User Guide

1. Deployment

Step 1

Excute 'compile.sh' for compilation. When the process is finished the binaries 'mwis' can be found in the root directory.

Step 2

Run the executable file 'mwis' with the following options and parameters.

Option	**Parameter**	**Description**
-r	filepath	Read the input file from the 'filepath'
-o	filepath	Write the output file to the 'filepath'
-w		Choose how the algorithm run, reduce only or run the whole
-t	timelimit	Set the time limit in seconds. (Default: 1000s)
-h		Print helps

2. I/O Format

Input File Format

A graph $G = (V, E)$ with n vertices and m edges is stored in a plain text file that contains $n + 1$ lines. The input graph has to be undirected, without self-loops and without parallel edges. The first line contains two integers

$$n \ m$$

The remaining n lines store information about the actual structure of the graph. In particular, the i-th line contains information about the i-th vertex v_i, in the form of

$$w_i \ u_{i1} \ u_{i2} \ \dots \ u_{ik}$$

where w_i is the weight associated with v_i and $u_{i1} \dots u_{ik}$ are the neighbors of v_i.

Output File Format

There are two kinds of output files depending on the algorithm types we choose. If we only reduce the graph, the output is a graph obtained from the input by reductions. The output file contains $n + 2$ lines, where n is the number of vertices in the output. The first $n + 1$ lines has same format wiith the input file. The last line contains two integers

$$w_1 \ w_2$$

where w_1 is the weight got by reductions and w_2 is the total weight of the output.

Otherwise, we run the whole algorithm. Now, the output is an independent set found by the algorithm. The set is stored in a plain text that contains two lines. The first line contains two integer

$$k \ w$$

where k is the size of the set and w is the total weight of the set. The second line contains the vertices of the set in the form of

$$u_1 \ u_2 \ \ldots \ u_k$$

3. Algorithmic Recap

Our algorithm is based on a branch-and-reduce framework, where reductions and branches are excuted iteratively. Following previous works (refs to be added), we design some powerful reductions to improve the experimental performence significantly.

In reality, almost all graphs follow power-law distribution. It means almost graphs are sparse since there are many low-degree vertices. When solving MIS, a important reason that the exsiting algorithm can runs efficiently is that low-degree verices can be reduced, especially degree-2 vertices (all can be reduced). However, most reductions for MIS can not hold for MWIS, including the low-degree reductions.

Here, we introduce some new reductions that can works for low-degree graph structure. Almost all our reductions can be explained by a meta-reduction, named by unconfined reduction.

Unconfined Reduction. Let S be a set contained in all maximum weighted independent set. Then, for any independent set $S' \subseteq N(S)$, it holds that

$$w(S') < w(S \cap N(S')) + w(S''),$$

where S'' is some independent set of $G[N(S') \setminus N[S]]$. Furthermore, if S'' is unique, then $S \cup S''$ is must contained in all maximum weighted independent set.

The first condition is called termination and the second one is named by expansion. Use this two condition, for any vertex v, either v is terminated when there is a maximum weighte indepdent set not containing v or finding a set S_v confining v by expansion. The set S_v is said to be confining v if there is a maximum weighted independent either containing S_v or not containing v.

This reduction is too complex to apply in practice. So, our reuctions are obtained from this by relaxtion and some special cases. For more details, you can read the paper to be published.