**Fast Algorithm for**

**Modularity-based Graph Clustering**

Hiroaki Shiokawa NTT Software Innovation Center,

NTT Corporation, July 23rd, 2013

**Group #7**

Vamsi Angajala

Raja Mummidi

Kshitija Joshi

Bharadwaj Aryasomayajula

**Index**

1. Abstract
2. Background and Motivation
3. Graph Clustering
4. Proposed Algorithm
5. Objective and Contributions
6. Clustering coefficient & Power-law degree distribution
7. Incremental Aggregation
8. Incremental Pruning
9. Efficient ordering of node selection
10. Graph clustering algorithm

11.Comparison with state of the art algorithm-BGLL

i. Modularity Scores

ii. Computational time

iii. Computational time- power-law differences

iv. Computational time-size differences

12. Conclusion

## 1.Abstract

Modularity is a measure of the structure of networks or graphs. The network is divided into groups, clusters or communities. Modularity is used to measure the strength of division of a network.

Modularity-based graph clustering algorithms are applied to various applications related to Artificial Intelligence and Web Communities. However, existing algorithms (K-means, Hierarchal clustering algorithms) are not applied to large graphs because they have to scan all vertices/edges iteratively.

This graph clustering algorithm compute clusters with high modularity from extremely large graphs with more than a few billion edges. The main part of the solution is to compute clusters by incrementally pruning unnecessary vertices/edges and optimizing the order of vertex selections.

The algorithm is based on three ideas. First, it incrementally aggregates vertices, which are placed in same cluster, into a single vertex to eliminate unnecessary vertices/edges from the graph. Second, it incrementally prunes vertices whose clusters can be obtained without modularity computing. Last, it optimized the order of vertex selections to reduce the number of modularity computations in the clustering process and dynamically selects the vertex with the smallest degree.

The advantages of the approach are that it can extract clusters with quite-small computational cost for complex networks (Newman 2003). With the high clustering coefficients and power-law degree distributions, this algorithm runs faster on large size complex networks. Because of pruning methods and prevention of imbalanced clustering results, this algorithm can produce results with high modularity.

**2.Background and Motivation**

In the last decade it became apparent that a large number of the most interesting structures and phenomena of the world can be described by networks: separable elements, with connections (or interactions) between certain pairs of them.

These huge networks pose exciting challenges for the mathematician. Graph Theory (the mathematical theory of networks) faces novel, unconventional problems: these very large networks (like the Internet) are never completely known, in most cases they are not even well defined. Data about them can be collected only by indirect means like random local sampling.

Recently, modularity-base clustering proposed by Newman and Girvan (Newman and Girvan 2004) has become one of the most popular algorithms for extracting clusters in a graph. Modularity evaluates the density of edges inside clusters as compared to edges between clusters. The better clustering results are achieved with higher modularity scores. Modularity-based algorithms have been applied to many applications, described below, in AI and Web communities due to its effectiveness:

**Example:**

Large-scale graphs become available

• Facebook: 1.11 billion active users /month

• Twitter: 140 million active users / day 340 million new posts / day

There are lot of techniques for analyzing massive-scale graph. Massive data require so much time for analysis. It is important to analyze large scale data quickly.

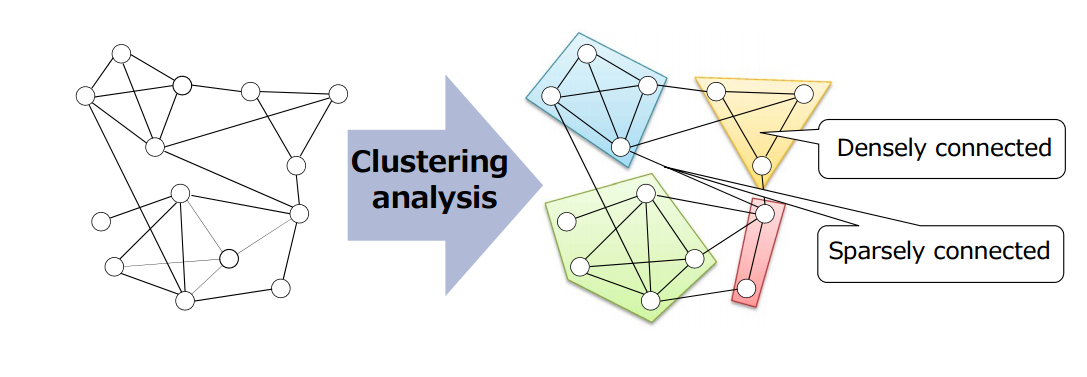
**3.Graph Clustering**

Graph Clustering is finding natural grouping or clusters or communities with in the sub graphs of sets G = (V, E). |V| is the number of vertices and |E| is the number of edges. Within-graph, clustering methods divides the nodes of a graph into clusters which are densely connected separated from the sparsely connected nodes.

E.g. In a social networking graph, these clusters could represent people with same/similar hobbies.

Algorithms for Graph Clustering

* k-Spanning Tree
* Shared Nearest Neighbor
* Betweenness Centrality Based
* Highly Connected Components
* Maximal Clique Enumeration
* Kernel k-means



**4. Proposed Algorithm**

Modularity-based Graph Clustering is done based on the measure Modularity which is used to measure the strength of division of a network into modules.

Let euv be the total number of edges between cluster u and v; au be the total number of edges that are attached to vertices in cluster u; and m be the total number of edges in the whole graph. The following equation gives the modularity score of the clustering result.

Modularity Q is measured as

**Q =**

C: set of cluster

euv: Number of edges between cluster u, v

m: total number of edges in a graph

: The fraction of outgoing edges from the cluster i

au/2m is the expected fraction of edges of u, which can be obtained when we assume the graph to be a random graph. Therefore, well clustered graphs will have high modularity scores, since the value of euu is highly different from the random graph.

This clustering method finds the division of graph to maximize the modularity measure. Finding the division which maximizes modularity is NP-complete. A lot of greedy approaches were proposed. Modularity based clustering improves the clustering speed about 100M nodes/hour

Where as BGLL processes 10M nodes/hour, CNM process 1M nodes/hour, Newman method process 100k nodes/hour and Girvan-Newman method processes 10k nodes/hour

**5.Objective and Contributions**

**Objective:**

The objective of the algorithm is to improve the fastness of graph clustering method with high modularity using 3 key techniques

1. Incremental nodes aggregation
2. Incremental nodes pruning
3. Efficient ordering of nodes selections

**Contributions of our algorithm**

Efficiency:

The efficiency of the algorithm MBGC (Modularity based graph clustering) is considerably faster than BGLL. The fastness is measured as forming clusters with the rate of 100M modes/3 minutes

High Modularity:

The scores of modularity is almost same or better than BGLL.

Effectiveness: It improves the performances for complex networks having huge number of nodes.

These three techniques use following measures to measure the strength of the graph clusters.

1. Clustering coefficient
2. Power-law degree distribution

**6.Clustering coefficient & Power-law degree distribution**

The clustering coefficient of adjoined vertices measures how close the pair of vertices is to being a clique.

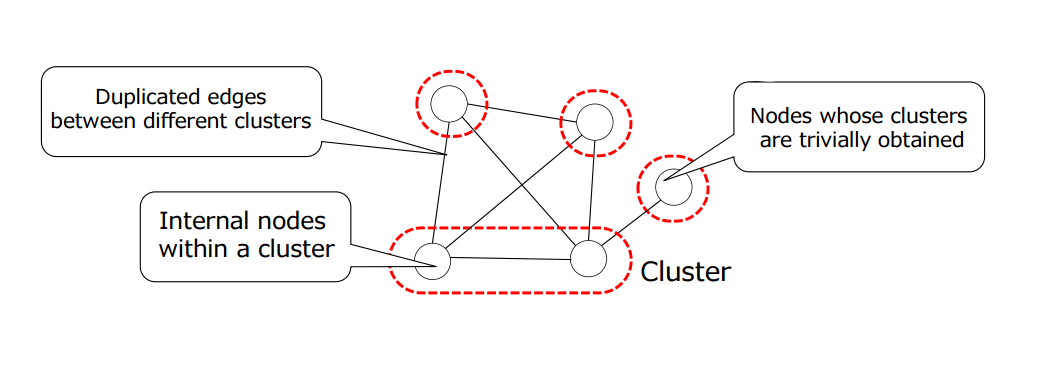
CC(v) = 2Nv/ (Kv\*(Kv-1))

v : a node in the graph

Kv: degree of the node v

Nv: number of links between neighbors of v

There are many duplicated nodes/edges in a graph which has large clustering coefficient



**Power-law probability distribution**

a power-law probability distribution is a distribution whose density function (or mass function in the discrete case) has the form

(x) \propto L(x) x^{-\alpha}

where alpha > 1, and (x) is a slowly varying function which is any function that satisfies lim_{x\rightarrow\infty} L(r\,x) / L(x) = 1 for any positive factor https://upload.wikimedia.org/math/4/b/4/4b43b0aee35624cd95b910189b3dc231.png.

Complex networks have highly skewed degree distribution following the power-law distribution. Most of nodes only have a few neighbor nodes, and only few nodes have large number of neighbor nodes

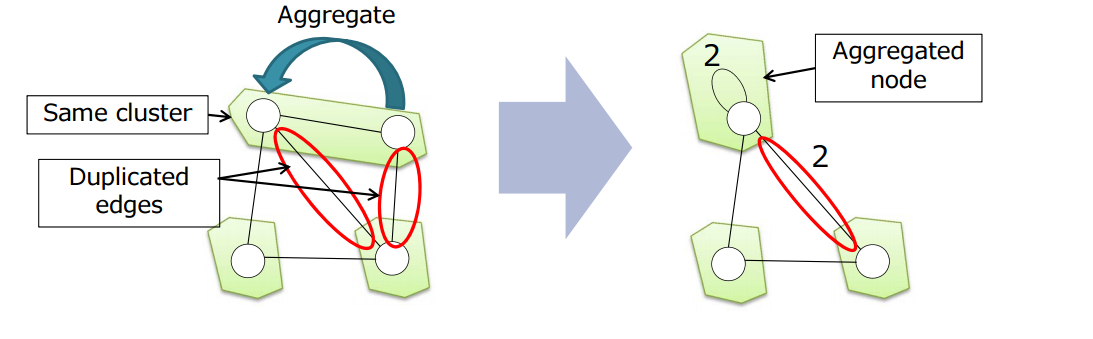
The frequency F of nodes with degree d is **F α d^(-α)**

**7.Incremental Aggregation**

We extract clusters by incrementally aggregating vertices placed in a same cluster into an equivalent single vertex with weighted edges. In contrast to previous algorithms, our proposal does not traverse all vertices/edges multiple times. In this section, we formally introduce our incremental aggregation technique and its properties

By using modularity gain, this algorithm finds clusters in a local maximization manner. When vertex u is selected, it computes the modularity gain of u for each v in Γ(u), where Γ(u) is the set of vertices neighboring u. After computing all modularity gains between u and v, our algorithm incrementally aggregates u and v that yields the highest rise in modularity.

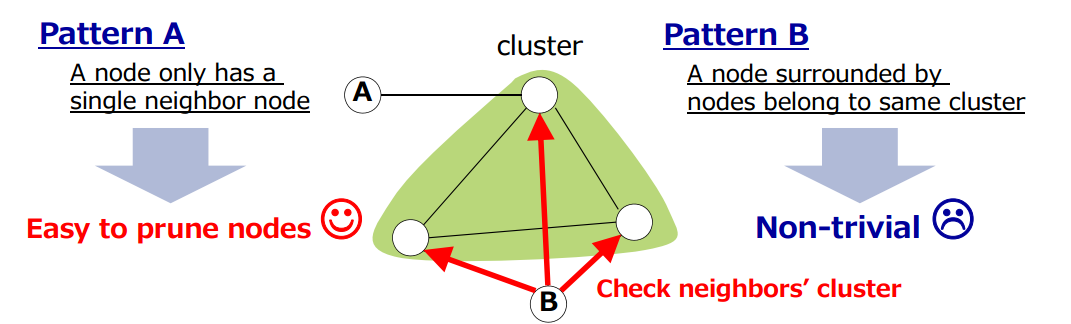
If a pair of vertices have n neighboring vertices, the pair is expected to have cn neighboring vertices that are co-referenced from both of the pair. cn edges, indicates co-referenced vertices from the pair, will be eliminated by aggregating the pair into a single vertex. Therefore, we can eliminate cn edges in each aggregation. it is obvious that our algorithm performs well when the given graph has a high clustering coefficient.



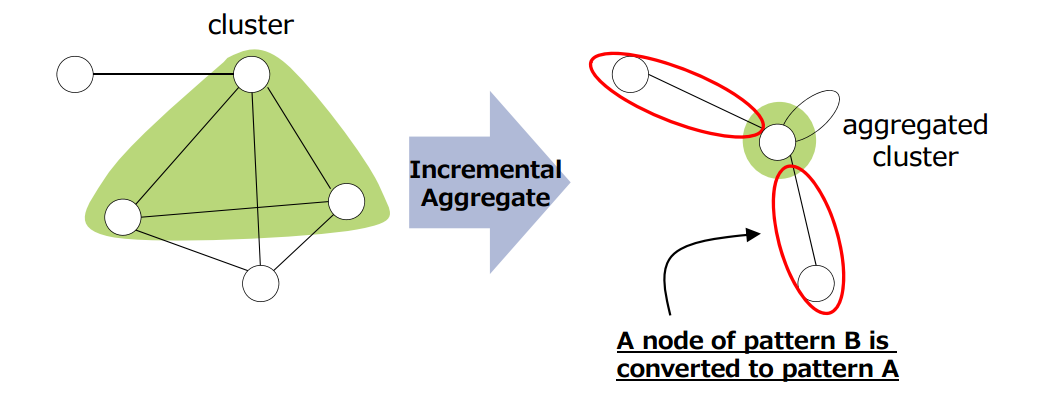
**8.Incremental pruning**

In practice, there are a lot of vertices whose clusters are trivially determined, we call them prunable vertices. We call the set of vertices whose modularity gains are to be computed as target vertices, in the clustering process. Unlike existing algorithms, our algorithm computes the modularity gain for only target vertices by dynamically removing prunable vertices in incremental manner. We formally introduce below the definitions of prunable vertices and target vertices with their theoretical properties.

Incrementally prune nodes whose cluster is trivially obtained. We can easily assign nodes to clusters without computing modularity gains. From the graph structure, there are 2 patterns of pruning



Efficient pruning approach for pattern 2.All nodes within the same cluster have been aggregated to a node by incremental aggregation.We can find all prunable nodes by obtaining nodes such that they have only a single adjacent node



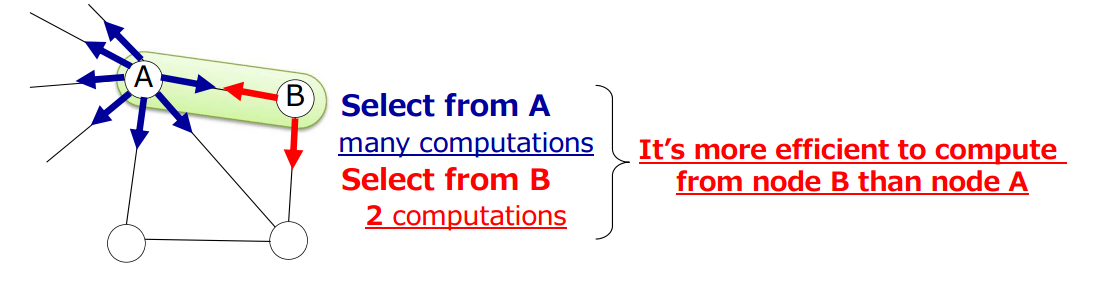
**9. Efficient ordering of node selection**

Establishes efficient ordering of vertex selections for local modularity maximization to reduce the computations. Our proposal dynamically selects vertex with the smallest degree by handling the power-law degree distribution. One of the famous properties of complex graphs is the power-law degree distribution. Most vertices have relatively few neighbors while a few vertices have many neighbors. Under the powerlaw degree distribution, the frequency of vertices with degree number of d is proportional to d −α, where exponent α is a positive constant that represents the skewness of the degree distribution. A high α implies that the vast majority of vertices have small degree. As α decreases, the graph density and the number of large degree vertices increases.

we find vertices of the highest modularity gain with low computation cost by dynamically selecting vertices with the smallest degree. By combining the ordering and the incremental aggregation, we reduce the size of degrees. Thus, the ordering reduces the computational cost especially for high degree vertex. Additionally, we find the vertex of the highest modularity gain more efficiently as the graph strongly follows power-law degree distribution. This is because vertices in the power-law degree distribution tend to have highly skewed degree.

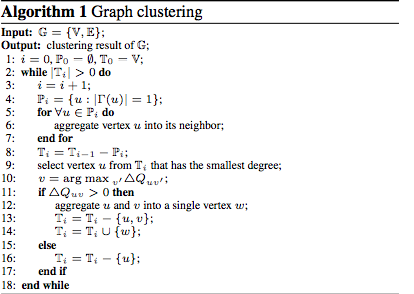
Dynamically selects a node with the smallest degree.

Example: Node A and B being assigned to the same cluster. By selecting node with the smallest degree, we can avoid producing super-cluster structures



**10. Graph clustering algorithm**

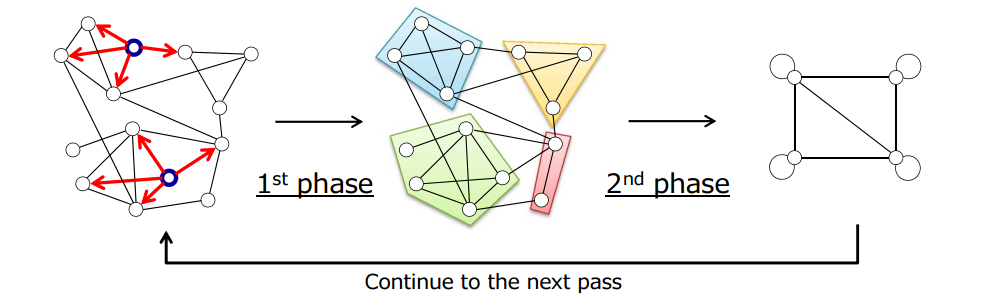
In this algorithm. First, if i = 0, the algorithm initializes P0 = ∅ and T0 = V (line 1). Next, it incrementally computes prunable vertices Pi (line 4) and merge each vertex in Pi into its neighboring cluster (lines 5-7). Next, it obtains target vertices Ti. It selects vertex u with the smallest degree from Ti based on Observation 1 (line 9), and finds neighbor vertex v that maximizes the modularity gain (line 10). If the modularity gain 4Quv is positive, it then aggregates vertices u and v into a single vertex (lines 11-14). Otherwise, vertex u is pruned from Ti (line 16). If Ti contains no vertices, it terminates its iteration cycle. Finally, it returns aggregated vertices as a result; all vertices included in an aggregated vertex belong to same cluster.



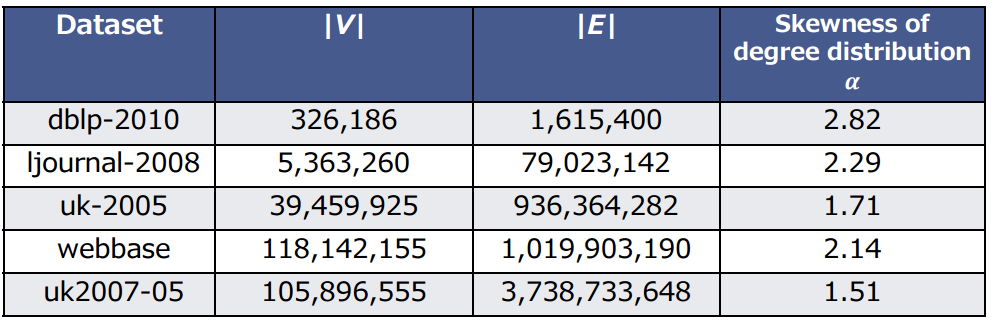
**11. Comparison with state of the art algorithm-BGLL**

Continuing following passes until the modularity score is maximized

1st phase: Local clustering 1) Selects a node 2) Computes the modularity gain 3) Places the neighbor node in the same cluster 2nd phase: Data reduction.Aggregates all nodes in the same cluster as a single node



Datasets & Experimental Environment • Real world datasets • 2 Social networks and 3 Web graphs of IP domains

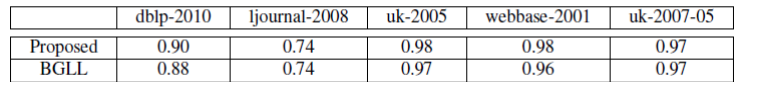


Experimental Environment • All experiments were conducted on a Linux 2.6.18 server with Intel Xeon CPU L5640 2.27GHz and 144GB RAM • Run all methods on 1 core, 1CPU

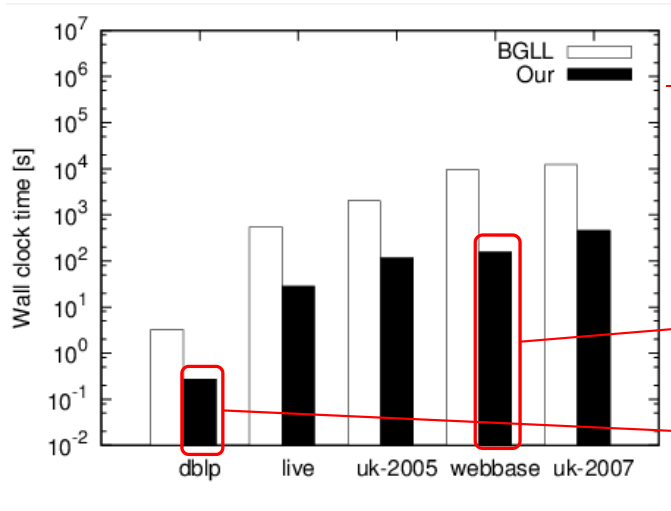
**i. Modularity score**

Modularity score for datasets. Large modularity score means the output of algorithms is well clustered. Proposed method achieves almost same modularity scores as/slightly higher than BGLL.

Modularity Q of Proposed algorithm vs BGLL

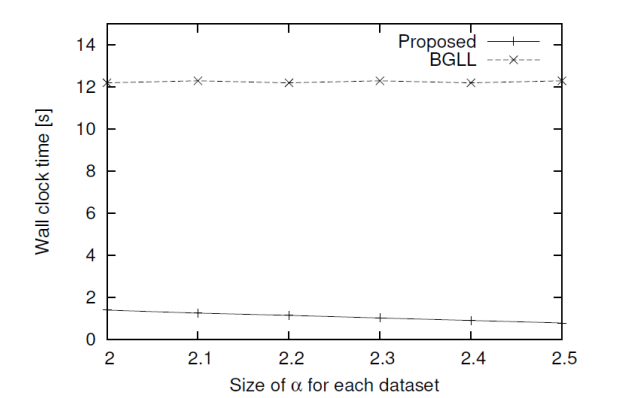


ii. **Computational time**

Proposed algorithm is up to 60 times faster than the state of the art algorithm BGLL. One major advantage of our algorithm is that it outputs clusters with high modularity. Table shows modularity Q for each of the real world datasets. In contrast, our algorithm successfully avoid to produce super-clusters by using a local modularity maximization and efficient ordering of vertex selections. Furthermore, Table shows that our proposal achieves slightly higher modularity than BGLL even though BGLL also performs higher modularity than CNM. The computation time of BGLL is significantly larger than ours as shown in Fig. That is, these results show the superiority of our approach over the previous approaches.

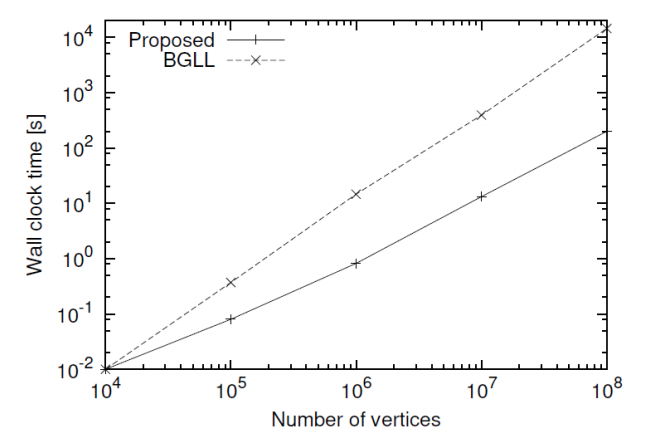
**iii Computational time – power-law differences**

Fig. shows the computation times of our proposal and BGLL for different α values (from 2.0 to 2.5) of graphs with 1 million vertices; α represents the skewness of the powerlaw degree distribution. It is known that the clustering coefficient also follows a power-law degree distribution. Graphs with large α tend to have high clustering coefficients. As shown in Fig., BGLL shows almost constant computational time under all conditions examined. In contrast to BGLL, our algorithm increases its clustering speed as α increases. In the most efficient case, i.e. α = 2.5, our proposal is up to two times faster than the result of α = 2.0. This is because our algorithm eliminates a significant number of vertices/edges as shown in Lemma 3 and 5, when the graph has large α. Thus, our algorithm outperforms BGLL at high α values.



**iv. Computational time – size differences**

Fig. shows the scalability for our proposal and BGLL; we show the wall clock time as a function of the number of vertices. We varied the number of vertices from 10 thousand to 100 million with α = 2.5. As shown in Fig. , our algorithm scales better than BGLL. This is because we do not traverse all vertices/edges multiple times. Thus, our proposal clearly achieves higher scalability than BGLL.



**12.Conclusion**

Fast clustering algorithm is based on three ideas. First, it incrementally aggregates vertices, which are places in a same cluster, into a single vertex. Second, it incrementally prunes computations for vertices whose clusters can be obtained. Last, it dynamically selects the vertex with the smallest degree. Experiments show that this algorithm can achieve efficient clustering with high modularity. Modularity-based algorithms are fundamental to many current and prospective applications in various disciplines. This algorithm will improve the effectiveness of future applications in AI and Web communities.

**Reference:**

Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Modularity-Based Graph Clustering Algorithm by

Hiroaki Shiokawa, Yasuhiro Fujiwara and Makoto Onizuka

<https://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/viewFile/6188/6884>