where each vertex v is ϵ distance to u, we not only get each targeted pair (u, v), but also get $I(v)$, which we can easily use for producing the candidate set: for each $x \in I(v)$, we add (u, v) to C_x.

Algorithm 1 sketches the BFS-based algorithm for constructing the set cover instance. Especially, set $I(v)$ is computed in Line 5, and when BFS reaches the ϵ level (Line 11), it adds (u, v) to the ground set U (Line 12) and to each C_x ($x \in I(v)$) (Line 13). The algorithm will be invoked for each vertex u in the graph. Finally, Figure 2 illustrates a simple running example of Algorithm 1 for vertex u with $\epsilon = 3$.

Computational Complexity: The overall set-cover based mining algorithm for discovering gate-vertex set includes two key steps: 1) Constructing set-cover instance (Algorithm 1) and 2) the greedy set-cover discovering algorithm. The first step for collecting ground set and each candidate

Algorithm 1 BFSSetCoverConstruction($G = (V, E)$,ϵ,U,u)

```
1:  I(u) ← ∅; level(u) ← 0; Q ← {x} {queue for BFS};
2:  while Q ≠ ∅ do
3:      u ← Q.pop();
4:      if level(v) ≥ 2 {d(u, v) ≥ 2} then
5:          I(v) ← ⋃_{(x,v)∈E∧level(x)+1=level(v)} I(x) ∪ {x}
6:      end if
7:      for all v ∈ Neighbor(u) {(u, x) ∈ E} do
8:          if v is not visited then
9:              if level(v)<ε {d(u, v)<ε} then
10:                 Q.push_back(v);
11:             else
12:                 U ← U ∪ {(u, v)};
13:                 ∀x ∈ I(v), C_x ← C_x ∪ {(u, v)};
14:             end if
15:         end if
16:     end for
17: end while
```



(a) Example for Algorithm 1 (b) Example for Set Cover

Figure 2: Example for Gate Discovery Algorithm ($\epsilon = 3$)

set takes $O(\sum_{v\in V}(|N_\epsilon(v)|^2 + |E_\epsilon(v)|))$, where $|N_\epsilon(v)|$ ($|E_\epsilon(v)|$) is the number of vertices (edges) in the v's ϵ-neighborhood. For the greedy set cover procedure, by utilizing the speedup queue technique [16, 10], we only need to visit $d \ll |V|$ vertices in the queue (i.e., all vertices are ranked in ascending order in the queue), and each step takes $O(d(\log|V|+1))$ time to exact and update the queue. As greedy procedure has $O(|V^*|)$ steps, it takes $O(d|V^*|(\log|V|+1))$ in total. Putting together, the overall algorithm's time complexity is $O(d|V^*|\log|V| + \sum_{v\in V}|N_\epsilon(v)|^2)$.

Once the gate vertices are selected, then we construct the gate graph, which minimally connects those gate vertices while still preserving the ability of representing the non-local distances. The algorithm for gate graph construction is rather straightforward and the details can be found in the full technical report [14].

IV. Experimental Evaluation

In this section, we empirically study the performance of our approaches on both real and synthetic datasets. Specifically, we compare two methods in the experiments: 1) **FS**, which corresponds to the approach utilizing adaptive sampling [18] for gate vertices discovery; 2) **SC**, which corresponds to the approach using set cover framework for gate vertices discovery (Subsection **??**). Here, we are interested in understanding how many vertices can be reduced by the gate-vertex set and how many edges are needed in the gate graph, and how they are affected by the locality parameter ϵ? We implemented our algorithms in C++ and Standard Template Library (STL). All experiments

were conducted on a 2.8GHz Intel Xeon CPU and 12.0GB RAM running Linux 2.6.

Here, we collect 7 real-world datasets listed in Table 3 to validate the performance of our approaches. (An extensive study on the synthetic datasets can be found in the full technical report [14]). Among them, CA-GrQc and CA-HepTh are collaboration networks from arXiv describing scientific collaboration relationships between authors in General Relativity and Quantum Cosmology field, and in High Energy Physics field, respectively. Moreover, P2PG08, P2PG09, P2PG30 and P2PG31 are 4 snapshots of the Gnutella peer-to-peer file sharing network collected in August and September 2002, respectively. Wiki-Vote describes the relationships between users and their related discussion from the inception of Wikipedia until January 2008. All datasets are publicly available at Stanford Large Network Dataset Collection [1].

Table I reports the size of gate-vertex set and the number of edges in gate graphs by varying locality parameter ϵ from 3 to 6. Their corresponding shortest distance distribution and vertex degree distribution are shown in Figure 4 and Figure 5, respectively. Since the distances and vertex degrees of P2PG30 and P2PG31 have similar distribution with that of P2PG08 and P2PG09, and their large values would affect other datasets' distribution visualization, we omit them in both figures. We make the following observations:

Size of Gate-Vertex Set: Table I shows that the sizes of gate-vertex set discovered by both FS and SC are consistently smaller than that of original graphs. Among them, SC always obtains the better results, which are on average approximately 76%, 65%, 63% and 56% of the one from FS with ϵ ranging from 3 to 6. For SC approach, the size of gate-vertex set by SC is on average around 26%, 21%, 27% and 24% of the corresponding original graph when ϵ varies from 3 to 6. We also observe that, as locality parameter ϵ increases, the number of gate vertices discovered by SC is gradually reduced. Particularly, reduction ratios of CA-GrQc, CA-HepTh and Wiki-Vote are consistently better than that of P2P08, P2P09, P2P30 and P2P31. In Figure 5, CA-GrQc, CA-HepTh and Wiki-Vote seem to fit the power-law degree distribution very well, while there are a significant portion of vertices with degree ranging from 10 to 15 in P2P08, P2P09, P2P30 and P2P31. In other words, there exists a small portion of vertices with high degree potentially serving as the intermediate connectors for traffics between a large portion of vertex pairs in CA-GrQc, CA-HepTh and Wiki-Vote. By SC's gate vertices discovery method using set cover framework, those vertices can be selected as gate vertices and thus dramatically simplify original graphs. However, for file-sharing network, a relatively large number of vertices with high connectivity potentially leads to larger size of gate-vertex set by the same selection principle. From the perspective of application domains, the results of SC on three social networks (i.e., CA-GrQc, CA-HepTh and Wiki-Vote) suggest a small highway structure capturing major non-local communications in the network. Interestingly, the consistent decreasing trends regarding the size of gate-vertex set with increasing ϵ are not observed in the results of FS on P2P30 and P2P31. Since adaptive sampling approach follows the spirit of greedy algorithm - choosing each gate vertex only based on local information, the mis-selection of gate vertices at earlier stages probably leads to significant increase of gate vertices at later stages. In other words, some important vertices selected as gate vertices in the procedure with small ϵ might be missed in the procedure with larger ϵ. Therefore, it is reasonable to observe that the number of gate vertices discovered by FS unexpectedly becomes larger when ϵ increases.

Edge Size of Gate Graph: The number of edges in original graphs are significantly reduced by SC on three datasets CA-GrQc, CA-HepTh and Wiki-Vote. Especially, on average, the number of edges in gate graphs generated by SC are 6.5, 6, 6.3 and 6 times smaller than that of original graphs for ϵ to be 3, 4, 5 and 6. Besides that, SC still outperforms FS on those datasets, such that the number of edges in gate graphs by SC are on average about 49%, 51%, 53% and 48% of the one from gate graph by FS ranging ϵ from 3 to 6. Interestingly, as ϵ becomes larger, the number of edges in gate graphs generated by SC increases on CA-GrQc and CA-HepTh. The reason is, in order to guarantee that shortest paths between all non-local vertex pairs can be recovered utilizing fewer gate vertices, more edges are needed to build stronger connections among gate vertices. However, the number of edges in gate graphs generated by FS on CA-GrQc and CA-HepTh becomes smaller when ϵ increases. This demonstrates the effectiveness of edge sparsification algorithm for pruning redundant edges, since some of gate vertices discovered by FS are non-essential and are not necessarily to be connected to its ϵ neighbors. For other four datasets (P2P08, P2P09, P2P30 and P2P31), gate graphs generated by FS from those datasets contains fewer edges compared to the one of SC. Overall, they are on average about 1.3, 1.1, 1.4 and 1.8 times smaller than that of SC varying ϵ from 3 to 6. Also, as ϵ increases, the number of edges in gate graphs generated by both approaches on those four datasets increases. This is consistent with our earlier discussion that in these graphs, their interactions seem to be more random and the relatively large number of vertices with degrees between 10 to 15 may increase their chance to connect to other vertices with local walks.

Running Time: We take $\epsilon = 3$ as an example. The running time of FS for all 7 datasets are 65ms, 132ms, 3s, 127ms, 158ms, 447ms and 811ms. The running time of SC are 23s, 53s, 1166s, 183s, 293s, 279s and 661s for CA-GrQc, CA-HepTh, Wiki-Vote, P2P08, P2P09, P2P30 and P2P31, respectively. As locality parameter ϵ increases, the computational cost of both approaches become larger, because more vertex pairs should be considered in SC and more vertices would be traversed in FS. The average

[1] http://snap.stanford.edu/data

Dataset	#V	#E	Dia.	Avg.Dist
CA-GrQc	5242	28980	17	6.1
CA-HepTh	9877	51971	18	6
Wiki-Vote	7115	103689	7	3.3
P2PG08	6301	20777	9	4.6
P2PG09	8114	26013	9	4.8
P2PG30	36682	88328	11	5.7
P2PG31	62586	147892	11	5.9

Figure 3: Real Datasets

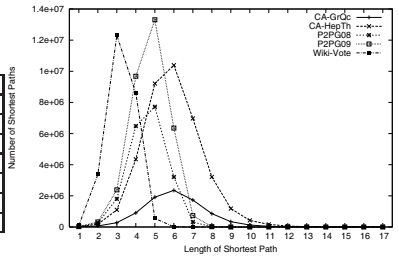


Figure 4: Distance Distrib.(Real Graph)

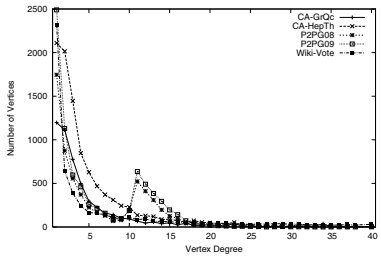


Figure 5: Degree Distrib. (Real Graph)

Dataset	$\epsilon = 3$				$\epsilon = 4$				$\epsilon = 5$				$\epsilon = 6$			
	$\#V$		$\#E$		$\#V$		$\#E$		$\#V$		$\#E$		$\#V$		$\#E$	
	FS	SC	FS	SC	FS	SC	FS	SC	FS	SC	FS	SC	FS	SC	FS	SC
CA-GrQc	2836	869	9266	2655	1625	655	6848	2933	1116	567	5580	2984	908	500	5192	2858
CA-HepTh	5131	2208	15831	7674	3381	1669	14921	10241	2525	1364	14316	11249	2134	1157	14476	11456
Wiki-Vote	2564	1598	84607	59132	2457	879	85051	34736	2236	584	83681	22652	2964	571	193913	19571
P2P08	2359	2340	10892	10738	2313	1920	23787	25406	2082	1584	26497	40218	2043	1095	28002	49139
P2P09	2930	2904	13633	13394	2874	2474	30219	32976	2643	2047	34870	53851	2556	1530	37022	75113
P2P30	9688	9627	37708	36708	10874	8820	98194	92820	8713	7551	108720	151677	8914	6845	127565	232971
P2P31	16493	16394	64624	146765	18248	14996	161883	155099	14745	12847	182599	256578	14895	11738	215742	383725

Table I: Sizes of Simplified Graph on Real Datasets

running time of SC on $\epsilon = 5$ can cost up to a few hours, which is around 100 times slower than that of FS. Indeed, the selection between FS and SC is a trade-off between reduction ratio and efficiency. In general, we can see that with rather smaller ϵ (2 or 3), the vertex reduction by SC is quite significant which is also much better than that of FS, and their running time are reasonable in practice. In contrast to FS, the size of gate-vertex set discovered by SC is guaranteed to hold logarithmic approximation bound.

V. Conclusion

In this paper, we study a new graph simplification problem to provide a high-level topological view of the original graph while preserving distances. Specifically, we develop an efficient algorithm utilizing recursive nature of shortest paths and set cover framework to discover gate-vertex set. More interestingly, our theoretical results and algorithmic solution can be naturally applied for minimum k-skip cover problem, which is still open problem. In the future, we would like to study whether approximate distance with guaranteed accuracy can be gained based on our framework. We also want to investigate how our simplified graph can be applied for graph clustering, multidimensional scaling and graph visualization.

Acknowledgment

We would like to thank the anonymous reviewers for their insightful comments. R. Jin and N. Ruan were partially supported by the National Science Foundation, under CAREER grant IIS-0953950. Y. Huang was partially supported by the National Science Foundation, under grant IIS-1017926.

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] Therese C. Biedl, Brona Brejová, and Tomás Vinar. Simplifying flow networks. In *MFCS '00*, pages 192–201, 2000.

[3] V. Chvtal. A greedy heuristic for the set-covering problem. *Math. Oper. Res*, 4:233–235, 1979.

[4] Vin de Silva and Joshua B. Tenenbaum. Global Versus Local Methods in Nonlinear Dimensionality Reduction. In *NIPS'03*, 2003.

[5] Tomás Feder and Rajeev Motwani. Clique partitions, graph compression and speeding-up algorithms. In *STOC '91*, 1991.

[6] Ronald J. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *ALENEX/ANALC'04*, pages 100–111, 2004.

[7] D Haussler and E Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the second annual symposium on Computational geometry*, SCG '86, pages 61–71, 1986.

[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] Herman, G. Melançon, and M. S. Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[10] Ruoming Jin, Yang Xiang, Ning Ruan, and David Fuhry. 3-hop: a high-compression indexing scheme for reachability query. In *SIGMOD Conference*, pages 813–826, 2009.

[11] Chinmay Karande, Kumar Chellapilla, and Reid Andersen. Speeding up algorithms on compressed web graphs. In *WSDM '09*, 2009.

[12] Ewa Misiolek and Danny Z. Chen. Two flow network simplification algorithms. *Inf. Process. Lett.*, 97:197–202, 2006.

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

[14] Ning Ruan, Ruoming Jin, and Yan Huang. Distance preserving graph simplification. Technical report, http://arxiv.org/abs/1110.0517, 2011.

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

[16] Ralf Schenkel, Anja Theobald, and Gerhard Weikum. Hopi: An efficient connection index for complex xml document collections. In *EDBT '04*, pages 237–255, 2004.

[17] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *STOC'08*, pages 563–568, 2008.

[18] Yufei Tao, Cheng Sheng, and Jian Pei. On k-skip shortest paths. In *Proc. ACM SIGMOD Int'l Conf. Mgmt. data*, pages 43–54, 2011.

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

[20] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16, 1971.

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