**PS-MCL**

**Developer Guide**

v 1.0.0

Contents

1. **Overview**2
   1. General Information
   2. Document Version
2. **Installation**3
   1. Environment
   2. Install
3. **Running**3
   1. Input
   2. Running algorithms with scripts
   3. Outuput
4. **Source Codes**4
   * 1. List of Source Codes4
   1. Class Description
      1. MCL5
      2. Node7
      3. Edge8
      4. Graph9
      5. Matrix10

1. Overview
   1. General Information

PS-MCL : Parallel Shotgun coarsening Markov Clustering

Version: 1.0.0

Date: Aug. 17th, 2015

Authors: InJae Yu, Yongsub Lim, and U Kang

* 1. Document Version

|  |  |  |
| --- | --- | --- |
| **Version** | **Content** | **Author** |
| 1.0 | First draft of guide | InJae Yu |

1. Installation
   1. Environment

To run PS-MCL, Java should be installed in your system in advance.

* Java: Java 1.8.x or greater version is required.
  1. Install

Before you use it, please check that the script files is executable. If not, you may manually modify the permission of scripts or you may type "make install" to do the same work.

1. Running
   1. Input

To run PS-MCL, you have to prepare an undirected graph edge file directory. It should be “src node \t dst\_node”

* 1. Running algorithms with scripts

The running algorithms and scripts are the followings:

|  |  |
| --- | --- |
| **Running Script** | **Demo Script** |
| PS-MCL | do\_PS-MCL.sh |
| MCL |  |
| R-MCL |  |
| B-MCL |  |

./PS-MCL [Input Graph File Path] [Output Directory] [CoarseMode] [Coarse Level] [Balance Factor] [MCL Mode] [Number of Thread] [Epsilon]

- Input: graph file path

- Output Directory : directory for output

- CoarseMode: Type of coarsen strategy. -sc or –hem.

- Balance Factor: Balance factor of B-MCL

- MCL Mode : Type of MCL. –reg or –basic

- Number of Thread : # of threads to use. Default is 1.

- Epsilon : Estimation of error

MCL : ./MCL [INPUT (Graph File)] [Output Directory] [epsilon]

R-MCL : ./R-MCL [INPUT (Graph File)] [Output Directory] [Coarse Level] [epsilon]

B-MCL : ./B-MCL [INPUT (Graph File)] [Output Directory] [Coarse Level] [Balance Factor] [epsilon]

* 1. Output

DataName\_AlgorithmName\_CoarseningInfo\_ThreadInfo.result : time, NCut, # of threads

DataName\_AlgorithmName\_CoarseningInfo\_ThreadInfo.assign : cluster assignment of each node

DataName\_AlgorithmName\_CoarseningInfo\_ThreadInfo.dist : cluster size distribution of the graph

1. Source Codes
   1. List of Source Codes

Here is the list of source codes and their functions.

|  |  |
| --- | --- |
| **Source Code** | **Function** |
| **mcl package** | MCL main package |
| MCL.java | MCL main class |
| **graph package** | Graph package for MCL |
| Edge.java | Edge class |
| Node.java | Node class |
| Graph.java | Graph class |
| **matrix package** | CSC format matrix package |
| CSCMatrix.java | CSC format matrix |

* 1. Class Description
     1. MCL

|  |
| --- |
| **Main Control flow of MCL** |
| Build graph **g** from edge file  Coarse g N times  M : adj matrix of coarsend graph g  \\first loop  Until g reaches to initial graph  Mr : adj matrix of coarsened graph g  MatirxMult\_Inf\_Prun(M, Mr) – 4 times  Recover g 1 time  M : projectFlow(M)  \\second loop  Until the error converges to epsliion  Ms : MatrixMult\_Inf\_Prun(M, Mr)  Epsilon : 1 norm of (M – Ms)  Write result |

* run: do main operation of MCL

|  |  |
| --- | --- |
| **Funtion** | **run** |
| **Input** | fileName - path of file  coarsen\_mode - coarsening strategy  coarseLevel - coarsening level  bFactor - balance factor.  mcl\_mode - mcl mode  thread - the number of thread  epsilon - A number for checking convergence |
| **Operation** | do main operation of MCL described as above |
| **Output** | json array of clusters |

* matrixMultInfPrune: Matrix multiplication, inflation, pruning by thread

|  |  |
| --- | --- |
| **Funtion** | **matrixMultInfPrune** |
| **Input** |  |
| **Operation** | Matrix multiplication, inflation, pruning by thread |
| **Output** |  |

* prepareBMCL : Calculate the propensity vector

|  |  |
| --- | --- |
| **Funtion** | **matrixMultInfPrune** |
| **Input** |  |
| **Operation** | Calculate the propensity vector |
| **Output** |  |

* buildCSC: Build CSC Format matrix from thread results

|  |  |
| --- | --- |
| **Funtion** | **buildCSC** |
| **Input** |  |
| **Operation** | Build CSC Format matrix from thread results |
| **Output** |  |

* projectFlow: From the curGraph, expand the flow matrix into bigger size matrix

|  |  |
| --- | --- |
| **Funtion** |  |
| **Input** |  |
| **Operation** | From the curGraph, expand the flow matrix into bigger size matrix |
| **Output** |  |

* prunThreshold: From given vector v and parameters, calculate the prune threshold

|  |  |
| --- | --- |
| **Funtion** |  |
| **Input** | column - inflated column vector |
| **Operation** | From given vector v and parameters, calculate the prune threshold |
| **Output** | prune threshold |

* calcNorm: Calculate the one-norm of subtraction of previous and current state of the matrix

|  |  |
| --- | --- |
| **Funtion** | **calcNorm** |
| **Input** |  |
| **Operation** | Calculate the one-norm of subtraction of previous and current state of the matrix |
| **Output** |  |

* Ncut: Calculate the NCut of the cluster output

|  |  |
| --- | --- |
| **Funtion** | Ncut |
| **Input** | cluster - input cluster |
| **Operation** | Calculate the NCut of the cluster output |
| **Output** | NCut of a cluster |

* problemRange: Indicate the range of subproblem to each thread

|  |  |
| --- | --- |
| **Funtion** | **problemRange** |
| **Input** | col - size of column |
| **Operation** | Indicate the range of subproblem to each thread |
| **Output** | split the problem of each thread should cover |

* main: Get input, and run the algorithm.

|  |  |
| --- | --- |
| **Funtion** | **main** |
| **Input** | args - parameters of algorithm |
| **Operation** | Get input, and run the algorithm. |
| **Output** |  |

* + 1. Node
* getSuperNode : Return the superNode of this node

|  |  |
| --- | --- |
| **Funtion** | **getSuperNode** |
| **Input** |  |
| **Operation** | Return the superNode of this node |
| **Output** | superNode |

* getWeight : Return weight of this node

|  |  |
| --- | --- |
| **Funtion** | **getWeight** |
| **Input** |  |
| **Operation** | Return weight of this node |
| **Output** | weight |

* getIndex :

|  |  |
| --- | --- |
| **Funtion** | **getIndex** |
| **Input** |  |
| **Operation** | Return index of this node |
| **Output** | index |

* insertEdge : Insert an edge between "this" node and "to" node with weight

|  |  |
| --- | --- |
| **Funtion** | **insertEdge** |
| **Input** | to - insert a new edge between this node  weight - weight of the edge |
| **Operation** | Insert an edge between "this" node and "to" node with weight |
| **Output** |  |

* addEdge : Insert an edge to this node

|  |  |
| --- | --- |
| **Funtion** | **addEdge** |
| **Input** | e - edge to be add |
| **Operation** | Insert an edge to this node |
| **Output** |  |

* getAdj\_list: Return adj\_list of this node

|  |  |
| --- | --- |
| **Funtion** | **getAdj\_list** |
| **Input** |  |
| **Operation** | Return adj\_list of this node |
| **Output** | adjacency list |

* matchingNode:

|  |  |
| --- | --- |
| **Funtion** | **matchingNode** |
| **Input** | coarse\_mode - coarsening mode |
| **Operation** | Find the contracting edge and insert this node to superNode |
| **Output** |  |

* degree : Return the size of adjacency list

|  |  |
| --- | --- |
| **Funtion** | **degree** |
| **Input** |  |
| **Operation** | Return the size of adjacency list |
| **Output** | # of incident edges |

* addWeight : Increase the weight of this node by weight of node v

|  |  |
| --- | --- |
| **Funtion** | **addWeight** |
| **Input** | node - node to be merged |
| **Operation** | Increase the weight of this node by weight of node v |
| **Output** |  |

* removeEdge : Remove e from this node's adjacency list

|  |  |
| --- | --- |
| **Funtion** | **removeEdge** |
| **Input** | e - edge to be removed |
| **Operation** | Remove e from this node's adjacency list |
| **Output** |  |

* + 1. Edge
* isLoop : Return true if the edge is loop.

|  |  |
| --- | --- |
| **Funtion** | **isLoop** |
| **Input** |  |
| **Operation** | Return true if the edge is loop. |
| **Output** | True if this edge is loop |

* getFrom :

|  |  |
| --- | --- |
| **Funtion** | **getFrom** |
| **Input** |  |
| **Operation** | Return from\_node |
| **Output** | from\_node |

* getTo : Return to\_node

|  |  |
| --- | --- |
| **Funtion** | **getTo** |
| **Input** |  |
| **Operation** | Return to\_node |
| **Output** | to\_node |

* getWeight : Return the weight of this edge

|  |  |
| --- | --- |
| **Funtion** | getWeight |
| **Input** |  |
| **Operation** | Return the weight of this edge |
| **Output** | edge\_weight |

* addWeight : Add weight to this edge by w

|  |  |
| --- | --- |
| **Funtion** | **addWeight** |
| **Input** | w - weight |
| **Operation** | Add weight to this edge by w |
| **Output** |  |

* incident: Return the incident node that is not equal with from\_node

|  |  |
| --- | --- |
| **Funtion** | **incident** |
| **Input** | from\_node |
| **Operation** | Return the incident node that is not equal with from\_node |
| **Output** | incident\_node |

* isUntouched : Return true if this edge is not visited

|  |  |
| --- | --- |
| **Funtion** | **isUntouched** |
| **Input** |  |
| **Operation** | Return true if this edge is not visited |
| **Output** | is\_untoched |

* remove : Remove this node from neighbor nodes

|  |  |
| --- | --- |
| **Funtion** | **remove** |
| **Input** |  |
| **Operation** | Remove this node from neighbor nodes |
| **Output** |  |

* + 1. Graph
* getMap : Node set of the graph

|  |  |
| --- | --- |
| **Funtion** | **getMap** |
| **Input** |  |
| **Operation** | Node set of the graph |
| **Output** | Node set of the graph |

* size : size of the graph

|  |  |
| --- | --- |
| **Funtion** | **size** |
| **Input** |  |
| **Operation** | size of the graph |
| **Output** | size of the graph |

* Coarse : coarsened graph

|  |  |
| --- | --- |
| **Funtion** | Coarse |
| **Input** | coarse\_mode - HEM or SC |
| **Operation** | Coarse an graph |
| **Output** | coarsened graph |

* + 1. CSCMatrix
* getVal : value array

|  |  |
| --- | --- |
| **Funtion** | getVal |
| **Input** |  |
| **Operation** | value array |
| **Output** | value array |

* getRow\_ind :

|  |  |
| --- | --- |
| **Funtion** | **getRow\_ind** |
| **Input** |  |
| **Operation** | row index array |
| **Output** | row index array |

* getCol\_ptr : column\_ptr array

|  |  |
| --- | --- |
| **Funtion** |  |
| **Input** |  |
| **Operation** | column\_ptr array |
| **Output** | column\_ptr array |

* getMax\_col\_size : maximum column size

|  |  |
| --- | --- |
| **Funtion** | getMax\_col\_size |
| **Input** |  |
| **Operation** | maximum column size |
| **Output** | maximum column size |