Regulated Distance Algorithm in Large Networks for Graph Partitioning

Anusha Swaminathan

Department of Computer Science Engineering
Meenakshi Sundararajan Engineering College
Chennai, India

Varsha Alangar

Department of Computer Science Engineering
Meenakshi Sundararajan Engineering College
Chennai, India

Abstract

One of the fundamental needs in many networking applications is the computation of shortest paths. There are many methodologies and algorithms to compute the shortest path but the time and complexity plays a major role in choosing the algorithm that has less execution time and executes with low complexity. We thus propose a method that partitions the graph more efficiently by clustering method. This method also reduces the search space by clipping the unnecessary sub graph branches due to the creation of the abstraction graphs. The algorithm can be further extended to routing algorithms to compute the shortest paths of various routes. This algorithm also proves to reduce the traversing distance ratio as the clustering increases.

1. Introduction

Any field including geographic information systems (GIS), computer networks and social networks today requires the computation of shortest path for some purpose. It is one of the most fundamental necessity in networking area. There are various shortest path algorithms designed to deliver optimal solutions. In 1959, Dijkstra developed a successful shortest path algorithm that produced a complexity of O(|*E*|+|*V*|log|*V*|), where |*V*| is the number of vertices and |*E*| is the number of arcs. Though Dijkstra algorithm computes the optimal solution in a theoretical sense, it is often too slow in practical applications, giving way for several other techniques for improving its response time.

A general path query is a regular expression over the labels of a graph G. For crucial applications, the path queries need to be solved with both time and accuracy kept in mind. Pre-computing, though fast, consumes lot of time and storage. A lot of research has been tried to balance between the preprocessing and the query times most of which uses graph partition techniques for the original problem decomposition. The query execution can be effectively enhanced through appropriate division and partitioning of graphs.

In this work, we concentrate on shortest path queries in large networks. With minimal amount of storage of data, and a fast preprocessing technique, our algorithm aims to accelerate the path queries based on the most suitable graph partitioning without a layout or an embedding.

2. The Hierarchical Graph Model

Assume a road network or individuals in a friendship network. Let G=(V,E) be a graph, where each vertex in V represents the network objects and the edges E be represented as $E=\{(u,v)|(u,v \in V) \wedge (u \neq v)\}$ are the connections between the preceding objects. The length of a path P is the sum of the weights or cost of all the nodes on the path, denoted by cost(P) and the distance dist(a,b) between two nodes a and b is defined by the length of the shortest path from a to b in the graph G.

Given a graph G=(V,E), the subgraphs of G is represented as SG={$G_1(V_1, E_1)$, …,$G_k(V_k, E_k)$} with $V_i \subseteq V$, $E_i \subseteq E$, $1 \leq i \leq k$. For a node $v \in V$, let Sub(v) denote the subgraph to which v belongs to.

2.1. Inter community edge

An edge (u,v)∈E is called an intercommunity edge if u,v belongs to adjacent subgraphs G_i and G_j respectively. The intercommunity edge set between G_i and G_j is denoted by $IC(G_i,G_j)=\{(u, v)\in E|(u\in B(G_i)) \wedge (v\in B(G_j)) \wedge (i \neq j)\}$.

2.2. Border node and Inner node

A node $u \in V_i$ is called a border node of G_i if there exists an edge (u,v)∈E with $v\in V_j$ and $i\neq j$, and an inner node of G_i otherwise; G_j is then called a neighbour subgraph of u, denoted by Ns(u) and the subgraphs G_i and G_j are said to be adjacent.

2.3. Abstraction graph

Abstraction graphs are those in which

- Each node $u\in V^A$ is a supernode representing the subgraph formed by {v∈V| Sub(v)=u}
- Each edge $(u, v)\in E^A$ represents the collection of edges { (u′, v′)∈E | (Sub(u′) = u) ∧ (Sub(v′) = v) }
- The weight of any node $u\in V^A$ is defined by wv^A :=minTd(u)

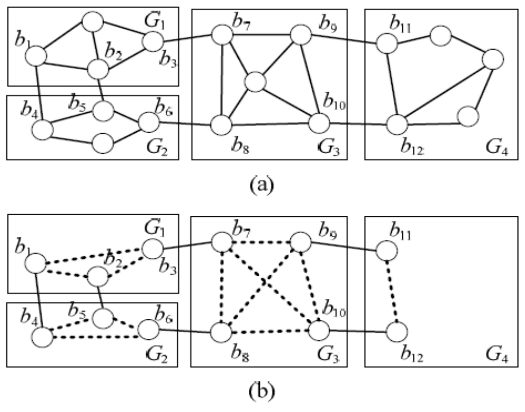

Figure 1(a). Graph G with sub graph partitions.
Figure 1(b). High-level graph constructed from the sub graphs of G

For example, Figure 1(a) shows a graph G whose sub graphs G_1, G_2, G_3, and G_4 are shown in Figure 1(b). Fig 2 shows the high-level graph constructed from G_1 to G_4, where each sub graph is represented as a complete graph composed of its border node set and the community edge set. The intercommunity edge can be thought of as forming bottlenecks between sub graphs. The corresponding abstraction graph is shown in Figure 2, which contains far fewer nodes and edges.

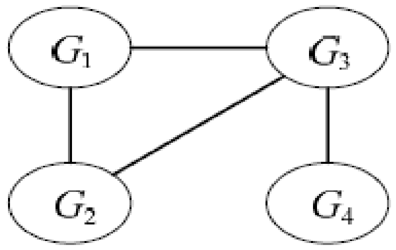

Figure 2. Abstraction Graph G for the Figure 1

Thus, this process motivates in estimating the real shortest path value efficiently by eliminating unnecessary computations. This can be done by first searching in the abstraction graph when the maximum traversing distance equals the minimum one for each subgraph. This can be used to efficiently execute the path queries.

3. Partitioning of Graphs

In this section, we discuss about the graph partitioning technique adapted and also the notations followed for the Graph Partitioning using Regulated Distance Algorithm (RDA). The traversing distance is used for the partitioning process.

3.1. Rate Determination

Suppose that the original network is partitioned into k subgraphs G1, G2, …, Gk. Let s(Gi) be the size of subgraph Gi,and Ri the ratio of the maximum traversing distance to the minimum traversing distance for subgraph Gi, Ri := max(Td(Gi)) min(Td(Gi)) , with $1 \le i \le k$. The ratio Ri:=1 if Gi has only one neighbor subgraph. For any node u ∈ B(Gi), Num(u) represents the number of intercommunity edges incident to u. The permitted upper and lower bounds for subgraph size are denoted by U δ and L δ , respectively.

In general,

min {Ri | ∀ 1 ≤ i ≤ k }

is subject to

Num(u)=1, u∈B(Gi) (A)
L δ ≤ s(Gi) ≤ U δ (B)

- Condition A is necessary to ensure that the shortest path will pass through a subgraph via atleast one community edge rather than just one border node.
- Condition B is necessary to avoid too large or small partitions and thereby regulating the path searching within each subgraph.

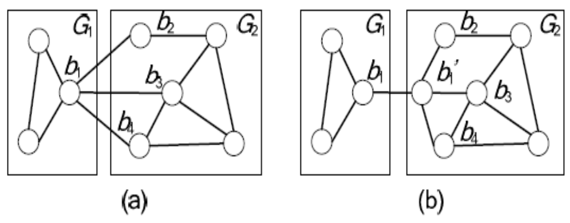

Figure 3(a). Actual Graph partitioning with boundary nodes {b_1} and {b_2,b_3,b_4}
Figure 3(b). Modified graph partitions with boundary nodes {b_1} and {b_1'}

3.2. RDA Algorithm

The Regulated Distance Algorithm (RDA) comprises of two phases namely

Phase1. Initialization

In this phase, the network is divided into a series of small subgraphs. Initally, all the nodes are unmarked. The nodes are considered one by one. The node u is marked with a new subgraph number and a set "temp" is maintained for the nodes added to this subgraph in turn. This is repeated to avoid too small subgraphs. The first node is removed from temp and judged whether it has a neighbour in a different subgraph. If not, all the unmarked neighbours are added to temp. Otherwise, it is checked if the neighbour has a neighbour node in a different subgraph and added to the end of temp. If the temp becomes null, the algorithm exits.

Algorithm1. Sub graph Initialization

```
Node u          # unmarked node
Set temp        # unmarked neighbors of u
Node v          # first node from temp
Node x          # neighbor node

initialise()
{
    while(subgraph size too small)
    {
        u.num=sn;  //with subgraph number
        temp={unmarked neighbors of u};
    }
    while(temp!=null)
    {
        if(v has neighbour node x in different subgraph)
        if(x has a neighbor node in a different subgraph)
            temp=append(x);
        else
            temp=append(unmarked neighbors of v and x);
    }
}
```

At the end of the sweep, we randomly put the unmarked degree one node in its neighbor subgraph and mark it with the subgraph number. At that point, all nodes are marked, and each node is adjacent to one or two subgraphs (for inner nodes and border nodes, respectively). Then, for any border node which is adjacent to more than one intercommunity edge, we turn to add a zero-weighted node to replace the multiple nodes adjacent to it.

In Fig 3(a), node b1 has three intercommunity edges which makes the partitioning very chaotic. When a node is added, say b1' in its neighbor subgraph and linking up the endpoints of the original intercommunity edges via b1', we get a modified graph where each border node is connected to only one intercommunity edge, as shown in Fig 3(b) .

Phase2. Clustering

During this phase, the subgraph clustering process is performed on the graph partition produced in the phase 1. This is carried out in order to reduce the ratio R_i and regulating the subgraph size to $[\delta_L, \delta_U]$.

Algorithm2 : Sub graph Clustering

```
Set Sg                         #set of subgraphs
{G1(V1,E1),....,Gk(Vk,Ek)}
Set tdi                        #Traversing distance
set for each subgraph 1<=i<=k where k is the
number of subgraphs of graph G
int i                          #present node
int j                          #neighboring node
Ri                             #ratio between
maximum traversing distance and the minimum
traversing distance

clustering()
{

    for (each subgraph Sgi in graph G)
    {
        computer Ri;
        add to set tdi;
    }

    for (each subgraph Sgi in graph G)
    {
        degradationi=merge(Gi,Gj);
        if(degradationi is maximal)
        {
            combine(Gi,Gj);
            update neighbor
    information;
            computer Rj;
        }
    }

    compute Rcom for the combined subgraph
    Gcom;
}
```

The ratio R_i is calculated for each subgraph. At the end of the loop, the traversing distance set for each subgraph, and the ratio R_i is calculated. Then, for each subgraph the degradation of R_i is calculated by merging a neighboring subgraph. At the end, the constructed community edges are employed to facilitate the computation of the traversing distance ratio R_{com} for a combined subgraph G_{com}.

4. Evaluation

When the algorithm is evaluated for a generalized road network, it is seen that the average traversing distance ration follow a downward trend as the subgraph clusters, though it may fluctuate slightly.

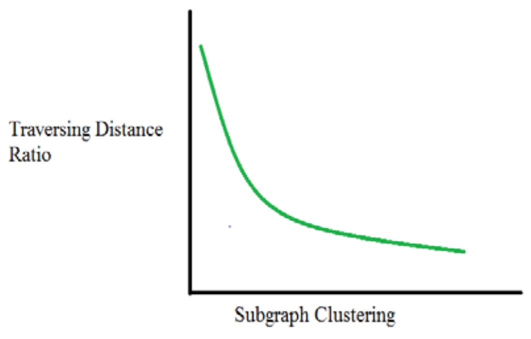

Figure4. Plot between Traversing Distance ratio R and the subgraph clustering

5. Conclusion

In this paper, we propose an effective graph partition method for accelerating the path queries on large node weighted networks. We propose a new regulated distance algorithm based on our hierarchical graph model, which could compute optimal subgraph partitioning in both static and dynamic environments. The proposed method can also be applied to edge-weighted graphs through several conversions and is focused in another piece of our work. As part of future research, it would be beneficial to quantify the effect of the number of subgraphs and the average traversing distance ratio on the performance improvement of a query algorithm so as to determine the optimum values. Also, it is worth developing more fast and effective partition methods to further reducing the traversing distance ratio.

6. References

[1] E. W. Dijkstra, “A note on two problems in connexion with graphs”, Numer. Math., vol. 1, pp. 269-271, 1959.
[2] R. Möhring, H. Schilling, B. Schütz, D. Wagner, and T. Willhalm, “Partitioning graphs to speed up Dijkstra’s algorithm”, ACM J. Exp. Algor., vol.11, article no.2.8, pp. 1-29, 2006.
[3] Y. W. Huang, N. Jing, and E. A. Rundensteiner, “Effective graph clustering for path queries in digital map database”, in Proc. CIKM, Rockville, 1996, pp. 215-222.
[4] S. Jung and S. Pramanik,“An efficient path computation model for hierarchically structured topographical road maps”, IEEE Trans. Knowl. Data Eng.,vol. 14, no. 5, pp. 1029-1046, 2002.
[5] J. Maue, P. Sanders, and D. Matijevic, “Goal directed shortest path queries using precomputed cluster distances”, ACM J. Exp. Algor., vol.14, article no.3.2, pp. 1-27, 2009.
[6] G. Karypis and V. Kumar,“A fast and high quality multilevel scheme for partitioning irregular graphs”, SIAM J. Sci. Comput., vol. 20, no. 1, pp. 359-392, 1998.
[7] D Gusfield, "Partition-distance : A problem and class of perfect graphs arising in clustering", 2002
[8] M. B. Habbal, H. N. Koutsopoulos, and S. R. Lerman, “A decomposition algorithm for the all-pairs shortest path problem on massively parallel computer architectures”, Transp. Sci., vol. 28, no. 4, pp. 292-308, 1994.
[9] Huaijun Qiu, Edwin R.Hancock, "Graph matching and clustering using spectral partitions", 2005.
[10] Rodney Michael Miles, "Graph Clustering with an Emphasis on Algorithms Employing the Commuting Times Distance", 2013
[11] Philippe Gambette, Alain Guenoche, "Bootstrap Clustering for Graph Partitioning", 2001

11. Authors

Anusha Swaminathan is also pursuing her B.E in Computer Science Engineering at Meenakshi Sundararajan Engineering College which is affiliated to the Anna University of Chennai, Tamil Nadu, India. She has presented several papers and also won many accolades for the same. She specializes in Algorithms and programming.

Varsha Alangar is currently pursuing her B.E in Computer Science Engineering at Meenakshi Sundararajan Engineering College which is affiliated to the Anna University of Chennai, Tamil Nadu, India. She has presented several papers on networking including “Resource allocation in Wireless Mesh networks” and other papers related to Operating systems and Cloud computing .