SemiClustering

PREGEL Based Semi-Clustering Algorithm Implementation

Introduction

What is Semi-Clustering

The main concept of Semi-Clustering algorithm on top of social graphs are:

1. Vertices in a social graph typically represent people, and edges represent connections between them. 2. Edges may be based on explicit actions (e.g., adding a friend in a social networking site), or may be inferred from people’s behaviour (e.g., email conversations or co-publication). 3. Edges may have weights, to represent the interactions frequency or strength. 4. A semi-cluster in a social graph is a group of people who interact frequently with each other and less frequently with others. 5. What distinguishes it from ordinary clustering is that, a vertex may belong to more than one semi-cluster.

Bulk Synchronous model proposes it's own smart way of parallelization of programs. We can specify input path for problem and number of peers. Framework reads the input and divides it between peers. Peers can be a processors, threads, separate machines, different items of cloud. BSP algorithm is divided in sequence of supersteps. Barrier synchronization of all peers is made after each superstep. The implementation of BSP(Apache Hama) contains primitives for defining peer number, communication with other peers with different communication primitives, optimizations of communication between peers, also it inherits most of features and approaches of Hadoop project.

Hama Vertex API

Writing a Hama graph application involves subclassing the predefined Vertex class. Its template arguments define three value types, associated with vertices, edges, and messages.The user overrides the Compute() method, which will be executed at each active vertex in every superstep. Predefined Vertex methods allow Compute() to query information about the current vertex and its edges, and to send messages to other vertices. Compute() can inspect the value associated with its vertex via GetValue().

Project Description Parallel greedy Semi-Clustering Algorithm

1. Algorithm input is a weighted, undirected graph. 2. Its output is at most Cmax semi-clusters containing at most Vmax vertices.

Algorithm description

Vertex Of The Graph

Each vertex V maintains a list containing atmost Cmax semi-clusters, sorted by score. Each semi- cluster can be a class which implements WritableComparable interface and the class contains 3 fields.

1. Semi Cluster Id : Unique Id fo a cluster 2. Cluster score : Score of the semi cluster calculated from the above formula 3. Vertex List : List of vertices in the semi-cluster

Basic Execution Steps

Superstep 0 : Vertex V enters itself in that list as a semi-cluster of size 1 and score 1, and publishes itself to all of its neighbours. In subsequent Supersteps:

1. Vertex V iterates over the semi-clusters c1 ,...,ck sent to it on the previous superstep. If a semi-cluster c does not already contain V , and Vc < Mmax , then V is added to c to form c' . 2. The semi-clusters c1 , ..., ck , c'1 , ..., c'k are sorted by their scores, and the best ones are sent to V’s neighbours. 3. Vertex V updates its list of semi-clusters with the semi- clusters from c1 , ..., ck , c'1 , ..., c'k that contain V

Algorithm Termination Condition:

1. The algorithm terminates either when the semi-clusters stop changing or (to improve performance) when the number of supersteps reaches a user-specified limit. 2. At that point the list of best semi-cluster candidates for each vertex may be aggregated into a global list of best semi-clusters.

Semi-Cluster Score Calculation

A semi-cluster c is assigned a score Sc Sc = (Ic fB Bc)/(Vc (Vc 1)/2)

1. Ic is the sum of the weights of all internal edges. 2. Bc is the sum of the weights of all boundary edges. 3. Vc is the number of vertices in the semi-cluster. 4. fB , the boundary edge score factor, is a user-specified parameter, usually between 0 and 1. 5. The score is normalized ,divided by the number of edges in a clique of size Vc , so that large clusters do not receive artificially high scores.